Grounding LLMs as Autotelic Reinforcement Learning Agents

By **Clément ROMAC**

Under the co-supervision of **Thomas Wolf** and **Pierre-Yves Oudeyer**

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

University of Bordeaux

Graduate school of Mathematics and Computer Science

Major in Computer Science

Submitted on October 21, 2025. Defended on January 13, 2026.

Composition of the jury:

Dr. Ellie PAVLICK	Associate Professor	Brown University / Google DeepMind	Reviewer
Dr. Prithviraj AMMANABROLU	Assistant Professor	Univ. of California San Diego / Nvidia	Reviewer
Pr. Hugo LAROCHELLE	Professor	University of Montréal / MILA	Examiner
Dr. Timothy LILLICRAP	Research Scientist	University College of London / Google DeepMind	Invited
Pr. Matthieu CORD	Professor	Sorbonne University / Valeo	President
Dr. Pierre-Yves OUDEYER	Research Director	INRIA	Co-advisor
Dr. Thomas WOLF	Chief Science Officer	Hugging Face	Co-advisor

Ancrage des LLMs comme agents autoteliques d'apprentissage par renforcement

Résumé : La construction de machines capables de traiter et comprendre le langage naturel est un objectif historique de l'intelligence artificielle (IA). Récemment, les approches distributionnelles par réseaux neuronaux profonds, en particulier les grands modèles de langage (LLMs), ont permis des avancées spectaculaires. Ces modèles, entraînés à générer du texte en imitant de vastes corpus issus d'Internet, reposent toutefois sur un paradigme purement statistique, dont les limites sont de plus en plus mises en lumière. Cela contraste fortement avec l'acquisition du langage chez l'humain, profondément ancrée dans l'interaction sensorimotrice et sociale. Le langage humain est acquis pour atteindre des objectifs, dans un cadre fonctionnel, guidé par la motivation intrinsèque et la curiosité.

Dans cette thèse, nous explorons comment rapprocher les LLMs des théories développementales de l'acquisition du langage, en les incarnant comme des agents curieux capables d'apprendre par renforcement via l'interaction avec un environnement.

Nous introduisons d'abord le concept d'ancrage fonctionnel : l'alignement des représentations internes d'un agent avec un environnement externe afin d'agir efficacement. Pour cela, nous proposons GLAM, une méthode d'apprentissage par renforcement en ligne qui entraîne les LLMs dans des environnements textuels. GLAM améliore significativement leur compétence fonctionnelle — c'est-à-dire leur capacité à utiliser le langage pour atteindre des buts. Une analyse approfondie montre que combiner cet ancrage avec des contextes variés et un apprentissage contrastif en augmente la robustesse. Nous abordons ensuite la modélisation du monde avec WorldLLM, un cadre dans lequel les LLMs génèrent et affinent des théories en langage naturel à partir d'interactions curieuses, améliorant leurs capacités prédictives.

Nous étendons ensuite ce cadre à des environnements complexes, où les objectifs possibles sont nombreux ou ouverts. Inspirés par l'apprentissage autotelique humain — où l'on choisit et poursuit ses propres buts — nous proposons SAC-GLAM, une extension de GLAM combinant apprentissage off-policy et réétiquetage a posteriori, afin de mieux exploiter des signaux de récompense rares ou bruités. Nous traitons ensuite le problème de la sélection d'objectifs via le progrès en apprentissage, et introduisons MAGELLAN, un module métacognitif permettant aux LLMs d'estimer leur compétence et de prioriser les objectifs les plus bénéfiques. MAGELLAN structure l'exploration des modèles en tenant compte des relations sémantiques et des dynamiques sous-jacentes entre les buts. Enfin, nous montrons que ces capacités métacognitives permettent également aux LLMs de reconnaître leurs limites et de demander de l'aide extérieure lorsqu'ils ne sont pas compétents.

Cette recherche montre que l'incarnation des LLMs comme agents autotelique ouvre des perspectives fortes pour le développement de modèles de langage ancrés, adaptatifs et auto-améliorables. Elle constitue une avancée vers des modèles capables non seulement d'utiliser le langage de manière fonctionnelle, mais aussi de mieux comprendre le monde et leur propre fonctionnement. Néanmoins, de nombreuses questions restent ouvertes et atteindre un ancrage fonctionnel dans notre monde physique et social demeure un défi majeur pour la prochaine génération de systèmes intelligents.

Mots-clés : intelligence artificielle, modèles de langage, apprentissage par renforcement, motivation intrinsèque, curiosité artificielle, acquisition du langage

---

Unité de Recherche
INRIA, 33000 Bordeaux, France.

Grounding LLMs as Autotelic Reinforcement Learning Agents

Abstract: Building machines capable of processing and understanding natural language has long been a central goal of artificial intelligence (AI). In recent years, distributional approaches based on deep neural networks have dominated the field. In particular, large language models (LLMs)—trained to generate text by imitating large internet corpora—have driven remarkable progress. However, their purely statistical learning paradigm faces growing criticism. In contrast, developmental sciences emphasize that human language acquisition is grounded in sensorimotor experience and social interaction. Humans acquire language by interacting with the physical and social world and use it in functional, goal-directed ways—driven by intrinsic motivation and curiosity.

This thesis explores how to bridge the gap between LLMs and developmental theories of language acquisition by embodying LLMs as curiosity-driven reinforcement learning (RL) agents capable of learning from interaction. We begin by introducing the concept of functional grounding: aligning an agent's internal representations with the external environment to enable prediction, control, and goal achievement. To this end, we propose GLAM, an online RL-based approach that trains LLMs through interaction with a textual environment. We show that GLAM significantly improves functional competence—that is, the model's ability to use language effectively to achieve goals. A follow-up analysis identifies limitations (e.g., sensitivity to prompt format) and shows that combining grounding with varied contexts and contrastive learning increases robustness. To enhance predictive capabilities, we also introduce WorldLLM, a framework where LLMs generate and refine natural language theories through curiosity-driven interaction, improving their world modeling abilities.

We then explore functional grounding in more complex environments involving an open-ended set of goals. Inspired by human development, we adopt an autotelic learning approach—where agents generate, choose, and pursue their own goals. We first address the challenge of sample efficiency with SAC-GLAM, an extension of GLAM that incorporates off-policy RL and hindsight relabeling to better learn from sparse or noisy rewards. Next, we tackle goal selection by extending Learning Progress (LP) methods to language-defined goals. We introduce MAGELLAN, a metacognitive module that enables LLMs to estimate their competence and prioritize goals that maximize LP. MAGELLAN structures exploration by capturing semantic and dynamic relationships between goals. We show that metacognitive abilities help LLMs not only decide what to learn next but also recognize when help is needed and seek external assistance.

Altogether, this research demonstrates that embodying LLMs as autotelic agents opens new avenues for creating grounded, adaptive, and self-improving language models. It offers concrete steps toward building LLMs that not only use language functionally but also learn to model the world and themselves—through metacognitive capabilities. Yet, much remains to be done, and achieving functional grounding in our physical and social world remains a major challenge for the next generation of intelligent systems.

Keywords: artificial intelligence, language models, reinforcement learning, intrinsic motivation, artificial curiosity, language acquisition

Unité de Recherche
INRIA, 33000 Bordeaux, France.

Acknowledgments

This manuscript marks the completion of a rather crazy idea we once discussed, many years ago, in a small room at our university. To Pierre, Vincent, Cédric, and Axel.

From that idea to the official start of this PhD, the path was far from straight. To the people I met along the way. To Patrick, Hayssam, Damien, Nathanaël, Pierre-Yves.

I was incredibly lucky to join the Flowers team five years ago. I enjoyed every single day spent with this group. To Rémy, Tristan, Cédric, Laetitia, Mayalen, Grgur, Gautier, Alex, Rania, Isabeau, Mathieu, Jesse, Zachary, Chloé, Olivier, Marion, Cécile, Hélène, Juliette, Clément, Loris, Guillaume, Guillaume, Jérémy, Marko, Sina, Nicolas... In particular, to Thomas. For the debates, the discussions, the late-night coding sessions, and more broadly, our collaboration. From this collaboration, I was also fortunate to receive support from your advisors. To Sylvain and Olivier.

I was equally lucky to join Hugging Face during this PhD. The list of brilliant and kind people I met there would be far too long to write down. To Thomas, Thomas, Quentin, Edward, Lewis, Quentin, Dylan, Gloria...

And of course, to my two advisors. For their support. For their trust. For everything. To Thomas and Pierre-Yves.

Finally, to those who have always been there. To my family and friends.

Contents

1 Introduction **1**
- 1.1 Usage-based language acquisition . . . 3
  - 1.1.1 The symbol grounding problem . . . 4
- 1.2 Statistical approaches in NLP . . . 4
  - 1.2.1 From Transformers to LLMs . . . 5
  - 1.2.2 LLMs are passive learners . . . 6
- 1.3 Towards grounded LLMs through interactions . . . 6
  - 1.3.1 Existing models of grounded language understanding . . . 7
  - 1.3.2 Building embodied LLM agents . . . 8
  - 1.3.3 Contributions . . . 9
  - 1.3.4 Collaborations . . . 11
  - 1.3.5 Publications . . . 12

2 Foundations: Language and agents **13**
- 2.1 Computational models of language understanding . . . 13
  - 2.1.1 Symbolic approaches . . . 14
  - 2.1.2 Statistical approaches . . . 18
- 2.2 Computational models of action learning . . . 24
  - 2.2.1 The agent-environment framework . . . 25
  - 2.2.2 Reinforcement learning . . . 27
  - 2.2.3 From curiosity-driven to autotelic RL agents . . . 30
  - 2.2.4 Language in RL agents . . . 32
- 2.3 LLM agents . . . 34
  - 2.3.1 Embodied LLMs . . . 34
  - 2.3.2 RL applied to LLMs . . . 36
  - 2.3.3 Building autotelic LLM agents . . . 38

3 Functional grounding of LLMs through online RL **40**
- 3.1 Functional grounding: external dynamics and goals . . . 42
- 3.2 GLAM: Functional grounding of LLMs in interactive environments with online RL . . . 43
  - 3.2.1 Related work . . . 44
  - 3.2.2 Grounding LLMs with online RL (GLAM) . . . 46
  - 3.2.3 Experiments . . . 48

3.2.4 How fast can an LLM adapt and learn to solve tasks? (Q1) 50
3.2.5 Q2. Generalization to new objects 53
3.2.6 Q3. Generalization to new tasks 54
3.2.7 What is the impact of using RL vs BC for grounding? (Q4) 55
3.2.8 Conclusion 55
3.3 An analysis of functionally grounded representations and knowledge in LLMs 56
3.3.1 Related work 57
3.3.2 Problem statement and methods 58
3.3.3 Contrastive learning 59
3.3.4 Experimental Protocol 60
3.3.5 Task-solving abilities with respect to prompt formulation 61
3.3.6 Analyzing the functionally grounded latent representations 62
3.3.7 Environmental knowledge acquired through functional grounding . 65
3.3.8 Conclusion 66
3.4 Discussion 67

4 Moving beyond control: World modeling 69
4.1 World models 70
4.1.1 World models in RL agents 70
4.1.2 Mental models with theories 72
4.2 WorldLLM: Grounding LLMs' world modeling using curiosity-driven theory making 73
4.2.1 Improving LLMs' world model with WorldLLM 74
4.2.2 Experiments 78
4.2.3 Related work 84
4.3 Discussion 85

5 Towards functional grounding in complex goal spaces 88
5.1 Social interactions for autotelic functional grounding: hindsight relabeling and off-policy RL with SAC-GLAM 91
5.1.1 SAC-GLAM 91
5.1.2 Experiments 94
5.1.3 Conclusion 97
5.2 MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces 98
5.2.1 Related Work 100
5.2.2 Methods 101
5.2.3 Experiments 106
5.2.4 Conclusion 111
5.3 When goals are beyond reach: Metacognitive monitoring guides autonomous discovery of frugal assistance-seeking in LLMs 112
5.3.1 Related Work 114
5.3.2 Methods 116
5.3.3 Experiments 118
5.3.4 Conclusion 124
5.4 Discussion 124

6 Discussion **127**
6.1 Summary of contributions . . . 127
6.2 Perspectives . . . 129
6.2.1 From language models to action models . . . 129
6.2.2 On functional grounding and reasoning . . . 130
6.2.3 Beyond reinforcement learning . . . 131
6.2.4 Towards self-generated exercises . . . 131
6.2.5 Towards LLMs capturing a causal model of the world . . . 133
6.2.6 Metacognition, alignment, and safety . . . 133
6.2.7 The quest of autotelic agents . . . 134
6.3 Conclusion . . . 136

Appendices **137**

A GLAM **138**

B Functionally grounded representations and knowledge in LLMs **162**

C WorldLLM **174**

D SAC-GLAM **185**

E MAGELLAN **190**

F Autonomous discovery of frugal assistance-seeking in LLMs **215**

G Jack of All Trades, Master of Some, a Multi-Purpose Transformer Agent **231**

H TeachMyAgent: a Benchmark for Automatic Curriculum Learning in Deep RL **260**

Bibliography **308**

ACL	Automatic Curriculum Learning
AI	Artificial Intelligence
ALP	Absolute Learning Progress
BC	Behavioral Cloning
CLM	Causal Language Modeling
LLM	Large Language Model
LP	Learning Progress
MDP	Markov Decision Process
ML	Machine Learning
MLM	Masked Language Modeling
MLP	Multi-Layer Perceptron
NLG	Natural Language Generation
NLP	Natural Language Processing
NMT	Neural Machine Translation
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network
TD	Temporal Difference
VLM	Vision Language Model

Chapter 1

Introduction

Contents

1.1 Usage-based language acquisition 3
1.1.1 The symbol grounding problem 4
1.2 Statistical approaches in NLP 4
1.2.1 From Transformers to LLMs 5
1.2.2 LLMs are passive learners 6
1.3 Towards grounded LLMs through interactions 6
1.3.1 Existing models of grounded language understanding 7
1.3.2 Building embodied LLM agents 8
1.3.3 Contributions 9
1.3.4 Collaborations 11
1.3.5 Publications 12

Developing machines capable of understanding human language and responding appropriately has been a long-standing objective in the field of artificial intelligence (AI). Since the early days of AI, evaluating machine intelligence has often been associated with the machine's ability to process and generate natural language. In his seminal work, Alan Turing proposed a test in which a machine and a human are placed in separate rooms, and a third party—the judge—interacts with both through written communication without knowing which is which (Turing, 1950). If the judge is unable to reliably distinguish the machine from the human based solely on their responses, the machine is considered capable of "thinking." For several decades, Turing's test was regarded as a valid benchmark for machine intelligence. However, subsequent experiments demonstrated that relatively simple rule-based systems, such as ELIZA (Weizenbaum, 1966), could deceive human interlocutors, revealing a tendency to anthropomorphize conversational behavior (Block, 1981). These findings raised a fundamental and still unresolved question: how can we determine whether a machine genuinely understands the meaning of the language it processes?

Answering this question invites us to delve into theories from psychology and linguistics concerning how humans acquire and use language. Contemporary theories of language acquisition, such as usage-based approaches (see the survey from Tomasello (2003)), emphasize the role of language as a developmental tool that supports and emerges

from the child's broader cognitive growth. According to these perspectives, language is not acquired in isolation but develops progressively through interaction, initially serving communicative functions—particularly with caregivers—and facilitating the abstraction and categorization of the complex environment with which the child engages. Over time, language becomes internalized and functions as a cognitive tool to support higher-order processes such as imagination, planning, and prediction (Vygotsky, 1962). This developmental trajectory suggests that language is inherently **grounded** in interaction—both with the physical world through sensorimotor experience and with the social world through interpersonal communication.

Returning to the challenge of designing machines capable of answering natural language questions, the field has advanced significantly since early heuristic systems such as ELIZA. These efforts have coalesced into the domain now known as natural language processing (NLP). Whereas initial approaches were based on hand-crafted rules, the field has since transitioned toward data-driven methodologies, primarily relying on machine learning (ML) techniques. This shift has been driven by several factors, including advancements in ML, developments in language acquisition theories, and the widespread availability of large-scale linguistic data via the internet. By training models to predict the most probable next word in a sequence using massive datasets and billions of parameters, recent years have witnessed remarkable progress in computational language modeling. These models, referred to as large language models (LLMs), generate text sequentially and are now widely deployed. They can process and respond to a variety of human queries, including those involving complex capabilities such as reasoning (Huang et al., 2023; Yao et al., 2022), planning (Ahn et al., 2022; Huang et al., 2022), and commonsense inference (Brown et al., 2020; Chowdhery et al., 2022). Despite their purely statistical nature and lack of grounding in sensory or social experiences, LLMs appear to perform tasks traditionally thought to require semantic understanding. However, they continue to exhibit notable limitations, suggesting that some of their apparent competencies may be superficial (Bender & Koller, 2020; Shojaee et al., 2025). These shortcomings point to the possibility that grounding—through interaction with the physical and social world—may be essential to surpass current performance boundaries.

The present research aims to explore how the gap between grounded theories of language acquisition and contemporary LLMs can be addressed through an embodied approach. This introductory chapter begins by examining theoretical perspectives on language development, with a particular emphasis on the role of sensorimotor experiences in grounding linguistic understanding. We then provide an overview of current computational models of language—namely, LLMs—highlighting both their capabilities and their limitations. Finally, we conclude the chapter by motivating and outlining our research contributions, which center on embedding LLMs in interactive environments using curiosity-driven reinforcement learning (RL) to support embodied and grounded language learning—a direction extensively explored in developmental sciences for modeling language acquisition, but largely overlooked in NLP.

1.1 Usage-based language acquisition

Early theories of language acquisition from linguistics, mostly initiated by Noam Chomsky, proposed the existence of innate grammatical structures underlying all human languages, such as generative grammars (Chomsky, 1957). For decades, linguistic research was primarily oriented toward uncovering these presumed universal, abstract syntactic rules. However, the absence of conclusive empirical evidence supporting the existence of such innate grammars, as well as the emergence of new theories in developmental psychology and cognitive science, motivated linguistics to propose alternative approaches.

In particular, constructivists in psychology proposed theories of the child's cognitive development where language acquisition is tightly bound to sensorimotor and social experiences. Notably, Piaget introduced the concept of schemas—internal representations of concepts that integrate both perceptual inputs and motor programs—to which linguistic labels are attached (Piaget, 1954). As illustrated by Cangelosi et al. (2010), for example, "the word heavy can be understood as grounded in haptic expectations associated with lifting actions". In Piaget's framework, words serve as labels for one's sensorimotor experiences that are abstracted into schemas. Complementing this view, Vygotsky's sociocultural theory of development emphasized the central role of language in cognitive growth (Vygotsky, 1962; Mirolli & Parisi, 2011). Language initially functions as a tool for interpersonal communication—particularly between the child and caregivers—before being gradually internalized to support individual thought processes.

Similarly, numerous studies in cognitive science have highlighted the limitations of classical cognitive theories, which conceptualize the human mind as processing abstract symbols—such as words or grammatical rules—independently of the body (Barsalou, 1999; Glenberg & Robertson, 2000; Wilson, 2002; Pecher & Zwaan, 2005). In contrast, sensorimotor experiences and embodiment have increasingly been recognized as central to cognition, echoing the emphasis made in the constructivist movement. Specifically, the embodied cognition movement advocates a fundamental shift in perspective: rather than viewing the body as merely a support system for a disembodied mind solving abstract problems, it posits that the mind itself has evolved primarily to serve the needs of the body in interacting with and navigating the world (Wilson, 2002).

Given this evidence that cognition and language may emerge solely from one's interactions with their sociocultural environment, usage-based theories of language appeared in linguistics, proposing that language structure emerges from language use (Tomasello, 2003). Rather than positing pre-specified linguistic capacities or universal generative templates, they therefore suggest that language emerges through and supports the child's cognitive development within a rich sociocultural context. Several key principles arise from this perspective. First, grammatical abstractions and the compositional nature of language are seen as emerging from the child's need to conceptualize and structure experiences in a complex environment. Second, statistical learning mechanisms play a critical role in acquiring linguistic patterns. Finally, and most crucial to the present research, language acquisition is fundamentally grounded in both sensorimotor and social interactions. This grounding encompasses not only perceptual experiences (e.g., visual, auditory) but also the motor actions through which the child actively engages with and explores their environment.

1.1.1 The symbol grounding problem

The importance of grounding concepts and language in one's experiences is also central to discussions regarding the notion of *meaning*. To illustrate the limitations of syntactic rule-based theories in explaining how meaning stems from language, Searle introduced the *Chinese Room* thought experiment in his seminal paper *Minds, Brains, and Programs* (Searle, 1980). In this scenario, Searle imagines himself receiving sequences of Chinese characters (i.e., questions) along with a set of syntactic rules for manipulating these symbols based purely on their visual form, despite having no understanding of the Chinese language. By mechanically applying these rules, he is able to produce responses in Chinese that are indistinguishable from those of a native speaker. In such a case, Searle could seemingly pass the Turing Test, yet without any genuine understanding of the language—thereby challenging the notion that syntactic processing alone constitutes understanding.

Expanding on this critique, Stevan Harnad proposed a related thought experiment, in which a person attempts to learn Chinese using only a Chinese-to-Chinese dictionary (Harnad, 1990). In this scenario, each symbol is defined solely in terms of other symbols, resulting in an infinite regress with no anchoring in meaning. Both thought experiments underscore a fundamental issue in symbolic theories of language and cognition: symbols that are manipulated solely based on formal rules lack semantic grounding. These arguments support the view that meaning arises through connections to perceptual and sensorimotor experience. This leads us to a central question:

> *"[...] how are symbols grounded in something else than other symbols to obtain their meaning?"* – The symbol grounding problem Harnad (1990)

I shall conclude this section by defining what I will mean by *grounded* in the present document. Here, grounded will refer to internal representations that are shaped by interactions with an external environment. Accordingly, grounding will denote the process through which such internal representations are formed in response to external stimuli. This interpretation is closely aligned with the definition proposed by Roy (2005b), who describe grounding as "the processes by which an agent relates beliefs to external physical objects." However, the definition adopted in the present work generalizes this view: it does not restrict grounding to beliefs alone—as it may involve various forms of internal representations—nor does it limit the referents to physical objects, as grounding may also pertain to abstract concepts.

1.2 Statistical approaches in NLP

In Section 1.1, I examined the limitations of theoretical frameworks that conceptualize language processing as a purely symbolic, rule-based manipulation of abstract representations—where symbols derive meaning exclusively through their relations to other symbols. I also reviewed a range of perspectives from linguistics, psychology, and cognitive science that converge on the importance of the interplay between sensorimotor experience, internal representation, and language. With this theoretical foundation in

place, we now return to our central focus: the challenge of designing machines capable of understanding language.

Historically driven by rule-based theories such as generative grammars, symbolic approaches dominated the field of NLP until the 1990s. These methods relied on abstract, human-defined rules and symbols to model language. However, the emergence of statistical approaches stemming from machine learning—models composed of parameters optimized through gradient descent—gradually shifted the field toward data-driven techniques. Rather than relying on hand-crafted rules, these methods learn distributed vector representations of words directly from large-scale textual corpora (e.g., books, web pages). This shift was strongly influenced by the theory of distributional semantics, initially proposed by Firth (1957) and Harris (1954), which posits that the meaning of a word can be inferred from the linguistic contexts in which it typically appears. Leveraging this idea, artificial neural networks trained on large corpora to predict the most likely word given its surrounding context were shown to yield rich vector representations of words (Mikolov et al., 2013; Boleda, 2020). These learned embeddings were found to capture not only syntactic properties but also meaningful aspects of semantics (Bengio et al., 2003).

1.2.1 From Transformers to LLMs

In addition to enabling the learning of rich word representations, neural approaches have been successfully applied to directly address core NLP tasks. Among these, machine translation—where the goal is to generate a sentence in a target language given an input sentence in a source language—has played a pivotal role in advancing neural methods (Sutskever et al., 2014; Bahdanau et al., 2014). Early neural machine translation (NMT) systems primarily relied on recurrent neural networks (RNNs), particularly Long Short-Term Memory networks (LSTMs) (Hochreiter & Schmidhuber, 1997). This paradigm shifted with the introduction of the *Transformer* architecture in 2017 (Vaswani et al., 2017), which replaced recurrence with a self-attention mechanism (Bahdanau et al., 2014) and enabled parallel processing of sequences. The transformer's ability to model complex dependencies more efficiently led to substantial improvements in translation performance and quickly established it as the dominant architecture across a wide range of NLP tasks.

Compared to RNNs, large transformer architectures offer significant advantages in terms of training efficiency and scalability. This computational efficiency enabled the training of much larger models on tasks far beyond translation. A key milestone in this evolution was the release of GPT-2 (Radford et al., 2019), which marked a turning point in the field. GPT-2 is a transformer-based model trained using the *causal language modeling* objective—that is, predicting the most probable next word given a preceding sequence—on an extensive corpus of text scraped from the internet. The study demonstrated that scaling up model size and training data could yield models capable of impressive linguistic competence. These models, soon referred to as large language models (LLMs), exhibited remarkable emergent capabilities (Chowdhery et al., 2022). Beyond generating fluent and coherent text, GPT-2 showed the ability to perform tasks such as summarization, translation, and question answering—despite not being explicitly trained for these tasks.

1.2.2 LLMs are passive learners

Since the release of GPT-2, a wide range of increasingly large and powerful LLMs have been developed (e.g., (Brown et al., 2020; Scao et al., 2022; Touvron et al., 2023; Chowdhery et al., 2022; Jiang et al., 2023a)). These models have shown continuous and substantial improvements across a broad spectrum of tasks. Notably, they exhibit capabilities such as *in-context learning*—the ability to adapt to new tasks given only a few examples—alongside robust performance in open-domain question answering, mathematical problem solving, and even forms of reasoning (Lampinen et al., 2024).

The impressive performance of LLMs challenges existing theories of how linguistic meaning is acquired (Bender & Koller, 2020; Bisk et al., 2020; Merrill et al., 2021; Piantadosi & Hill, 2022; Mollo & Millière, 2023; Pavlick, 2023; Gubelmann, 2024). Many of the tasks these models successfully perform are traditionally assumed to require an understanding of the meaning of words. However, the training paradigm of LLMs makes it evident that they lack any form of grounding. These models are trained solely to predict the next word in a sequence, having passively processed vast corpora of text without any direct interaction with the physical or social world. In this sense, their operation closely parallels the scenario described in Searle's Chinese Room thought experiment: they manipulate symbols according to statistically derived rules without any access to the underlying meaning. As Searle argued, such symbol manipulation, devoid of experiential grounding, is insufficient for genuine understanding.

The earlier discussions of Turing's test and ELIZA prompt a fundamental question: do LLMs truly understand the meaning of the words they use? While it is difficult to claim that LLMs possess no semantic understanding at all, it is equally unclear where their limitations lie. Bender & Koller (2020) caution that many of the seemingly impressive abilities of LLMs may be misleading, especially considering the human tendency to anthropomorphize and attribute meaning or intelligence to machines (Block, 1981). Several studies have highlighted critical shortcomings in LLMs. For instance, McCoy et al. (2019) demonstrated that LLMs often rely on shallow heuristics to succeed on standard benchmarks. More recently, Shojaee et al. (2025) showed that LLMs' reasoning abilities break down entirely beyond a certain level of task complexity—even when explicitly provided with step-by-step plans or algorithms. Broadly speaking, Mahowald et al. (2024) argue for a distinction between formal competence (i.e., mastery of linguistic patterns and regularities) and *functional competence* (i.e., the capacity to use language meaningfully in real-world contexts to pursue goals (Roy, 2005b)), a distinction also observed in human cognition. While LLMs excel at formal competence, their functional abilities remain significantly limited.

1.3 Towards grounded LLMs through interactions

Building computational models of language understanding that are grounded in experience is not a new endeavor and has been extensively explored across several research fields. However, much of this literature has remained largely overlooked within the NLP community. In particular, many of these models provide compelling evidence for the embodied framework of language acquisition, emphasizing how language is acquired and

grounded in one's sensorimotor experiences.

1.3.1 Existing models of grounded language understanding

Among the fields investigating computational models of grounded language understanding and acquisition, developmental robotics has arguably been the most active. By leveraging physical robots, this field aims to design machines that learn and develop in ways analogous to human—particularly child—development (Lungarella et al., 2003). Robots offer a uniquely well-suited framework for studying the role of embodiment, sensorimotor experience, and social interaction in cognitive development.

Over the past two decades, a wide range of models of grounded language acquisition have been proposed (see reviews by Cangelosi et al. (2010); Cangelosi & Schlesinger (2015); Cangelosi & Stramandinoli (2018); Oudeyer et al. (2019)). These models provide empirical support for key aspects identified by embodied and usage-based theories of language learning. For example, they highlight the importance of constructing an internal world model (Roy, 2005a; Dominey et al., 2009) and of incorporating social learning mechanisms (Steels, 2001; Steels & Kaplan, 2002; Dominey et al., 2009). The models address different forms of grounding, from mapping words to sensory modalities (e.g., naming perceived objects or performed actions) to grounding abstract words via associations with concrete ones. To achieve this, developmental roboticists have employed a mix of symbolic and statistical AI techniques, drawing on supervised, unsupervised, and reinforcement learning.

The latter refers to learning mechanisms in which behavior is acquired through trial and error, guided by scalar feedback signals known as rewards (Sutton & Barto, 2018). Reinforcement learning (RL) has been widely used to develop artificial agents capable of learning from interaction with their environment. The goal in RL is to discover a strategy—or policy—that maximizes the cumulative reward obtained over time. Early RL methods were limited to simple, discrete environments with small state and action spaces, where policies could be learned using tabular approaches. For example, methods such as Q-learning (Watkins & Dayan, 1992) represent the expected reward of each state-action pair in a lookup table. The introduction of Deep Q-learning (Mnih et al., 2015) and policy gradient methods (Sutton & Barto, 2018) extended the applicability of RL to more complex domains by using neural networks for function approximation. These advances have enabled significant progress in areas such as control systems (Degrave et al., 2022), games (Silver et al., 2016), robotics (Kalashnikov et al., 2018), and natural language generation (Ranzato et al., 2016).

The situated and embodied nature of RL has established it as a key learning mechanism for modeling embodied cognition, including grounded language acquisition. This holds true both within developmental robotics and in broader AI efforts to build autonomous artificial agents. A notable line of work has explored RL agents interacting with video game environments, where the agents must learn to achieve goals specified through natural language instructions. These studies reinforce the idea that embodied, situated learning facilitates the acquisition of new nouns (Hermann et al., 2017; Hill et al., 2021), and even supports language comprehension tasks such as answering natural language questions (Das et al., 2018). In parallel, other works have investigated language as a cognitive tool for labeling the skills performed by RL agents (Lair et al., 2019; Akakzia et al., 2021;

Colas et al., 2020). By associating language with learned behaviors, these approaches enable autonomous learning driven by the compositional nature of language—allowing agents to combine known skills into novel ones to expand their behavioral repertoire.

Such autonomous learning mechanisms are known to play a crucial role in human cognitive development, including language acquisition. Developmental robotics models have extensively incorporated curiosity-driven learning, where intrinsic motivational signals encourage learners to actively engage with their environment (Cangelosi et al., 2010; Cangelosi & Schlesinger, 2015; Oudeyer et al., 2019). Moreover, Oudeyer & Kaplan (2006); Moulin-Frier et al. (2014); Oudeyer & Smith (2016) provided evidence that curiosity alone can explain the emergence of vocal speech and language—initially through imitation of peers, and later through the strategic use of speech to attract attention and influence one's social environment (i.e., functional competence).

Curiosity-driven learning has also become a central paradigm in AI, particularly in the design of autonomous RL agents operating in complex environments. This includes methods that promote low-level exploration through intrinsic motivation signals (Schmidhuber, 1991b; Oudeyer & Kaplan, 2007; Baldassarre & Mirolli, 2013; Pathak et al., 2017; Burda et al., 2019b,a), as well as approaches that enable higher-level, structured exploration by developing *autotelic* agents—agents that set and pursue their own goals (Colas et al., 2022a,b; Sigaud et al., 2024). In RL, curiosity-driven learning empowers agents to autonomously explore and interact with their environment, thereby supporting the acquisition of new knowledge (e.g., discovering novel states or refining internal models of the world) and the development of new skills.

1.3.2 Building embodied LLM agents

Turning back to LLMs, efforts to improve their grounding have primarily focused on integrating multimodal inputs. In particular, there has been a growing interest in developing vision language models (VLMs), which are trained on datasets containing paired visual and textual inputs (e.g., images with associated captions, questions, or answers) (Yun et al., 2021; Alayrac et al., 2022; Radford et al., 2021; Gallouédec et al., 2024; Team et al., 2025). This strategy corresponds to a form of multimodal, or "direct" grounding (Cangelosi & Stramandinoli, 2018), where linguistic symbols are explicitly anchored to perceptual modalities—most commonly vision. Another avenue has involved leveraging human feedback, particularly through the fine-tuning of LLMs using reinforcement learning to optimize for human-aligned behavior (Ouyang et al., 2022; Gulcehre et al., 2023; Ahmadian et al., 2024). Known as reinforcement learning from human feedback (RLHF), this approach enables LLMs to explore and learn strategies for generating text sequences that maximize reward models derived from human preferences. In doing so, RLHF represents a rudimentary form of social learning.

While these constitute meaningful steps toward improving grounding in LLMs, the theoretical perspectives on human language acquisition reviewed earlier—as well as the existing computational models—indicate that grounding language should be approached within a situated and embodied framework. Several position papers in NLP have similarly called for a stronger focus on embodied language models (Kiela et al., 2016; Lucy & Gauthier, 2017; McClelland et al., 2019; Bender & Koller, 2020; Bisk et al., 2020; Tamari

et al., 2020). The principal contribution of this manuscript is to propose and empirically investigate several approaches for embodying LLMs and enabling them to be incrementally updated—i.e., grounded—through interaction with external environments. In particular, we focus on grounding their functional competence, that is, their ability to use language effectively to achieve goals.

This work is not the first to explore the embodiment of LLMs—that is, enabling them to perform actions and interact with external environments. Early contributions such as Yao et al. (2020); Ahn et al. (2022); Huang et al. (2022); Liang et al. (2023); Wang et al. (2022); Yao et al. (2022) showed that the planning capabilities of LLMs could be leveraged for action selection in robotic and interactive settings. However, a fundamental limitation of these approaches lies in the potential misalignment between the LLM's internal representations and the external environment it operates in. This misalignment constrains the model's ability to use language effectively in context—i.e., its functional competence. We argue that these pioneering works overlook a key component: the integration of a feedback loop that allows the model's internal representations to be continuously updated and aligned with the outcomes of its interactions.

The central question that follows is: how can such an update mechanism be implemented? We propose to build on existing evidence from computational models of grounded language acquisition, which highlight how reinforcement learning and curiosity-driven learning can be leveraged to shape autonomous embodied agents that actively explore their environment and incrementally update their knowledge. We argue that our contributions are particularly timely given the growing body of work investigating how LLMs can be trained with reinforcement learning to enhance their reasoning abilities, especially in settings where models interact with programs or external tools to solve problems (Wen et al., 2024b; Zhou et al., 2024b; DeepSeek-AI et al., 2025; Wang & Ammanabrolu, 2025).

1.3.3 Contributions

This present research aims to bridge the gap between embodied theories of language learning—emphasizing action-perception interactions—and current LLMs. Specifically, we investigate how LLMs can be embodied in external environments, enabling them to explore, acquire new knowledge, and ground their internal representations by using language to solve tasks through curiosity-driven RL. The core contribution of this work is, therefore, the proposal of an interaction-based approach to grounding LLMs' functional competence. We start Chapter 3 by introducing what we define as **functional grounding**: the grounding of symbol processing used to control, model, and predict external environmental dynamics in order to pursue goals. One important clarification is that functional grounding does not inherently imply multimodality. Functional competence in symbol processing can emerge through interaction with an external environment using only symbolic inputs and outputs—provided that these interactions allow symbols to be grounded in the environment's external dynamics. For this reason, all empirical contributions in this research adopt a strict definition of embodiment: an agent is considered embodied if it can perceive and act upon an external environment, regardless of modality. We therefore exclusively use textual environments throughout this manuscript, allowing us to isolate functional grounding from other forms of grounding and focus specifically on the development of functional competence in LLMs.

We then present our first line of work (Chapter 3), which explores how LLMs' can be functionally grounded by learning a policy that enables an agent to perform actions and solve tasks within an environment. To our knowledge, we introduce the first approach that embodies an LLM as an agent's policy within an external environment and grounds its functional competence through RL via online interactions. Our method, termed *GLAM* (Grounded LAnguage Model), provides a foundation for enhancing LLMs' functional competence and has since been extended by several research teams (e.g. (Yan et al., 2023; Zhou et al., 2024b; Wen et al., 2024a,b)). Following a detailed presentation of GLAM, Chapter 3 offers an in-depth analysis of how functional grounding shapes the internal representations and knowledge of LLMs. We show that increasing the diversity of textual observations, especially when paired with contrastive learning, is crucial for developing robust and consistent functional competence—and even enables LLMs to internalize the dynamics of their environment.

While Chapter 3 examines the control aspect of functional competence, Chapter 4 turns to the prediction component. Specifically, we investigate how LLMs can serve as world models—that is, systems capable of predicting how actions alter the state of an environment. We argue that the ability to model environmental dynamics is a core element of functional competence that LLMs should develop. To this end, we introduce *WorldLLM*, a method inspired by developmental theories, particularly the way humans refine internal models through (1) curiosity-driven exploration and (2) the formulation of natural language intuitive theories. In WorldLLM, the agent is rewarded for discovering states that the LLM fails to predict accurately. It then leverages these experiences to generate natural language hypotheses about the environment's dynamics, which are incorporated into the LLM's prompt to improve future predictions. Through an iterative loop of exploration, hypothesis generation, and refinement, WorldLLM enables LLMs to progressively update their internal representations and enhance their capacity as predictive world models.

Finally, Chapter 5 further investigates the challenge of functionally grounding LLMs in complex environments containing a vast—and potentially infinite—number of tasks. We argue that achieving more general functional grounding of LLMs requires moving beyond constrained environments and embracing open-ended learning setups. In such contexts, prior work has explored autotelic RL agents that benefit from social guidance—where a peer or teacher provides linguistic descriptions of events—drawing inspiration from how caregivers support children's learning. These agents use mechanisms such as hindsight learning and off-policy RL to acquire diverse skills efficiently (Colas et al., 2022b). Chapter 5 begins with the introduction of *SAC-GLAM*, an extension of the GLAM approach that incorporates off-policy RL and hindsight relabeling. We show that SAC-GLAM, by combining sample-efficient learning with socially guided feedback, significantly accelerates the functional grounding process for LLMs. We then turn to another crucial mechanism for autotelic agents: task prioritization, also known as automatic curriculum learning (ACL). While Learning Progress-based approaches (Matiisen et al., 2017; Portelas et al., 2020a; Kanitscheider et al., 2021) have proven effective, they do not scale well to large task spaces. To overcome this, we introduce *MAGELLAN*, a method that extends the LLM-based agent with metacognitive monitoring capabilities—that is, the ability to estimate its own competence. By leveraging the LLM's understanding of semantic relationships between language-specified tasks, MAGELLAN learns to generalize competence estimates

and Learning Progress across expansive task spaces. We show that MAGELLAN can: (1) efficiently prioritize tasks for the agent, and (2) shape—or ground—the LLM's internal representations to align with the environment's dynamics. The chapter concludes by addressing scenarios in which tasks exceed the LLM's capabilities but assistance (e.g., calling another LLM) is available. Using MAGELLAN's framework, we show that metacognitive abilities enable LLMs to autonomously learn help-seeking strategies through online interaction alone. Additionally, we demonstrate that these strategies can adapt to a user-defined budget, balancing the maximization of task performance with the minimization of the cost of external help, through a multi-objective bandit framework.

1.3.4 Collaborations

Every contribution presented in this manuscript is the result of collaborations with multiple researchers, without whom none of this would have been possible. My two advisors, Pierre-Yves Oudeyer (Inria) and Thomas Wolf (Hugging Face), participated in every single contribution I present here. Most of the research in this manuscript is also the result of a very fruitful collaboration with Thomas Carta (Inria), a fellow PhD student. Thomas and I started our PhD adventure at roughly the same time, and from our common research interests grew an efficient tandem. I was also very fortunate to have many interactions with Thomas's two non-Inria advisors: Olivier Sigaud (ISIR) and Sylvain Lamprier (University of Angers). Together with Pierre-Yves, Thomas, Olivier, and Sylvain, we co-authored GLAM and the follow-up grounding analysis (Chapter 3), as well as SAC-GLAM and MAGELLAN (Chapter 5). Thomas Carta also co-supervised two Master's internships with me: Loris Gaven (now a PhD student in our team at Inria), the first author of SAC-GLAM and MAGELLAN, and Guillaume Levy, the first author of WorldLLM (Chapter 4). Cédric Colas, a former PhD student at Inria and now a postdoctoral researcher at MIT, notably co-authored WorldLLM and MAGELLAN, in addition to providing insightful feedback on most of the other projects. Nicolas Yax, another PhD student (jointly at ENS and in our team) studying metacognition in LLMs, was also actively involved and significantly impacted our work on this topic (Chapter 5). I was also fortunate to collaborate with Olivier's PhD student, Salim Aissi (ISIR), as well as Salim's other advisors, Laure Soulier (ISIR) and Nicolas Thome (ISIR), on the study of GLAM's functional grounding (Chapter 3). Salim is the first author and led this project. Finally, the two projects described in appendices H and G were also the result of collaborations. First, TeachMyAgent was the project carried out during my Master's internship at Inria. It was co-authored with Rémy Portelas (at that time a PhD student and now a Research Scientist at Ubisoft), Pierre-Yves, and Rémy's other PhD advisor, Katja Hofman (Microsoft Research). The other project, Jack of All Trades (JAT), was the result of a collaboration between Quentin Gallouedec (former INSA Lyon, now at Hugging Face), Emmanuel Dellandrea (INSA Lyon), who was Quentin's PhD advisor, Edward Beeching (Hugging Face), and myself. Quentin led the project along with Edward.

Most of the experimental studies presented in this manuscript were carried out using the HPC resources of IDRIS under the allocation A0171011996 made by GENCI.

1.3.5 Publications

Part of the material presented in this manuscript has been presented in the following articles:

Conferences

- Thomas Carta*, **Clément Romac***, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In Proceedings of the 40th International Conference on Machine Learning (ICML), Vol. 202, Article 150, 3676–3713. ** equal contribution*
- Loris Gaven, Thomas Carta, **Clément Romac**, Cédric Colas, Sylvain Lamprier, Olivier Sigaud, Pierre-Yves Oudeyer. 2025. MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces, Proceedings of the 42nd International Conference on Machine Learning (ICML). Vol. 267.
- Mohamed Salim Aissi, **Clément Romac**, Thomas Carta, Sylvain Lamprier, Pierre-Yves Oudeyer, Olivier Sigaud, Laure Soulier, and Nicolas Thome. 2025. Reinforcement Learning for Aligning Large Language Models Agents with Interactive Environments: Quantifying and Mitigating Prompt Overfitting. In Findings of the Association for Computational Linguistics: NAACL 2025, pages 7030–7046.
- Guillaume Levy, Cédric Colas, Pierre-Yves Oudeyer, Thomas Carta, **Clément Romac**. 2025. WorldLLM: Improving LLMs' world modeling using curiosity-driven theory-making. Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM).
- **Clément Romac***, Rémy* Portelas, Katja Hofmann, and Pierre-Yves Oudeyer. 2021. TeachMyAgent: A Benchmark for Automatic Curriculum Learning in Deep RL. Proceedings of the 38th International Conference on Machine Learning (ICML), 139:9052-9063. ** equal contribution*

Workshops / Pre-prints

- Loris Gaven, **Clément Romac**, Thomas Carta, Sylvain Lamprier, Olivier Sigaud, Pierre-Yves Oudeyer. 2024. SAC-GLAM: Improving Online RL for LLM agents with Soft Actor-Critic and Hindsight Relabeling. Intrinsically Motivated Open-ended Learning (IMOL) Workshop, Neural Information Processing Systems (NeurIPS).
- Quentin Gallouédec, Edward Beeching, **Clément Romac**, Emmanuel Dellandréa. 2024. Jack of all trades, master of some, a multi-purpose transformer agent. Aligning Reinforcement Learning Experimentalists and Theorists (ARLET) Workshop, International Conference on Machine Learning (ICML).

Chapter 2

Foundations: Language and agents

Contents

2.1 Computational models of language understanding 13
2.1.1 Symbolic approaches 14
2.1.2 Statistical approaches 18
2.2 Computational models of action learning 24
2.2.1 The agent-environment framework 25
2.2.2 Reinforcement learning 27
2.2.3 From curiosity-driven to autotelic RL agents 30
2.2.4 Language in RL agents 32
2.3 LLM agents 34
2.3.1 Embodied LLMs 34
2.3.2 RL applied to LLMs 36
2.3.3 Building autotelic LLM agents 38

We begin by laying the foundations of this manuscript's contributions: grounding the functional competence of LLMs through interactions with an environment. This chapter first provides a brief overview of the history of computational approaches to modeling language understanding, highlighting the shift from symbolic methods to statistical techniques, and how the grounding problem has shaped the field (Section 2.1). Then, to better explore how embodied LLMs could be functionally grounded, Section 2.2 presents an overview of strategies for learning situated and autonomous agents that solve goals, with a focus on deep reinforcement learning and intrinsic motivation signals. Finally, Section 2.3 discusses attempts at building LLM agents, as well as applying reinforcement learning on LLMs.

2.1 Computational models of language understanding

The first prototypes of machines capable of understanding language emerged at the very beginning of computer science. For instance, I discussed in Chapter 1 the program ELIZA (Weizenbaum, 1966), a software that attempted to simulate a psychotherapist using simple rules to transform what the patient said into questions that fostered the

conversation. Another well-known attempt is SHRDLU by Winograd (1971), a program that allowed users to interact with a virtual environment made of "blocks" through language queries. From that point until today, the field of NLP (i.e., building machines that process language) has been highly active and has undergone significant paradigm shifts. The sections below provide a tentative and brief overview of this evolution. In particular, the evolution of NLP cannot be separated from developments in linguistics, psychology, and cognitive science regarding how humans learn and use language. For this reason, the following sections will alternate between theoretical perspectives and computational models.

2.1.1 Symbolic approaches

ELIZA and SHRDLU are both symbolic approaches: they rely on a set of pre-defined symbols (e.g., words) and a set of pre-defined rules for processing these symbols. Symbolic approaches were at the core of Artificial Intelligence for decades (Russell & Norvig, 2009). In the context of language, the linguist Noam Chomsky laid the foundations of modern linguistics with theories based on generative grammars—i.e., innate grammatical rules that humans possess and that underlie all languages (Chomsky, 1957). These theories relied on what is now considered a refuted claim: that children are not exposed to enough linguistic input to acquire the richness of natural language, implying that innate grammatical structures must exist. These ideas inspired the vast majority of early computational models, which for decades relied on purely syntactic rules to process language queries (Jones, 1994).

Meaning without grounding

However, several researchers have discussed the limitations of such approaches. In particular, Searle (1980), and a few years later Harnad (1990), proposed thought experiments highlighting that the meaning of words cannot be captured by systems solely based on grammatical rules. The latter notably showed that symbols must be grounded in something other than other symbols (i.e., rules for relating a symbol to others are not sufficient) and raised the question of how such a grounding process occurs—known as the *Symbol Grounding Problem*. These arguments were later supported by empirical results in neuroscience, which showed that sensorimotor brain areas are involved when humans process language (e.g., (Pulvermüller et al., 2001; Glenberg & Kaschak, 2002; Scorolli & Borghi, 2007; Chersi et al., 2010)). Soon after, the embodied cognition movement in cognitive science also challenged the classical model of "amodal" (i.e., not related to any perceptual inputs) symbol processing, in favor of a view of cognition as grounded in one's interactions with the environment (Barsalou, 1999; Pecher & Zwaan, 2005).

The functional role of language

This view that language is grounded in one's experience is also predominant in psychology among constructivists and developmentalists, such as Piaget (Piaget, 1954) and Vygotsky (Vygotsky, 1962). The former, for instance, discussed schemas in which

words' meaning is associated with sensorimotor experiences. The latter notably argued that language develops initially as a means to communicate with the child's caregiver, before being progressively internalized as a tool for structuring thought. Language is a tool that emerges from and supports the development of the child's cognitive capacities, such as categorization, planning, and imagination (Mirolli & Parisi, 2011; Colas et al., 2022a). All these findings gradually spread to linguistics, which moved away from innate grammar theories toward a usage-based approach in which:

> *"Language structure emerges from language use."* – Tomasello (2003)

If language is tightly bound to sensorimotor experience, the precise nature of the relationships between experience and word meaning remains largely unresolved (Mollo & Millière, 2023). The most direct—and arguably most extensively studied—form of such grounding is referential grounding, i.e., the mapping between words and their referents: objects, properties, or events in the world that linguistic expressions denote (Cangelosi & Stramandinoli, 2018; Mollo & Millière, 2023). However, referential grounding represents only one facet of the broader grounding problem. As discussed by Cangelosi & Stramandinoli (2018) and Mollo & Millière (2023), multiple forms of grounding have been proposed, and identifying which of these are necessary or sufficient for artificial systems to meaningfully process natural language remains an open question.

Beyond the association between words and experiences, Roy (2005b) argued that language also serves a functional role: it is used to act upon and control the environment in order to achieve goals, a notion he refers to as *functional meaning* (see Figure 2.1). Building on this perspective, Mahowald et al. (2024) recently introduced the concept of *functional competence*, defined as the ability to use language effectively for goal-directed behavior. Functional meaning also connects to classical philosophical approaches to language, such as Conceptual Role Semantics (CRS), which grounds the meaning of symbols in their functional role—namely, how they are used and manipulated in relation to other symbols within a system (Harman, 1982).

Throughout this manuscript, we focus on grounding mechanisms that aim to endow artificial agents with such functional competence.

Computational models of grounded language understanding

Compared to early rule-based theories of language understanding, building computational models of grounded language remains a significant challenge. Such models must operate within situated and embodied contexts while also accounting for the cognitive processes that are deeply intertwined with language use and development.

Drawing from both developmental psychology and robotics, the field of developmental robotics seeks to build computational models of human learning and development using robotic agents. A key strength of this approach lies in its use of physically embodied systems, which aligns with the strong emphasis in developmental and cognitive sciences on the importance of situated, sensorimotor experience. As a result, developmental robotics has made substantial contributions to the study of grounded language acquisition over the past two decades (see reviews by Cangelosi et al. (2010); Cangelosi & Schlesinger (2015);

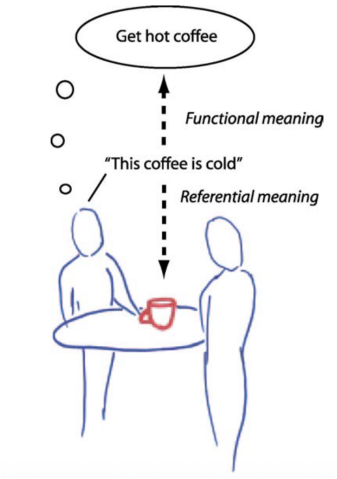


Figure 2.1: Roy (2005b) identifies a duality in language meaning. In referential meaning, words are used to talk about a physical situation. In functional meaning, words are used to achieve a goal.

Cangelosi & Stramandinoli (2018); Oudeyer et al. (2019)). These models are implemented in robots capable of interacting with the physical world and social partners—whether humans or other robots—and have been used to investigate a range of developmental stages. Notably, they have provided support for theoretical hypotheses regarding the cognitive mechanisms underlying language understanding.

For example, Roy (2005b,a) proposed a computational model based on Piagetian schemas, in which a robot maintains and updates an internal world model to map natural language instructions onto sequences of actions. In parallel, several models emphasized the role of social interaction in language acquisition, particularly in learning to associate words with objects (Steels, 2001; Steels & Kaplan, 2002; Dominey et al., 2009). For instance, Dominey et al. (2009) demonstrated that social learning mechanisms can significantly accelerate the acquisition of new vocabulary. In another line of work, Steels (2001) introduced language games—interactive scenarios in which two embodied agents negotiate and converge on shared labels for referents in their environment. Additionally, several models underscore that understanding language acquisition requires consideration not only of individual developmental timescales but also of the broader cultural evolution of language (Kirby et al., 2014; Oudeyer et al., 2019).

Overall, Cangelosi et al. (2010) summarizes how language development is interconnected with other cognitive skills in the broader process of cognitive development (see Figure 2.2). Existing models have primarily focused on the interaction between language and these related skills, such as motor control and social learning. A key aspect assumed in many of these models is the presence of intrinsic motivation in the learner, which drives active engagement in the learning process (Cangelosi & Schlesinger, 2015). While some models take this motivation as a given, others—such as the model proposed by Oudeyer & Kaplan (2006)—explicitly investigate how intrinsic motivation supports language acquisition, for example by guiding exploration and enabling the emergence of communication abilities. Importantly, Oudeyer & Kaplan (2006); Moulin-Frier et al. (2014); Oudeyer & Smith (2016) showed that curiosity-driven learning alone can explain the emergence of vocal speech and language. This emergence first starts through babbling and the

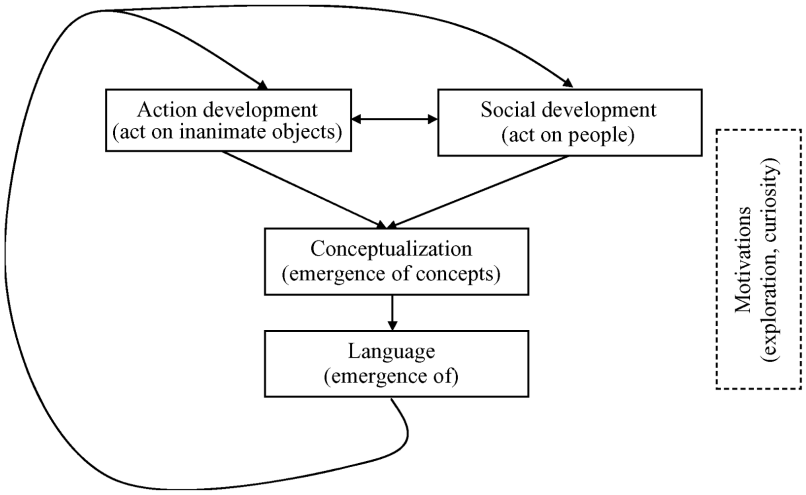


Figure 2.2: The connections between actions, social learning, concepts, and language illustrated by Cangelosi et al. (2010). Actions and social learning drive the emergence of concepts, which themselves drive the emergence of language further used in actions and social learning. Importantly, intrinsic motivation supports the development of these cognitive skills.

imitation of their caregivers. But progressively, children discover that vocal speech (and eventually language) can be used to affect and influence their caregivers, **discovering the functional meaning of language**.

In terms of grounding mechanisms, computational models have explored multiple forms of grounding. The most immediate and extensively studied form in developmental robotics is known as direct (or referential) grounding (Cangelosi & Stramandinoli, 2018; Mollo & Millière, 2023). This form aligns closely with the concept of referential meaning described by Roy (2005b), wherein the robot learns to associate words with experiences—such as naming observed objects or describing performed actions. Direct grounding thus involves mapping linguistic symbols to representations in other modalities. Numerous computational models have been proposed to support this form of grounding, as reviewed by Roy (2005a) and Cangelosi et al. (2010). A further challenge in grounded language modeling involves the representation and learning of abstract concepts, for which the contribution of sensorimotor experience is less well understood (Cangelosi & Stramandinoli, 2018; Mollo & Millière, 2023). In this context, grounding transfer (also called relational grounding by Mollo & Millière (2023))—i.e., grounding abstract symbols by linking them to already grounded concrete symbols—has been identified as a promising direction (Cangelosi & Stramandinoli, 2018). To support such symbolic connections, Thill et al. (2014) emphasized the potential of statistical methods from distributional semantics (e.g., (Turney & Pantel, 2010; Boleda, 2020)) to capture relationships between abstract and concrete concepts, enabling symbolic systems to inherit meaning from previously grounded terms. This emphasis on non-symbolic approaches for grounding transfer is not an isolated trend in developmental robotics. As noted by Oudeyer et al. (2019), many computational models integrate machine learning techniques, and several even combine symbolic and statistical approaches. Moreover, computational models of language acquisition often integrate and reveal the interplay between multiple learning paradigms—including supervised, unsupervised, and reinforcement learning—each

contributing to different aspects of the acquisition process.

Collectively, a wide array of models has been proposed by developmental roboticists to study grounded language acquisition and understanding. These models are grounded in theories that place sensorimotor experience (and curiosity-driven learning) at the core of cognitive and linguistic development—contrasting sharply with early syntactic rule-based theories of language understanding, which have been criticized for their lack of grounding and inability to capture meaning. Grounded computational models have provided useful empirical support for grounded theories of language. However, despite their strengths, these models face notable limitations. Most rely on carefully engineered, domain-specific architectures tailored to the particular cognitive processes under investigation, making it difficult to scale them toward the acquisition of the rich, complex structures found in natural human language. Moreover, many grounded models underutilize recent advances in statistical machine learning, particularly deep learning, which limits their ability to capture the complex statistical regularities that characterize real-world language use.

2.1.2 Statistical approaches

Distributional semantics is the research area that studies how the meaning of words can be captured from statistical patterns in the contexts in which they typically appear. Pioneering works by Firth (1957) and Harris (1954) laid the foundations for the revolution that occurred within the NLP field in the early 2010s.

> *"You shall know a word by the company it keeps."* – Firth (1957)

While early models of distributional semantics had been proposed before this revolution (e.g., (Dumais, 2004; Blei et al., 2003)), the amount of data accessible on the internet, the increase in computing power, and the development of ML approaches contributed to making distributional semantics approaches ubiquitous. In particular, a large number of works studied how neural network-based approaches could be used to produce vector representations of words based on large corpora (e.g., (Bengio et al., 2003; Turney & Pantel, 2010; Turian et al., 2010; Mikolov et al., 2013; Goldberg, 2016)). An example, illustrated in Figure 2.3, is the Word2Vec method from Mikolov et al. (2013), which learns vector representations of words by learning to predict which word best fits a context of other words (Continuous Bag Of Words, CBOW, approach) or which context a word would best fit in (skip-gram approach). These representations can then be used to solve specific NLP tasks such as entity recognition, summarization, language generation, or translation.

In fact, such models do not use words but what are called tokens—a set of integers that are associated with words, subwords, punctuation, etc. Several methods exist for transforming a text (a sequence of characters) into a sequence of tokens. They mainly differ in how they break down words into subwords, usually leveraging pre-defined mappings or learning the mapping from a corpus (e.g., (Gage, 1994; Schuster & Nakajima, 2012; Sennrich et al., 2016; Kudo, 2018; Song et al., 2021)).

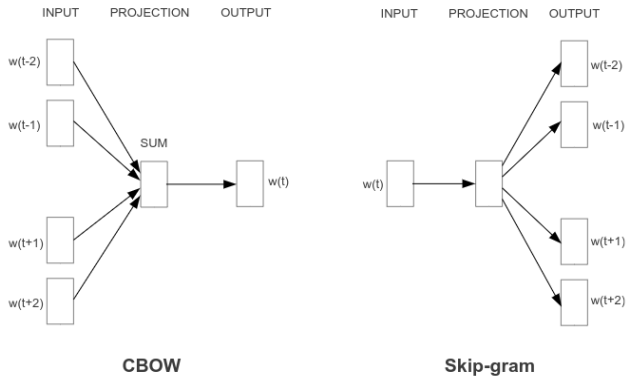


Figure 2.3: The two possible architectures of Word2Vec (Mikolov et al., 2013). A neural network learns vector representations of words by either (1) learning to predict which word best fits a context of other words (Continuous Bag Of Words, CBOW, approach) or (2) which context a word would best fit in (skip-gram approach).

From recurrent neural networks to Transformers

One task that has been of particular interest to the NLP community since its earliest days is machine translation (see, for example, (Hutchins, 2004)). Within the NLP revolution, neural network-based approaches also transformed translation and became known collectively as neural machine translation (NMT) (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). Because translation is considered a sequence-to-sequence task in NLP (i.e., given an input sequence of tokens in one language, the goal is to generate an output sequence of tokens in another language), NMT models typically adopt an encoder-decoder architecture composed of two neural networks—often recurrent models such as LSTMs (Hochreiter & Schmidhuber, 1997): (1) an encoder network, which reads the input sequence and produces a single vector representation of it, and (2) a decoder network, which generates the output sequence (see Figure 2.4). Each decoder block produces a probability distribution over the next possible token. The entire architecture is trained to maximize the likelihood of the expected output sequence (i.e., maximizing the likelihood of the next token given the previous ones), a training objective known as *causal language modeling* (CLM).

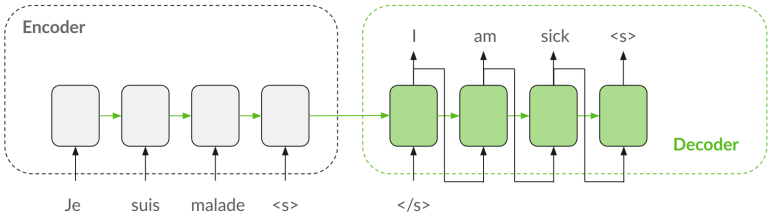


Figure 2.4: An example of an encoder-decoder architecture for NMT from French to English. Green arrows represent the flow of vector representations between words (i.e., the hidden state when using classic recurrent neural networks or the hidden state and memory when using LSTMs). Beginning of sequence ("</s>") and end of sequence ("<s>") tokens are used to start and stop decoding the output sequence.

However, these architectures suffer from a major bottleneck: a single vector representation encodes the entire input sequence. To overcome this limitation, Bahdanau et al. (2014) proposed the additive *attention* mechanism, which allows the decoder to access the vector representations produced by the encoder for each individual token of the input sequence. Specifically, additive attention learns a function (i.e., a neural network) that computes a similarity score between the current internal state of the decoder (called the *query*) and each representation produced by the encoder (called the *keys*). Based on these similarity scores, attention weights are computed (via a softmax function), and the decoder receives a weighted sum of all input representations (called the *values*). By enabling the decoder to "attend" to different parts of the input sequence as it generates the output, this mechanism led to state-of-the-art results in machine translation.

Two years later, Vaswani et al. (2017) introduced a new encoder-decoder architecture for NMT that eliminated recurrent neural networks (RNNs) entirely, relying instead on attention mechanisms and parallel computations across multiple representations (called *heads*) for each token. When processing a token, each head in a specific layer performs self-attention (i.e., a learned projection is used to compute the query for the current token, while two other learned projections produce the keys and values for all tokens), followed by feed-forward neural networks and residual connections (He et al., 2016). The outputs of all heads are then aggregated through a learned linear projection and passed to the next layer. This architecture, named the *Transformer*, not only achieved impressive results but also improved training stability and introduced highly parallelizable operations, making it feasible to train much larger models (i.e., with significantly more parameters). Like earlier architectures, transformers use an encoder-decoder setup. In particular, the encoder is bidirectional, meaning that each token attends to all other tokens in the input sequence—including those that follow. While bidirectional encoders existed prior to transformers (e.g., (Schuster & Paliwal, 1997)), they were significantly more difficult to train. The transformer encoder produces one vector representation per input token, forming a matrix. The decoder then autoregressively generates the output sequence—one token at a time—by computing the probability distribution over the next token using self-attention over previous output tokens and cross-attention over the encoder's representations.

Representational properties of transformers

Following the introduction of this new architecture, a line of research focused on the representational capacities of transformers. First, Radford et al. (2018) introduced Generative Pre-training (GPT), a model that uses only the decoder component of the transformer and is trained with CLM on large, general-purpose corpora. This model can then be adapted to specific NLP tasks (e.g., entity recognition, question answering) by fine-tuning it on data related to the target task. The idea of pre-training a model to learn general-purpose representations and then fine-tuning it for specific tasks was largely inspired by similar trends in computer vision (e.g., (Girshick et al., 2013; Donahue et al., 2014)). Building on this idea, Devlin et al. (2019) proposed BERT, a similar approach based on encoder-only transformers. The authors demonstrated that the bidirectional nature of the encoder leads to improved representations. Since each token in the encoder has access not only to preceding but also to following tokens, CLM is no longer suitable.

Instead, Devlin et al. (2019) used a masked language modeling (MLM) objective: randomly selected tokens in the input sequence are replaced with a special mask token, and the model is trained to predict the original tokens. BERT's representational abilities were extensively studied in a line of work known as "BERTology" (e.g., see the survey by Rogers et al. (2020)), which revealed its strengths in capturing syntactic knowledge (e.g., hierarchical structure (Lin et al., 2019)) and some aspects of semantic understanding (e.g., relationships between entities (Tenney et al., 2019)), while showing limitations in encoding world knowledge (e.g., (Forbes et al., 2019)).

Language models

Parallel to this, Radford et al. (2019) proposed GPT-2. This new model resembles GPT in all its technical aspects, but its underlying philosophy changes drastically: instead of producing a pre-trained backbone that must be fine-tuned for each specific task, the authors proposed that every NLP task could be approached through natural language generation. Their intuition was that a model trained solely to generate text (i.e., using CLM) could solve any NLP task by receiving a textual description of the task as input and generating the corresponding answer. To test this, Radford et al. (2019) trained a large transformer model (with up to 1.5 billion parameters, compared to BERT's 340 million) using a vast and curated dataset of web pages crawled from the internet. This language model (i.e., a model predicting the next token and trained with CLM) achieved state-of-the-art performance on most evaluation benchmarks in a zero-shot setting—i.e., without being specifically designed or fine-tuned for those tasks. In their conclusion, the authors stated:

> *" When a large language model is trained on a sufficiently large and diverse dataset it is able to perform well across many domains and datasets. "* – Radford et al. (2018)

Large language models, or LLMs, were born. From that point onward, a wide array of LLMs using decoder-only transformers and trained following the same paradigm as Radford et al. (2019) were developed. This includes, for instance, T5 (Raffel et al., 2020), GPT-3 (Brown et al., 2020), BLOOM (Scao et al., 2022), Gopher (Rae et al., 2022), PaLM (Chowdhery et al., 2022), and Llama (Touvron et al., 2023). Since GPT-2, few changes have been made to the overall architecture or training objectives. Among the most notable developments is that LLMs are now routinely fine-tuned to follow instructions and generate text aligned with human preferences (e.g., (Ouyang et al., 2022; Rafailov et al., 2023)). From few-shot learning—also known as in-context learning, the ability to adapt to new tasks given only a few examples—to reasoning, LLMs now demonstrate unprecedented capabilities across a broad range of tasks, extending beyond NLP (e.g., playing video games (Wang et al., 2023a) or contributing to scientific research (Lu et al., 2024)). These capabilities have led to the widespread deployment of LLMs in applications used by an ever-growing number of users (e.g., ChatGPT).

Limits of LLMs

The NLP community has come a long way—from early rule-based approaches such as ELIZA (Weizenbaum, 1966) to modern LLMs. While, at first glance, the longstanding goal of building machines that understand natural language appears nearly achieved with LLMs, the question of whether meaning can exist without grounding remains open (Bender & Koller, 2020; Bisk et al., 2020; Merrill et al., 2021; Piantadosi & Hill, 2022; Mollo & Millière, 2023; Pavlick, 2023; Gubelmann, 2024). Even before the advent of LLMs, studies such as Lucy & Gauthier (2017) and Kiela et al. (2016) questioned and empirically demonstrated the limitations of distributional semantics in capturing meaning without grounding. For LLMs, various studies have investigated which aspects of meaning can be captured through CLM alone. For instance, Abdou et al. (2021) and Patel et al. (2020) showed that LLMs can encode conceptual structures resembling those found in perceptual spaces (e.g., color categories or spatial relationships like cardinal directions). Bender & Koller (2020) further reviewed evidence that some LLM abilities are driven by shallow syntactic heuristics. More recently, Shojaee et al. (2025) investigated the limitations of LLMs' reasoning abilities, showing that performance breaks down beyond a certain task complexity—even when hints are provided. This aligns with broader findings on LLMs' limited multi-step reasoning capacities (Valmeekam et al., 2022; Wu et al., 2024). In particular, Mahowald et al. (2024) demonstrated that the overall functional competence of LLMs—i.e., their ability to use language effectively in real-world, goal-directed contexts—remains underdeveloped.

These limitations, together with the previously discussed evidence from linguistics, developmental psychology, and cognitive science on the role of sensorimotor grounding in language, have motivated several position papers advocating for the integration of distributional semantics with embodied learning approaches (Kiela et al., 2016; McClelland et al., 2019; Bender & Koller, 2020; Bisk et al., 2020; Tamari et al., 2020).

Multimodal LLMs

Bisk et al. (2020) identified the use of perceptual data (e.g., images) in addition to text during the training of LLMs as a natural way to improve their grounding. Joint training on perception and text closely relates to direct grounding (Cangelosi & Stramandinoli, 2018), i.e., the mapping of symbols into other modalities. Prior to LLMs, several works explored the multimodal training of distributional semantic models, from classic neural networks (Howell et al., 2005; Mikolov et al., 2013) to transformers (Sun et al., 2019; Li et al., 2020; Chen et al., 2020b; Su et al., 2020; Zhang et al., 2021; Radford et al., 2021). Such joint training was shown to yield partial improvements in the learned representations (Yun et al., 2021). Regarding LLMs, the recent trend of VLMs investigates how to build models that process both text and visual input (e.g., images or videos). A common approach is to train or fine-tune a visual encoder to produce representations compatible with an LLM—either by generating token embeddings directly (see Figure 2.5) or by inserting additional attention layers into the LLM (Cho et al., 2021; Tsimpoukelli et al., 2021; Alayrac et al., 2022; Laurençon et al., 2023; Chen et al., 2023; Laurençon et al., 2024). These models are typically fine-tuned using datasets that pair images with text (e.g., captions or visual question-answering examples), or web documents in which images and

text are interleaved (Laurençon et al., 2023). However, beyond visual question answering, the broader capabilities of current VLMs—such as reasoning—have been shown to be limited (Wang et al., 2024a; Aissi et al., 2025).

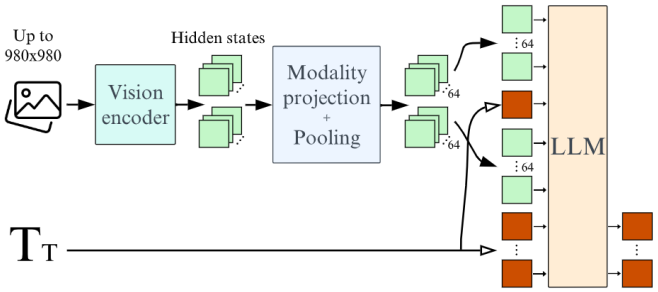


Figure 2.5: In Idefics2 (Laurençon et al., 2024), a visual encoder produces multiple token embeddings per image, which are then given to an LLM. Training usually involves fine-tuning both the visual encoder and the LLM.

In parallel, the integration of actions—in addition to perceptual inputs—into multimodal language models has also been explored. In particular, passive learning schemes such as behavioral cloning (i.e., imitating an expert's behavior by maximizing the likelihood of its actions) have been applied. Passively observing and imitating sequences of actions has been shown to be sufficient for encoding the meaning of certain words (Ebert et al., 2022). For example, Jiang et al. (2023b) and Brohan et al. (2023) trained multimodal transformers to control robots using datasets that combine task specifications with associated sequences of actions. These specifications include both perceptual inputs (e.g., images) and text (e.g., instructions). Reed et al. (2022) extended this idea with a model named Gato, which was trained to both imitate expert trajectories—i.e., sequences of actions in games and robots—and generate natural language. Their approach combines CLM, behavioral cloning, and CLM on multimodal inputs for visual question answering. In a similar vein, our work on the Jack of All Trades (JAT) model, described in Appendix G, proposes a unified approach. We collected a large-scale, open dataset consisting of 34 million expert trajectories from RL tasks across four domains, 250 million text documents, and 2.6 million text-image pairs. We then trained a single transformer model using the same objectives as Reed et al. (2022): CLM (on both text-only and text-image inputs) and behavioral cloning. When evaluated as a policy on the RL tasks, JAT achieves an overall 65.8% of the expert score (see Figure 2.6). However, we found that JAT's natural language generation capabilities remain limited. Moreover, we introduced an auxiliary objective on expert trajectories: predicting the next observation—i.e., how the environment changes as a result of an action. We showed that this objective improves performance on RL tasks. Often referred to as forward modeling, this mechanism closely relates to world modeling and will be further discussed in Section 2.2 and Chapter 4.

While JAT and Gato do not leverage any pre-trained LLM, Driess et al. (2023) proposed a VLM trained on a dataset that includes robotics tasks in which the actions are expressed not as low-level commands but as textual instructions for a low-level controller. Zitkovich et al. (2023) introduced RT-2, a similar model capable of directly producing low-level actions for a robot. RT-2 is trained using behavioral cloning to generate these actions. Other works, discussed in Section 2.3, have explored how LLMs or VLMs can

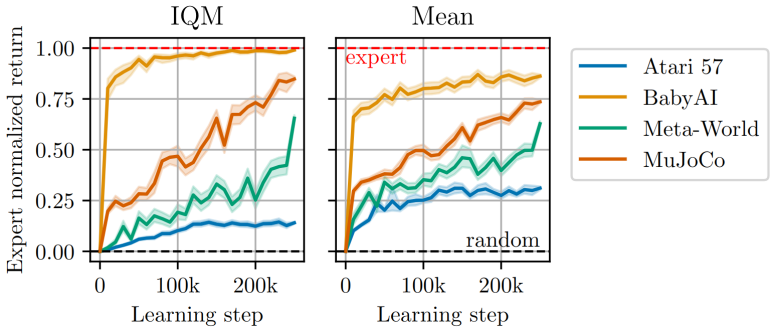


Figure 2.6: Aggregated expert normalized scores (interquartile mean, IQM, and mean) of JAT, with 95% Confidence Intervals for each RL domain as a function of learning steps.

be turned into agents that take actions to interact with an environment (e.g., in robotic settings).

In conclusion, this section has traced the evolution of the NLP's central challenges: building models of language understanding. Early attempts that relied solely on grammars and syntactic rules were shown to be insufficient for capturing meaning. In contrast, a substantial body of evidence from linguistics, psychology, and cognitive science emphasizes that language is grounded in one's sensorimotor and social experiences. This perspective led to the development of various grounded language models, particularly within developmental robotics. These models notably provided strong evidence for grounded language understanding. In parallel, statistical methods—especially distributional semantic models based on neural networks—have advanced significantly. Initially concerned with generating vector representations of words and sentences, the field has shifted toward training LLMs: generalist models capable of performing a wide range of NLP tasks through language generation, learned from massive internet-scale datasets. These models are purely statistical learners, yet they exhibit remarkable capabilities that suggest they capture key aspects of meaning. This is not entirely unexpected, as earlier distributional models already demonstrated an ability to capture conceptual relationships. These findings align with usage-based linguistic theories, which stress the importance of statistical learning in human language acquisition, and with insights from developmental robotics, which view statistical learning as a potential mechanism for grounding abstract concepts by relating them to concrete concepts grounded in experience. Nonetheless, many of these capabilities may be misleading, as humans tend to attribute intelligence too readily. Crucially, the lack of grounding in sensorimotor experience remains a significant limitation for the future development of LLMs.

2.2 Computational models of action learning

Building computational models of decision-making has a long history in artificial intelligence. In this section, however, I will focus on the area most relevant to the present research: *reinforcement learning (RL)*. RL encompasses a family of sequential decision-making problems in which an agent must perform a series of actions to maximize the cumulative *reward* received after each action. Before delving into RL itself, I will

first introduce key foundational concepts—such as the action-environment interaction framework and Markov Decision Processes—and briefly discuss alternative approaches that may be preferable in certain contexts.

2.2.1 The agent-environment framework

A fundamental concept in artificial intelligence is the notion of an *agent*. An agent is an entity equipped with sensors and actuators. It is situated in an *environment*, which it can observe through its sensors and modify using its actuators (Russell & Norvig, 2009). We formalize this by attributing to the environment an internal *state* $s \in S$, which encodes all relevant aspects of the environment. In the simplest case, we assume a setting with perfect information (also called fully observable), in which the agent can observe the entire current state of the environment through its sensors. We also define a space of possible *actions* A that the agent can perform via its actuators. Within this agent-environment framework, the agent alternates between observing the current state of the environment s_t, performing an action $a_t \in A$, and then observing the resulting state s_{t+1}. Central to RL, we also assume that, in addition to observing the new state s_{t+1}, the agent receives a scalar reward r_{t+1} for the action it took. This reward is typically tied to a specific *task* (also referred to as a *goal*) that the agent is expected to accomplish. These interactions between the agent and its environment are classically illustrated in Figure 2.7.

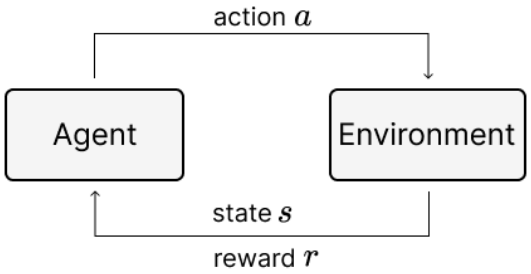


Figure 2.7: Agent-environment framework in which an agent interacts with an environment by observing its state s, performing an action a, observing the new state, and a reward r.

The sequence of interactions with the environment (where a single interaction is called a *step* and a sequence of steps a *trajectory*) can be either infinite or finite, depending on the problem. Finite trajectories may be bounded by a maximum number of steps or by termination conditions (e.g., task completion). When such sequences are bounded, the problem is referred to as episodic, and a single sequence from $t = 0$ to T is called an *episode*. This agent-environment framework has served as the foundation for the development of decision-making systems across various domains, including control (Bellman, 1957), economics (White, 1993), and games (Mnih et al., 2015).

Bandits

One particular problem that falls under the agent-environment framework is the case where only a single step has to be performed (i.e., episodes last for a single step). The agent must therefore select the action that yields the maximum immediate reward (i.e., the reward obtained after a single action). To address this, a substantial body of work has emerged under the umbrella of *bandit* methods. When dealing with a discrete action space A—and where the state has no influence on the optimal strategy—*Multi-Armed Bandit* approaches have been extensively studied (e.g., (Lai & Robbins, 1985; Auer et al., 2002a,b; Langford & Zhang, 2007)). These methods typically aim to minimize the cumulative regret, i.e., the difference between the reward obtained by the chosen action and the maximum reward that could have been achieved by selecting the optimal action:

$$\sum_{n=1}^{N}(\max_{a} r_{a,n} - r_n)$$

with N the total number of episodes. When the state influences the decision, methods known as contextual bandits have also been studied for cumulative regret minimization (e.g., (Li et al., 2010; Lu et al., 2010; Valko et al., 2013)).

One important aspect of this bandit scheme is that it constitutes an active learning process: the agent performs an action, receives a reward, and must adapt its behavior to progressively converge toward the optimal one. Moreover, minimizing cumulative regret implies seeking methods that require the fewest episodes to discover optimal behavior. I intentionally used the verb discover to highlight a central challenge common to all methods tackling agent-environment problems with active learning (e.g., bandits, but also RL): *exploration*. The agent must explore the space of possible strategies to identify the optimal one. However, it must also *exploit* the strategies it has already discovered to evaluate them and continue using one if it proves optimal. As a result, the agent faces a dilemma between exploration and exploitation.

Imitation Learning

In bandits, we assumed no access to any external help or a demonstrator showing how to behave. When an expert agent is available, one can learn to imitate it using demonstrations of trajectories (i.e., a dataset of transitions $< s_t, a_t, r_{t+1}, s_{t+1} >$). Methods studying how to best leverage such demonstrations are known as *imitation learning* methods. The most naive approach assumes that the expert follows the optimal strategy. In that case, the agent simply needs to clone the observed behavior—i.e., maximize the likelihood of selecting a_t from s_t (e.g., (Pomerleau, 1988; Argall et al., 2009; Gallouédec et al., 2024)). Rajaraman et al. (2020) and Kumar et al. (2022) notably discussed the limits and conditions under which imitation learning and behavioral cloning (BC) are effective. In particular, naive BC assumes access to optimal demonstrations. When suboptimal demonstrations are present, methods such as Reward Weighted Regression (Peters & Schaal, 2007) or Advantage Weighted Regression (Peng et al., 2021) apply weights to transitions or trajectories to favor higher-quality data, while others propose filtering to retain only high-quality transitions (e.g., (Chen et al., 2020a)). Additionally,

due to its passive learning scheme, imitation learning requires a sufficiently diverse dataset to enable the agent to behave appropriately across all possible encountered states. Finally, several works have studied the general limitations of imitation learning in capturing causal relationships (e.g., (Rajaraman et al., 2020; Lampinen et al., 2023; Gasse et al., 2023)).

Markov Decision Processes

When doing active learning in environments with interactions longer than one step (i.e., not bandits), the classical setting is to frame the environment as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ with:

- S: the space of possible states for the environment
- A: the space of possible actions in the environment
- $T : S \times A \times S \mapsto [0, 1]$: the transition function giving the probability of transitioning from a state s to a state s' by taking action a
- $R : S \times A \mapsto \mathbb{R}$: the reward function
- γ: a discount factor

The agent's strategy is formalized as a stochastic and Markovian *policy* function $\pi : S \times A \mapsto [0, 1]$, which gives the probability of selecting action a in state s. The policy's performance is measured by the cumulative discounted rewards obtained in a trajectory collected under π (also called the *return*):

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

This return can also be defined from a specific state s_t, which is called the *value*:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

Or even from a specific state-action pair s_t, a_t—i.e., the expected return from taking action a_t in state s_t:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

When the complete MDP is known to the agent, Dynamic Programming approaches such as Policy Iteration or Value Iteration can be used (Bellman, 1957). However, when the inner MDP is unknown to the agent, it must explore the environment and discover the optimal policy through trial-and-error: this is *reinforcement learning*.

2.2.2 Reinforcement learning

Reinforcement learning studies the problem of solving an unknown MDP (Sutton & Barto, 2018). Methods are usually grouped into two families: *model-free* and *model-based*. Model-free methods directly learn a policy by interacting with the environment.

Model-based methods use interactions to learn a model of the MDP, which can then be used for planning (i.e., selecting actions) or for learning the policy by interacting with the model instead of the environment. Learning a model of the MDP notably involves learning the transition function, usually called a *forward model*. In the present research, we mostly focus on model-free approaches, except in Chapter 4, where a forward model is learned—not for planning or policy learning, but to foster exploration of the environment by rewarding transitions that are poorly predicted (this will also be discussed in Section 2.2.3).

Value-based approaches

Among model-free RL approaches, *value-based* methods learn a policy by alternating between estimating the current policy's value (or Q-value) over encountered states and updating the policy according to this value estimation. Updating the value estimate is typically achieved using complete trajectories (this is called a Monte Carlo estimation) or using partial trajectories with Temporal Difference (TD) learning (Sutton & Barto, 2018). TD learning computes the estimation error (called the TD error) based on a partial trajectory by bootstrapping—i.e., using the current value estimate to approximate the ground truth. The most straightforward method for updating a Q-value function with TD uses single transitions (see Sutton & Barto (2018) for details about TD learning) and applies the following update rule:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha(r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t))$$

Based on this Q-value estimation, updating the policy can simply be achieved by selecting the action with the maximal Q-value:

$$\pi(s_t) = \underset{a \in A}{\operatorname{argmax}}\, Q_\pi(s_t, a)$$

Note that this policy is deterministic and requires another mechanism for exploration (e.g., ϵ-greedy, discussed below).

This update approach is implemented in the SARSA algorithm (Sutton & Barto, 2018). Watkins & Dayan (1992) later introduced Q-learning, a related method in which the TD error is computed not using the Q-value of the action taken by the policy at time $t+1$, but instead by using the maximum Q-value among all possible actions under the current Q-value estimation:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a \in A} Q_\pi(s_{t+1}, a) - Q_\pi(s_t, a_t))$$

Such an approach is called *off-policy*, as it uses information (here, the Q-value of the next action) that does not come from the current policy. Off-policy approaches are generally known for being more sample-efficient and better suited to support exploration mechanisms.

As an example, Q-learning and SARSA are usually implemented using an ϵ-greedy action selection scheme for exploration: with probability $1 - \epsilon$, the action is selected by the policy (i.e., taking the action with the maximal Q-value), and with probability ϵ, the

action is selected randomly. In SARSA, this stochasticity is integrated into the Q-value estimation, while Q-learning, which always assumes optimal action selection, ignores it. Both SARSA and Q-learning require the storage of a Q-table: a table containing the current Q-value estimation of all state-action pairs.

Deep RL

Scaling to more complex environments, one cannot store or even know the set of all possible state-action pairs. This is particularly the case in environments where states are represented by continuous values. To overcome this, several works proposed approximating the Q-value using a neural network (e.g., (Riedmiller, 2005; Mnih et al., 2015)). The Deep Q-Network (DQN) proposed by Mnih et al. (2015) was a major breakthrough in RL that successfully applied Q-learning to a large set of Atari video games (Bellemare et al., 2013). This Q-network Q_θ is a neural network that takes a state as input and outputs the Q-value of each possible action. Mnih et al. (2015) addressed the known instability issues of such Q-value estimators by computing the loss over transitions sampled from a buffer D, and by using a second network $Q_{\theta'}$—updated less frequently—to stabilize the TD errors:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_\theta(s, a)\right)^2\right]$$

One major limitation of the DQN is that its policy plays by computing the Q-value of all the possible actions. When actions are continuous values—as is the case, for instance, in robotics setups—DQN can no longer be used. Instead, *policy-based* approaches seek to directly learn a parametric policy function π_θ using the *Policy Gradient Theorem* (Sutton & Barto, 2018). These methods learn a neural network-based policy through gradient ascent to maximize the policy's expected return. As with value-based approaches, the question is how to estimate the policy's expected return over the MDP. Williams (1992) proposed REINFORCE, where Monte Carlo estimations are used. However, such estimations are known to produce high-variance in the gradient estimates. Other methods, usually named *Actor-Critic*, learn an additional critic network: a neural network estimating the return or substitutes used to compute the gradient, such as the value, the Q-value, or the advantage ($A(s, a) = Q(s, a) - V(s)$). Methods such as A2C (Mnih et al., 2016), DDPG (Lillicrap et al., 2016), or SAC (Haarnoja et al., 2018) use TD learning to estimate a Q-value, while methods such as TRPO (Schulman et al., 2015) or PPO (Schulman et al., 2017) estimate the advantage and the value.

There are two important characteristics of policy gradient methods for the present research. First, most methods are on-policy. Among the off-policy ones are SAC, DDPG, and ACER (Wang et al., 2017). As in the value-based approaches previously discussed, DDPG uses a deterministic policy. For SAC and ACER, as well as most policy gradient approaches, a stochastic policy outputting a probability distribution over the possible actions is learned. Deterministic policies require external exploration mechanisms (e.g., ϵ-greedy), while stochastic policies' exploration—which samples actions from an estimated probability distribution—is generally controlled by maintaining a high entropy on the actions' probability distribution.

2.2.3 From curiosity-driven to autotelic RL agents

Exploring the environment is a key challenge for RL approaches. In Section 2.2.1, I discussed the exploration versus exploitation dilemma for learning an optimal policy through interactions with an environment. Moreover, many environments feature sparse reward settings, where most transitions result in a reward equal to zero. In such settings, the agent must efficiently explore the environment to discover rewarding transitions. Applying RL methods usually leads to poor results or requires an unreasonable number of interactions for classic exploration mechanisms (i.e., random action selection) to properly explore the environment.

Knowledge-based intrinsic motivations

Humans are known to explore their environment, and a large body of work has drawn attention to intrinsic motivation signals and curiosity-driven learning (Berlyne, 1954; White, 1959; Kidd & Hayden, 2015; Oudeyer & Smith, 2016; Gottlieb & Oudeyer, 2018). As a result, various computational models of intrinsic motivation have been proposed (e.g., see the surveys from Schmidhuber (1991b); Oudeyer & Kaplan (2007); Baldassarre & Mirolli (2013); Aubret et al. (2019)). Oudeyer & Kaplan (2007) notably grouped intrinsic motivations into two categories: knowledge-based and competence-based. Knowledge-based signals motivate the agent to collect new information and acquire new knowledge. These methods are usually integrated into RL by introducing an additional reward (called an intrinsic reward). This includes novelty (e.g., discovering unknown states) (Bellemare et al., 2016; Stanton & Clune, 2018; Burda et al., 2019a; Raileanu & Rocktäschel, 2020), empowerment (Klyubin et al., 2005; Still & Precup, 2012; Gregor et al., 2017), surprise (Achiam & Sastry, 2017; Berseth et al., 2021), uncertainty (Burda et al., 2019b; Sukhija et al., 2025), prediction error (Schmidhuber, 1991a; Pathak et al., 2017), or information gain over prediction error (Schmidhuber, 1991a; Oudeyer & Kaplan, 2007; Lopes et al., 2012; Houthooft et al., 2016). Such intrinsic rewards have been shown to be useful for learning a forward model of the MDP (Lopes et al., 2012; Schmidhuber, 1991a; Kim et al., 2020), as well as for solving environments with extremely sparse reward settings (Bellemare et al., 2016; Pathak et al., 2017).

Autotelic agents

The other category identified by Oudeyer & Kaplan (2007) is competence-based motivation signals. These intrinsic motivations no longer focus on the agent's ability to "know" the world but rather to control it. In particular, competence-based approaches focus on goals that an agent selects to learn. This *autotelic* principle (Steels, 2004) plays an important role in human cognition (White, 1959; Elliot & Fryer, 2008; Oudeyer & Kaplan, 2007). Integrating this form of motivation into RL learners was called *autotelic RL agents* (Colas et al., 2022b), i.e., agents that set their own goals and learn to solve them through RL (see Figure 2.8). Setting a goal requires two steps: 1) sampling a point in a multi-dimensional goal space G and 2) conditioning a goal-conditioned reward function $R : S \times A \times G \mapsto \mathbb{R}$ (Colas et al., 2022b; Sigaud et al., 2024). There are multiple challenges when designing an autotelic RL agent: 1) defining G, 2) defining

R, 3) defining a goal-selection scheme, and 4) learning a policy π. The last challenge is usually achieved using goal-conditioned RL, in which the policy (and value function if needed) is conditioned on the goal to achieve (Schaul et al., 2015). The use of off-policy approaches plays a crucial role in exploiting noisy and sparse reward signals induced by self-generated goals (Colas et al., 2020).

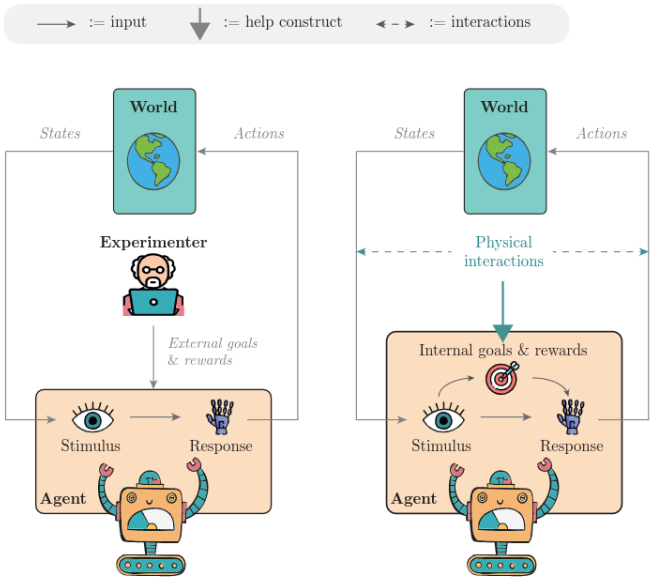


Figure 2.8: From goal-conditioned RL agents to autotelic RL agents (Colas et al., 2022a).

Defining the goal-selection scheme is a mandatory requirement, as the goal space might be too large for the agent to master all goals. Additionally, some goals might be impossible, or the possibly varying difficulty across goals may require the agent to prioritize which goal to learn next. The field of automatic curriculum learning (ACL) has notably proposed methods to address this goal-selection challenge:

> *"Automatic Curriculum Learning (ACL) [...] is a family of mechanisms that automatically adapt the distribution of training data by learning to adjust the selection of learning situations to the capabilities of Deep RL agents."* – Portelas et al. (2020b)

ACL methods have leveraged various forms of intrinsic motivations to select goals, such as intermediate difficulty (Florensa et al., 2018; Racaniere et al., 2020; Castanet et al., 2023; Rutherford et al., 2024), regret (Dennis et al., 2020; Jiang et al., 2021a; Parker-Holder et al., 2022), or Learning Progress (LP) (Stout & Barto, 2010; Matiisen et al., 2017; Portelas et al., 2020a; Colas et al., 2019) (see Portelas et al. (2020b); Narvekar et al. (2020) for surveys as well as Appendix H for a non-exhaustive empirical comparison of existing methods). I further discuss ACL methods, and in particular LP-based approaches, in Chapter 5, where exploration challenges for functional grounding of LLMs are discussed.

Finally, Colas et al. (2022b) discussed the existing approaches for defining G and R. For the latter, most current approaches assume access to a pre-defined goal-conditioned

reward function, but some works have attempted to learn this reward function (e.g., (Bahdanau et al., 2019a; Venkattaramanujam et al., 2020; Warde-Farley et al., 2019)). Recently, multiple works have started to study the use of pre-trained models such as VLMs to produce general reward functions (e.g., (Fan et al., 2022; Du et al., 2023; Zhang et al., 2024b; Klissarov et al., 2023; Colas et al., 2023; Zhou et al., 2024c; Sancaktar et al., 2025; Lee et al., 2025)). Concerning G, a large number of approaches also assume access to a pre-defined goal space. However, multiple methods learn goal embeddings, in particular when goals are expressed as images (Pere et al., 2018; Nair et al., 2018; Pong et al., 2020) or with language (Bahdanau et al., 2019b; Jiang et al., 2019; Hill et al., 2020a, 2021; Colas et al., 2023). This last series of work is of particular interest to this manuscript, as they highlight—following the argument of Colas et al. (2022a)—that language should play a major role in autotelic artificial agents, given the role it plays in autotelic mechanisms in humans (as discussed in Section 2.1).

2.2.4 Language in RL agents

The use of language within RL agents has been particularly active over the last decade, as discussed by Luketina et al. (2019). To study this, multiple testbed environments were proposed (e.g., (Côté et al., 2019; Chevalier-Boisvert et al., 2019; Hausknecht et al., 2020; Küttler et al., 2020; Shridhar et al., 2021; Zhong et al., 2021; Ammanabrolu & Riedl, 2021; Wang et al., 2022; Jansen, 2022)). In particular, Text Worlds (Côté et al., 2019) are environments in which an agent interacts through natural language: observations are descriptions of the environment's current state, goals are natural language instructions, and actions are textual commands. Notably inspired by early adventure video games such as Zork, Text Worlds are well-suited to study the use of commonsense knowledge by RL agents. Another type of environment combines multiple modalities such as visual or symbolic observations and textual instructions (e.g., (Chevalier-Boisvert et al., 2019; Bahdanau et al., 2019a; Hill et al., 2021)). These environments are particularly suited to study forms of grounding (e.g., direct), for instance, through embodied question answering (Das et al., 2018; Gordon et al., 2018; Liu et al., 2023b).

Grounding language through RL

In particular, several works studied language grounding in RL agents. They proposed using an embodied RL agent that must interact with an environment (through low-level actions) to solve instructions (i.e., goals specified with language). Hermann et al. (2017) and Hill et al. (2021) studied how to ground instructions into sequences of actions through RL in a visual environment. The latter notably examined the acquisition and direct grounding of new nouns before asking the RL agent to solve tasks involving these nouns (see Figure 2.9). Das et al. (2018) investigated similar navigation problems in visual environments but considered questions as instructions and used an answering module at the end of the trajectory to provide the response. Akakzia et al. (2021) also studied instruction-following RL agents (this time in a robotics setup), focusing on learning a mapping from natural language instructions to symbolic representations (i.e., a logical expression defining how objects must be positioned). They showed that decoupling language grounding and skill learning—a process observed in infants—leads to improved

performance. Lair et al. (2019) and Colas et al. (2020) went further by studying both the generation of instructions (using a restricted grammar and hand-designed rules) and the learning of an instruction-conditioned reward function. They notably demonstrated how simple language descriptions can be used to capture and generate complex behaviors. Finally, Lampinen et al. (2022) studied the use of explanations as an auxiliary objective for RL agents and how producing such explanations helps the agent capture causal structure.

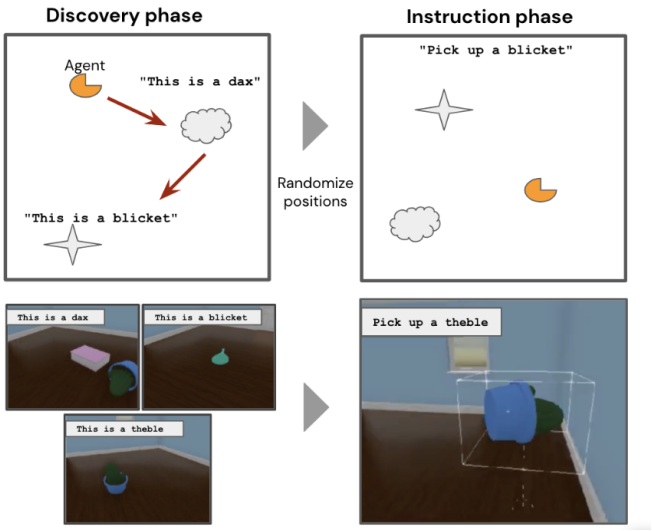


Figure 2.9: Hill et al. (2021) proposed two phases: one where the agent explores the environment and receives descriptions (Discovery phase) and another one where it is asked to solve tasks (Instruction phase). The authors show that their agent quickly learns nouns of new objects and is able to complete tasks that require interacting with the new objects.

To conclude, the field of RL has extensively studied and proposed methods enabling agents to solve sequential decision-making tasks through interactions with an environment. This learning process is grounded in trial-and-error, where the agent explores the environment and gradually refines its policy. Methods have been designed for both simple tabular and complex high-dimensional environments, as well as for deterministic and stochastic policies. Moreover, recent years have seen substantial progress toward more autonomous RL agents that do not rely on dense rewards but instead leverage intrinsic motivation signals. These signals range from acquiring new knowledge about the environment (e.g., improving a forward model) to developing new skills through autotelic mechanisms. The use of language in RL agents remains relatively understudied but has gained increasing attention. Language offers a natural way to define the goals that autonomous agents must pursue and is especially valuable for autotelic agents, who can leverage the compositional nature of language to continually generate new goals. A recent line of research has examined RL agents that follow natural language instructions, with a particular interest in how the learning process facilitates language grounding. They showed that learning to solve goals via RL is effective not only for grounding nouns to their referents (i.e., direct grounding) but also for capturing relational concepts (e.g., spatial relationships between objects). Luketina et al. (2019) conclude their review by emphasizing that language is a vector of knowledge and cultural artifacts, which could be pivotal in moving current RL agents beyond the classic tabula-rasa paradigm in which

everything must be learned from scratch. They notably identify large pre-trained NLP models (e.g., LLMs) as promising tools for incorporating such knowledge into RL agents. Given the evidence discussed in Section 2.1 that sensorimotor experience may be essential for grounding LLMs—especially for functional grounding—and the evidence presented in this section on the effectiveness of RL for solving tasks and supporting grounding, the present research investigates how LLMs could leverage RL for functional grounding by learning to use language to solve tasks in interactive environments. We argue that current LLMs' abilities invite us to study functional grounding in complex environments featuring a potentially infinite number of goals. In such a context, curiosity-driven RL has been shown to play a major role in both human and artificial learners.

2.3 LLM agents

The impressive capabilities exhibited by LLMs have inspired researchers to apply them beyond the traditional scope of NLP. One particularly challenging area is robotics, where the goal is to build agents that can understand and execute natural language instructions. The ability of LLMs to process linguistic input and encode commonsense knowledge has proven especially valuable for robots operating in everyday environments such as homes. More recently, the integration of LLMs into agents capable of executing actions has expanded to a broader range of applications, driven in part by LLMs' capacity to generate executable code. In this section, I provide a brief overview of how LLMs have been used in embodied agents. I then examine current approaches that employ RL to fine-tune LLMs. While much of this work has focused on RL for language generation in disembodied LLMs, a newer line of research—initiated notably by our GLAM contribution (see Section 3)—has begun to explore the use of LLMs as RL policies interacting directly with an environment.

2.3.1 Embodied LLMs

An initial line of research explored the integration of LLMs into embodied agents, including real-world robots and virtual agents in video games. These efforts primarily focused on using LLMs for planning, i.e., selecting a sequence of actions to accomplish a task. Early approaches such as Huang et al. (2022) and Ahn et al. (2022) relied on sensorless LLMs that generate action plans based solely on a given goal, without access to observations of the environment. Subsequent work introduced feedback mechanisms whereby the LLM receives textual observations—either directly from the environment or through a perception module such as a VLM—and can invoke itself recursively (Huang et al., 2023; Zeng et al., 2023; Yao et al., 2022). This recursive planning strategy was further expanded in recent studies (e.g., (Wang et al., 2023b,a,c)).

In these frameworks, LLMs have been employed either to directly generate complete action sequences or to propose candidate actions for selection by another module. For example, Ahn et al. (2022) used an LLM to estimate the probability of success for each high-level action, which was then combined with a separate estimation of the feasibility of executing that action using a corresponding low-level policy (see Figure 2.10). The final action was selected based on the highest combined probability, a strategy designed

to mitigate the selection of infeasible actions by a "blind" LLM. Similarly, Yao et al. (2020) leveraged an LLM to filter the action space of an RL policy operating in a textual environment, conditioning on both the goal and the current observation. These studies underscore the value of exploiting the commonsense knowledge encoded in LLMs. While many of these approaches relied on frozen LLMs, others introduced learning components using BC, either by fine-tuning the LLM itself (Yao et al., 2020; Wang et al., 2022) or by training additional neural networks on top of it (Li et al., 2022).

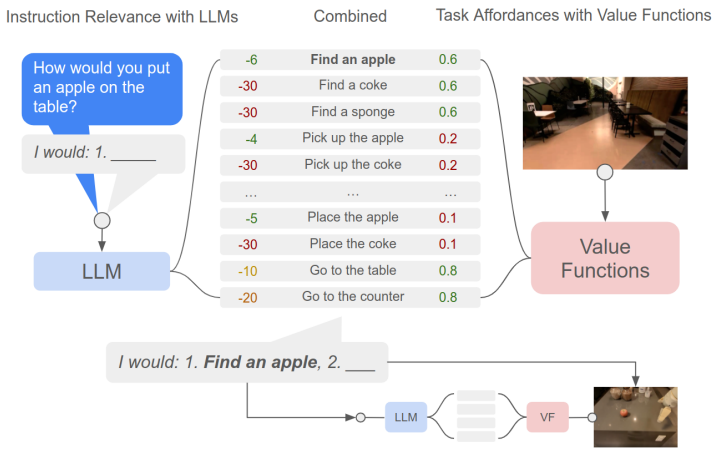


Figure 2.10: In SayCan (Ahn et al., 2022), an LLM is used to score a set of possible actions (i.e., compute the probability of each action's token sequence to follow a prompt). Each action is associated with a low-level RL policy (that was trained beforehand), which will control the robot to perform the action. However, the LLM has no access to any observation of the environment and might assign a high score to something that is not possible. Each action probability computed by the LLM is mixed with the value of the associated RL policy, which has access to a visual observation of the environment. The value is considered here as an affordance score: how likely the action is achievable in the current scene. This mix allows for the leveraging of commonsense knowledge of LLMs (e.g., for cleaning a table, a sponge is useful) while selecting actions that are feasible.

Beyond robotics, a recent and rapidly growing body of work investigates the use of LLM-based agents to control computers. This includes interactions with search engines (Lazaridou et al., 2022; Yao et al., 2022), calculators (Schick et al., 2023; Kadlčík et al., 2023), web browsers (Yao et al., 2022; Nakano et al., 2022; Zhou et al., 2024a), and user interfaces more broadly (Xie et al., 2024; Zhang et al., 2025). A core challenge in building LLM agents for such tasks lies in defining an appropriate textual action space that the LLM can interact with. In simulated environments, for example, Huang et al. (2022) or Wang et al. (2022) highlighted the difficulty of mapping free-form LLM outputs to the discrete actions that the environment supports.

To address this issue, Ahn et al. (2022) proposed an alternative strategy called "scoring," where the LLM does not freely generate an action but instead computes the probability of each possible action from a predefined set, given a prompt. This approach ensures that only valid actions are considered, but it suffers from scalability limitations since it requires computing probabilities for all candidate actions at each step. A third and

increasingly popular solution leverages the LLMs' strong capabilities in code generation (Du et al., 2024). Code offers a general and expressive medium for controlling software and hardware systems, and thus a growing number of recent works build LLM agents that act by generating executable code (Liang et al., 2023; Yang et al., 2023; Liu et al., 2024).

2.3.2 RL applied to LLMs

The rapid development of situated LLM agents naturally raises the question of whether reinforcement learning has been applied to optimize their behavior. However, the application of RL to LLMs is not new. In fact, RL was already extensively studied in the context of natural language generation (NLG), even before the emergence of LLMs, as a means to learn how to generate words (or tokens) sequentially.

RL for natural language generation

NLG can be framed as a sequential decision-making problem: at each step, a model selects the next word (or token) based on the prior context and an objective, or functional meaning. Given this structure, applying RL appears natural and has been extensively explored for training RNNs in NLG tasks (e.g., (Jaques et al., 2017; Bahdanau et al., 2017; Ranzato et al., 2016; Paulus et al., 2018; Scialom, 2022)). In particular, early RNN-based language models trained solely with causal language modeling (CLM) yielded suboptimal results, and fine-tuning them using RL demonstrated promising improvements (Ranzato et al., 2016; Scialom, 2022). Although LLMs later showed that scaling CLM to larger architectures and datasets produces strong NLG capabilities, the generated outputs can still diverge from human expectations (Ouyang et al., 2022).

Prior to the widespread adoption of LLMs, a line of work focused on using RL to fine-tune smaller RNN-based models to better align with human preferences (Stiennon et al., 2020; Ziegler et al., 2020). These approaches relied on human annotators to score or rank model outputs, from which a reward model was trained. Ouyang et al. (2022) applied this approach to LLMs with the goal of improving instruction following. They introduced a now-standard procedure known as RLHF, combining PPO with a Kullback-Leibler (KL) penalty to prevent excessive divergence from the original language model's behavior (Jaques et al., 2017). RLHF has since become a central component in LLM training pipelines. Several variants have been proposed that modify or challenge this procedure (Ramamurthy et al., 2023; Gulcehre et al., 2023; Rafailov et al., 2023; Ahmadian et al., 2024; Singh et al., 2024; Azar et al., 2024). For example, Rafailov et al. (2023) argue that RLHF does not involve true sequential interaction, as the reward is given only after generating the full output. They frame the problem instead as a contextual bandit and propose the RLOO algorithm. Additionally, Ahmadian et al. (2024) contend that learning a reward model is unnecessary and show that it is possible to directly optimize human rankings—a perspective further analyzed by Azar et al. (2024).

Beyond alignment, RL has also been explored as a means of enhancing LLMs' reasoning abilities. For example, Uesato et al. (2022) and Havrilla et al. (2024) investigated how RL could improve mathematical problem-solving by LLMs, while DeepSeek-AI et al. (2025) demonstrated that carefully designed reward functions can guide LLMs to produce

intermediate reasoning steps before answering. Overall, this use of RL to fine-tune LLMs marks a shift toward more active learning paradigms. In particular, RLHF can be seen as a primitive form of social learning, as it optimizes the model to produce responses aligned with human preferences. While this constitutes a step toward grounding in social interactions, it remains far removed from the rich, multifaceted social experiences through which children acquire language.

RL for embodied LLMs

Embodied LLMs are defined by their interaction with an environment—through sensing and acting. Early works such as Ahn et al. (2022), Huang et al. (2023), and Yao et al. (2022) demonstrated that LLMs can exploit the knowledge acquired through large-scale CLM to serve as decision policies in interactive environments. However, the internal representations developed during CLM training are often misaligned with the state and dynamics of the target environment. While incorporating environment observations can help LLMs adapt behaviorally, these adjustments are not persistent, as they do not modify the model's internal parameters—hence failing to achieve lasting grounding. To our knowledge, we introduced the first instance of a situated LLM that learns through online reinforcement learning to solve tasks within an environment. Our method, GLAM (see Section 3.2), converts the LLM into a policy by scoring each possible action—following the SayCan paradigm—and fine-tunes the LLM via PPO. Through on-policy RL, we demonstrate that the LLM progressively aligns its internal representations with the environment's structure, enabling effective control and task-solving. Crucially, we show that this active learning scheme results in improved grounding compared to passive imitation via BC.

Following its introduction, GLAM has been extended by several works. Tan et al. (2024) proposed normalizing action probabilities output by the LLM using action length. In SAC-GLAM, presented in Section 5.1, we introduced an off-policy alternative to PPO. Wen et al. (2024a) and Wen et al. (2024b) explored the granularity of RL application—contrasting token-level and action-level optimization—and found that token-level RL may provide more precise credit assignment. Similarly, Zhou et al. (2024b) proposed a hybrid architecture in which the LLM policy operates at the token level, while another LLM is used to estimate action-level values. Recent work by Aissi et al. (2025) extended GLAM to visual environments by using a VLM to translate visual input into text for the LLM policy. Concurrently, Zhai et al. (2025) investigated applying PPO directly to a VLM used as the agent's policy, further requiring the model to generate reasoning steps before selecting an action. Szot et al. (2024) also used RL to fine-tune VLM agents, but chose to train neural networks added on top of the VLM for the policy and value (and keep the VLM frozen).

In parallel, several studies explored offline RL for LLM-based agents—i.e., leveraging previously collected data without further interactions with the environment (Levine et al., 2020). Notable contributions include Snell et al. (2023) and Abdulhai et al. (2023), with Kumar et al. (2022) showing that offline RL can outperform BC. However, this dissertation focuses on active learning schemes, where the LLM learns by interacting with its environment. Offline RL will therefore not be further addressed. Finally, online RL has recently been extended to LLM agents interacting with digital interfaces. For instance, Bai et al. (2025) apply Advantage Weighted Regression to train a VLM policy

on both online and offline smartphone interaction data. In a similar vein, Chen et al. (2025) proposed a policy gradient approach to train LLMs to interact with APIs through trial-and-error.

2.3.3 Building autotelic LLM agents

Whether interacting with an external environment or generating text completions, current LLM agents trained with RL have so far been explored only in constrained settings, where the space of tasks remains limited—such as a fixed dataset of math problems or a single navigation environment. However, the remarkable capabilities of LLMs naturally invite scaling these agents to much larger, more diverse task spaces. Crucially, tasks for LLM agents are typically defined using text (spanning from natural language to programming or formal languages), which—thanks to its expressiveness and compositionality—offers a unified task space. Yet this same expressiveness implies a practically infinite goal space, posing a key challenge: how can we efficiently train LLM agents within such a vast space under finite resource constraints?

As discussed in Section 2.2.3, humans are autotelic learners—they self-define, select, and pursue goals throughout life, despite facing an infinite set of possible goals. Inspired by this, autotelic RL agents have been designed to explore and learn efficiently in large goal spaces. A central question in this manuscript is whether these autotelic RL approaches can be extended to LLM agents operating over natural language-defined tasks. When presenting autotelic RL agents, we identified multiple key challenges, namely: 1) defining G, 2) defining R, 3) defining a goal-selection scheme, and 4) learning a policy π. In the case of LLM agents, G encompasses all goals defined with text. While this makes G extremely expressive and flexible, it also renders the goal-selection process essential. Most existing approaches to automatic curriculum learning rely on random exploration over a pre-defined goal space—a strategy that does not scale well to natural language due to its combinatorial explosion.

Recent work has begun to address this. For instance, Zhang et al. (2024b), Klissarov et al. (2023), and Sancaktar et al. (2025) proposed using LLMs themselves as *models of interestingness*—filters that constrain the space of explored goals to those considered "interesting" by the LLM. This approach aligns well with how humans often explore within structured cultural or social constraints (e.g., narratives, tools, or games) (Bruner, 1990). Moreover, it opens new possibilities for defining R, by leveraging LLMs' ability to recognize goal success or evaluate outcomes in natural language. However, these approaches typically rely on frozen LLMs, whose understanding may be limited—especially in open-ended learning scenarios where both the task space and the agent's abilities evolve over time. Addressing this limitation requires a crucial step: enhancing the metacognitive abilities of LLMs. In humans, metacognition plays a central role in autotelic behavior, allowing individuals to assess their own competence and strategically select what to learn next. Similarly, navigating the vast and unbounded goal space available to LLMs demands mechanisms for evaluating one's own functional competence and identifying suitable learning targets. Importantly, augmenting LLMs with such self-evaluation capabilities has also been recognized as a key ingredient for building safer and more trustworthy AI systems—models that can detect and communicate when their abilities fall short

(Johnson, 2022; Steyvers & Peters, 2025; Johnson et al., 2025). The research presented in Chapter 5 directly tackles this challenge by introducing methods that enable LLMs to estimate their own functional competence through online interactions.

Finally, this manuscript proposes several approaches for learning the policy, i.e., improving the LLM's functional competence over time. While methods such as GLAM and its derivatives mostly employed on-policy RL (e.g., PPO), the nature of autotelic learning introduces inherent sparse reward challenges. Agents may frequently self-select goals beyond their current competence, making reward signals rare or uninformative. To address this, existing autotelic RL approaches often rely on off-policy RL and hindsight relabelling to better utilize collected trajectories. Notably, these methods often assume social learning contexts, where external cues help reinterpret failed attempts. Colas et al. (2023) explored how an LLM could be used to automatically relabel failed trajectories, interpreting what might have been achieved instead of the originally intended goal. However, just as with reward functions, how to continuously improve LLM-based relabelling mechanisms remains an open problem—especially in the context of continual and open-ended learning.

To conclude, LLM agents—i.e., language models that can sense and act within an environment—have seen significant development in recent years. Their applications now span a broad spectrum, from robotics to search engines and web browsing. However, early work primarily focused on frozen LLMs, whose internal representations often remain misaligned with the external environments they interact with. The use of RL with language models has a well-established history, particularly for improving NLG. This includes leveraging RL to align LLM outputs with human preferences or to enhance reasoning capabilities. With respect to LLM agents, the present research introduced the first method to align their internal representations with the external environment using online RL. However, despite the ongoing active research in this direction, much remains to be done to scale RL-based approaches to LLM agents that face a possibly infinite number of tasks.

Chapter 3

Functional grounding of LLMs through online RL

Contents

3.1 Functional grounding: external dynamics and goals 42

3.2 GLAM: Functional grounding of LLMs in interactive environments with online RL 43

3.2.1 Related work 44

3.2.2 Grounding LLMs with online RL (GLAM) 46

3.2.3 Experiments 48

3.2.4 How fast can an LLM adapt and learn to solve tasks? (Q1) . 50

3.2.5 Q2. Generalization to new objects 53

3.2.6 Q3. Generalization to new tasks 54

3.2.7 What is the impact of using RL vs BC for grounding? (Q4) . 55

3.2.8 Conclusion 55

3.3 An analysis of functionally grounded representations and knowledge in LLMs 56

3.3.1 Related work 57

3.3.2 Problem statement and methods 58

3.3.3 Contrastive learning 59

3.3.4 Experimental Protocol 60

3.3.5 Task-solving abilities with respect to prompt formulation . . . 61

3.3.6 Analyzing the functionally grounded latent representations . . 62

3.3.7 Environmental knowledge acquired through functional grounding 65

3.3.8 Conclusion 66

3.4 Discussion 67

In this chapter, we investigate the use of RL and online interactions with an environment for the functional grounding of LLMs (see Figure 3.1). We begin by introducing the concept of functional grounding, the grounding of functional competence, and examining its relationship to the broader symbol grounding problem (Section 3.1). We then present the first method for achieving functional grounding of LLMs through online RL: GLAM (Section 3.2). GLAM is the first study to explore embodied LLMs that update both their

strategy and internal representations through online interactions with an environment and RL. Using an interactive textual environment designed to study functional grounding, along with a series of spatial and navigation tasks, we evaluate: (1) GLAM's ability to update the LLM and discover optimal goal-solving strategies, (2) the generalization capabilities of the functionally grounded LLM, and (3) the benefits of active learning via RL compared to BC.

Following this, Section 3.3 analyzes the impact of functional grounding on LLMs. We propose an experimental protocol to evaluate the internal representations in functionally grounded LLMs. Our approach specifically investigates what information LLMs extract from environmental observations when solving tasks (i.e., their functional competence), and how functional grounding affects their broader understanding of environmental dynamics. To this end, we conduct a large-scale study spanning multiple LLM architectures, environments, and prompt formulations. Our findings reveal that LLM performance declines significantly when presented with prompt formulations that differ from those seen during training. However, we show that increasing prompt diversity—particularly through contrastive learning—enhances the robustness of functionally grounded LLMs. This results in more consistent information-seeking behaviors and improved comprehension of the environment.

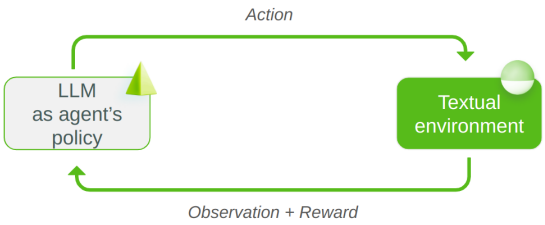


Figure 3.1: In this chapter, we study the functional grounding of LLMs by using them as an RL agent's policy that interacts with textual environments.

Collaborations and scientific output – This chapter stems from a collaborative project initiated in October 2022 by Thomas Carta and myself. At the time, I was exploring embodied LLM agents, while Thomas was focused on RL in textual environments. Our shared interest in fine-tuning LLM agents with online RL led to a joint project, to which we contributed equally—from conceptualization to experimental execution and paper writing. On the experimental side, Thomas primarily developed the BabyAI-Text environment, while I implemented the distributed training setup for fine-tuning LLMs with online RL. This collaboration also included all our respective PhD advisors, who were actively involved in shaping the research direction and co-authoring the paper. The resulting work was published in 2023 as a conference paper (detailed in Section 3.2): *Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In Proceedings of the 40th International Conference on Machine Learning (ICML), Vol. 202., Article 150, 3676–3713.* Following this publication, we identified several promising follow-up directions. One such direction focused on investigating how online RL-based fine-tuning affects the internal knowledge and representations of LLMs. This became the first PhD project of Salim Aissi, who had recently begun his PhD under the supervision of Olivier Sigaud (in 2023). I actively contributed to defining the research direction, designing experiments, and setting up the training infrastructure (including re-running GLAM on BabyAI-Text and additional environments). I also participated in the writing of a paper published in 2024 as a conference paper: *Mohamed*

Salim Aissi, Clément Romac, Thomas Carta, Sylvain Lamprier, Pierre-Yves Oudeyer, Olivier Sigaud, Laure Soulier, and Nicolas Thome. 2025. Reinforcement Learning for Aligning Large Language Models Agents with Interactive Environments: Quantifying and Mitigating Prompt Overfitting. In Findings of the Association for Computational Linguistics: NAACL 2025, pages 7030–7046. Section 3.3 presents a modified version of that paper, which I restructured and rewrote to align with the narrative of this thesis.

3.1 Functional grounding: external dynamics and goals

As discussed in Chapter 2, the symbol grounding problem, originally formulated by Harnad (1990), concerns how to connect the internal symbols processed by a system to the external world. Grounding is considered essential for symbols to acquire meaning. In the literature, language grounding has referred to a range of related objectives (Thill et al., 2014). For example, one well-studied goal is the association of "elementary" symbols—such as the names of objects—with invariant patterns in high-dimensional perceptual modalities like vision (Cangelosi et al., 2010). This type of grounding, known as direct grounding, has been the focus of extensive research and has led to a variety of successful approaches, ranging from VLMs (Cho et al., 2021; Tsimpoukelli et al., 2021; Alayrac et al., 2022; Laurençon et al., 2023; Chen et al., 2023) to robotic systems (Cangelosi & Stramandinoli, 2018). A complementary research direction involves grounding abstract concepts in terms of more elementary ones—what Cangelosi & Stramandinoli (2018) refer to as grounding transfer. This process addresses how symbolic meaning can propagate through internal symbolic relations. Distributional semantics has been particularly influential in this context, demonstrating that statistical co-occurrence patterns can capture conceptual relationships in language (Turney & Pantel, 2010; Boleda, 2020).

Such connections between words and environmental properties correspond to what Roy (2005b) terms *referential meaning*. However, Roy (2005b) also highlights a complementary dimension of meaning: *functional meaning*. While language serves to describe and refer to experiences, it is also used as a tool to act upon and control the environment in order to achieve goals. We argue that developing functional competence—as defined by Mahowald et al. (2024)—in computational models requires addressing a distinct form of grounding. In particular, a central challenge of grounding functional competence lies in understanding how internal symbol manipulations can effectively model, predict, and control external physical or social processes. For such competence to emerge, these internal processes must be aligned with and constrained by the dynamics of the external world. We refer to this alignment as *functional grounding*.

A key characteristic of functional grounding is that it inherently involves sequential decision-making—using language appropriately to achieve specific goals. As discussed in Chapter 2, multiple approaches exist for learning such decision-making strategies. In particular, we emphasized the power of active learning through interaction with the environment, as opposed to passive imitation learning. Unlike direct grounding or grounding transfer—which can often rely on static, pre-collected data—functional grounding demands exploration. It requires the agent to actively engage with its environment and update its internal representations based on the outcomes of its actions. This makes interactive learning mechanisms, such as RL, particularly well-suited for achieving functional

grounding.

3.2 GLAM: Functional grounding of LLMs in interactive environments with online RL

We begin our empirical contributions to functional grounding through online RL with the introduction of the Grounded LAnguage Model method, or **GLAM**. We consider interactive textual worlds Côté et al. (2019); Jansen (2022), which are precisely designed to focus on higher-level forms of grounding, such as functional grounding. In textual worlds, environments can encode rich forms of physical structures inspired by the ones in the human world, e.g. Wang et al. (2022), yet agents act and perceive in these environments only through the textual modality. In this context, this contribution aims to make progress towards the following open question for functional grounding: how could LLMs be used as agent policies producing actions towards goals in interactive environments, perceiving the outcome of these actions, and incrementally grounding and updating their knowledge with the new observations they collect?

Building on works successfully using RL to fine-tune LLMs for natural language generation tasks (e.g., (Stiennon et al., 2020; Ouyang et al., 2022; Ramamurthy et al., 2023)), we propose the first study about functional grounding of LLMs through incremental online RL. In particular, we aim at empirically answering the following open scientific questions:

• **Q1. Sample efficiency** How fast can an LLM adapt and learn to solve various spatial and navigation problems specified in natural language? How does the use of pre-trained knowledge from LLM boosts sample efficiency?

• **Q2. Generalization to new objects**: Once functionally grounded, how can an LLM generalize to various kinds of changes about objects, yet staying in trained tasks?

• **Q3. Generalization to new tasks**: How can such an interactively trained LLM perform zero-shot generalization to new tasks? How does generalization depend on the kind of new tasks?

• **Q4. Impact of online interventions**: What is the empirical impact of grounding using online RL with incremental interactions in comparison with offline BC from a dataset of expert trajectories?

To answer these scientific questions, Section 3.2.3 studies GLAM, our method for functional grounding of LLMs (see Figure 3.2 and Section 3.2.2) on BabyAI-Text, a textual version of the BabyAI environment (Chevalier-Boisvert et al., 2019). Additionally, we help the RL community further develop grounding techniques for LLMs in interactive environments by releasing, in addition to the code of this work[1], a Python library named *Lamorel*[2] facilitating the use of LLMs at scale for RL practitioners. While many tools exist for LLMs and NLP tasks, moving to an RL setting with interactive environments

[1] https://github.com/flowersteam/Grounding_LLMs_with_online_RL
[2] https://github.com/flowersteam/lamorel

requires adaptations, making previous tools not well suited for RL practitioners (see Section 3.2.2).

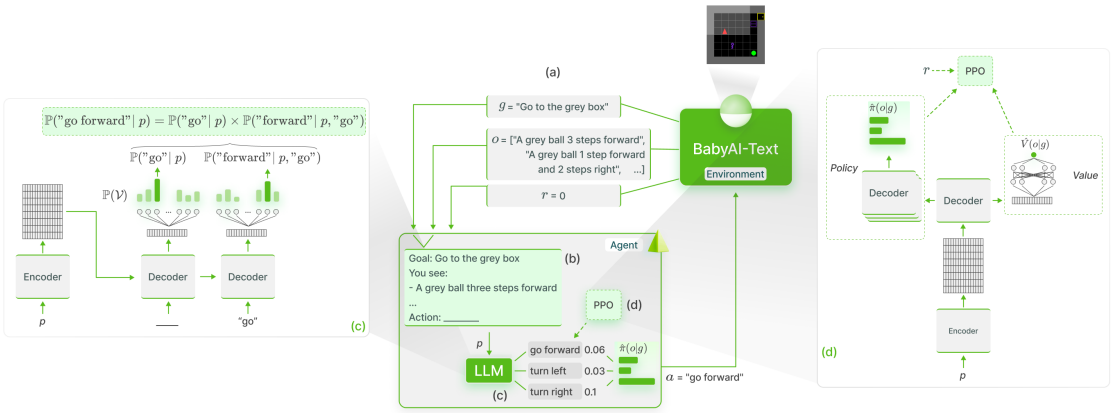


Figure 3.2: **The GLAM method: we use an LLM as agent policy in an interactive textual RL environment (BabyAI-Text) where the LLM is trained to achieve language goals using online RL (PPO), enabling functional grounding**. (a) BabyAI-Text provides a goal description for the current episode as well as a description of the agent observation and a scalar reward for the current step. (b) At each step, we gather the goal description and the observation in a prompt sent to our LLM. (c) For each possible action, we use the encoder to generate a representation of the prompt and compute the conditional probability of tokens composing the action given the prompt. Once the probability of each action is estimated, we compute a softmax function over these probabilities and sample an action according to this distribution. That is, the LLM is our agent policy. (d) We use the reward returned by the environment to fine-tune the LLM using PPO. For this, we estimate the value of the current observation by adding a value head on top of our LLM. Finally, we backpropagate the gradient through the LLM (and its value head).

3.2.1 Related work

Language-conditioned RL – We position this work in the language-conditioned RL setting, where an *instruction-following* agent learns a policy that executes actions in an interactive environment in order to fulfill a language instruction (Luketina et al., 2019). While several works studied this setting for various tasks in 2D or 3D environments (Hermann et al., 2017; Bahdanau et al., 2019a; Colas et al., 2020; Chevalier-Boisvert et al., 2019), we here focus on text-only interactions (i.e. performing textual commands given textual observations), as in Shridhar et al. (2021). However, our work studies how LLMs can not only encode this instruction (Hill et al., 2020b) but also be directly used as agent policies choosing actions given the observation.

Textual environments for RL – Many text-only environments have been used and developed, as discussed in Chapter 2. They usually implement high-level text commands along with very large action spaces and complex dynamics between entities, often aiming

to study functional grounding of policies dealing with high-level actions. While these environments offer interesting properties, we chose to introduce a new environment specifically designed for the systematic analysis of functional grounding. We focused on lower-level navigation skills in spatial environments (which lacks in most textual environments as the agent can usually change rooms, and has direct access to all objects in a room). Moreover, several ablation studies shown in Appendix A.2.6 required precise control over the procedural generation (usually not offered by textual environments). For these reasons, we adapted the BabyAI platform (Chevalier-Boisvert et al., 2019) into a procedural text-only version that enables decoupling exploration challenges from perception challenges. Additionally, we are still able to use BabyAI's visualization tools to analyze trajectories (see Figure 3.2).

Foundation models for decision making – Among the line of work using foundation models (in particular LLMs) for decision-making in embodied agents, SayCan (Ahn et al., 2022), Code as Policies (Liang et al., 2023), and Inner Monologue (Huang et al., 2023) used LLMs as high-level planners in robotics setups. Because their LLM is not directly used as an agent policy for low-level actions and is not grounded using its interactions with the environment, Ahn et al. (2022) had to use an external affordance function to re-rank the actions proposed by the LLM. Similarly, Yao et al. (2022) also featured a closed-loop feedback between an LLM that is the planner and an agent that is the actor, but this time in a textual environment. Expanding on this, Dasgupta et al. (2022) added a reporter observing the environment and reporting useful information to the planner. While hinting at the usefulness of prior knowledge contained in LLMs for embodied tasks, these works are limited by the absence of grounding. Second, several works proposed to first fine-tune LLMs on expert trajectories before using them in the environment. Using their ScienceWorld benchmark, Wang et al. (2022) showed that LLMs fine-tuned using BC performed worse than a much smaller and randomly initialized Deep Q-Network trained using RL supporting the hypothesis that grounding in the environment through direct interactions is crucial. Reid et al. (2022) reused LLMs to perform offline RL in non-linguistic environments leveraging the internal structures learned by LLMs but no longer using words or symbols they were trained to manipulate (Takagi (2022) investigated how these internal structures can be relevant for unrelated tasks). Finally, one may also pre-train a policy using BC or offline RL from expert trajectories before fine-tuning it with interactions with an environment. Related to our work, the Online Decision Transformer (Zheng et al., 2022) first uses offline RL to pre-train a transformer model and eventually fine-tunes it with online RL. But compared to our study, they did not use a general CLM pre-training objective and therefore did not study functional grounding of language symbols.

Fine-tuning LLMs with RL – RL applied to LLMs was used in particular to improve alignment between generated text and human preferences (Stiennon et al., 2020; Ouyang et al., 2022). In RLHF, text generation is viewed as a sequential decision-making problem where each "action" of the LLM is a new token and the "state" corresponds to the prompt. Most of these methods used PPO (Schulman et al., 2017) to fine-tune their LLMs using a reward function learned on a dataset of collected human interactions. With this technique, Ouyang et al. (2022) managed to generate more human-aligned outputs despite having a model (InstructGPT) with 100 times fewer parameters than GPT-3

(Brown et al., 2020). While our work shares the PPO-based fine-tuning with RLHF, our setup diverges from it in multiple aspects. First, our LLM is functionally grounded using an external task-conditioned reward from the environment (which happens to be sparse in our BabyAI-Text environment) and not a learned reward model. Second, the RLHF setup has no external environment dynamics controlling the next state given a previous state and an action (the next state in RLHF is just the previous state with the last generated token appended). In comparison, our work exposes an outer loop controlled by the environment whose dynamics, providing the next state and reward, are unknown to the LLM (in comparison to RLHF where the RL loop is an inner loop in the token generation process).

3.2.2 Grounding LLMs with online RL (GLAM)

We introduce the GLAM method (for Grounded LAnguage Models) where an LLM is used as agent policy and is functionally grounded in an interactive environment using online RL by leveraging collected observations and rewards to improve itself towards achieving goals formulated in language. We detail this method in the following paragraphs and redirect the reader to Figure 3.2 for a schematic view. We first formalize the textual RL problem we tackle (a). Then, we detail how we use an LLM as agent policy to interact with BabyAI-Text (b, c). Finally, we explain how online RL fine-tuning is used to ground the LLM in BabyAI-Text (d).

Problem statement

We assume a textual RL setting where, given a language vocabulary $\mathcal{V}$, our environment returns an observation $o \in \mathcal{V}^N$ and a reward $r \in \mathbb{R}$ following an action $a \in \mathcal{A} \subset \mathcal{V}^N$ (i.e. actions are sequences of tokens). We also assume a task or goal description $g \in \mathcal{G} \subset \mathcal{V}^N$ which conditions the reward. Such an environment can be framed as a goal-augmented Partially Observable Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G}, \mathcal{O}, \gamma)$ with $\mathcal{S}$ the state space, $\mathcal{A} \subset \mathcal{V}^N$ the action space, $\mathcal{G} \subset \mathcal{V}^N$ the goal space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \mapsto \mathbb{R}$ the goal-conditioned reward function, $\mathcal{O} : \mathcal{S} \mapsto \mathcal{V}^N$ the observation function mapping a state to a textual description and finally γ the discount factor.

In this work, we extend the BabyAI platform (Chevalier-Boisvert et al., 2019) initially designed for grounded language learning and propose a text-only extension named BabyAI-Text. We leverage BabyAI's inner procedurally generated minigrid environment where an agent navigates and interacts with objects through 6 text commands: *turn left, turn right, go forward, pick up, drop and toggle*. We also reuse the set of tasks introduced in BabyAI as well as their associated description along with the sparse scalar reward. Our key difference is the textual description $o \in \mathcal{V}^N$ of the agent's partial observation returned by BabyAI-Text instead of the symbolic representation initially returned by BabyAI (see Appendix A.1.2). We leverage BabyAI-Text in Section 3.2.3 to assess our functional grounding method.

LLMs as policies in interactive environments

In order to use the LLM as the policy in such a textual interactive environment, we gather the task description, the textual description of the current observation and the set of possible actions in a prompt used to feed the LLM. We chose a single arbitrary and simple prompt template (see Appendix A.3 for examples) and did not perform any intensive prompt engineering. Indeed, as we fine-tune the LLM, we expect it to adapt to the chosen prompt template. However, we will show in the next contribution, in Section 3.3, that our fine-tuning not only leads the LLM to adapt to the prompt template but also to overfit to it.

Given this prompt, we now need the LLM to output a probability distribution over the possible actions $\mathbb{P}(\mathcal{A})$. For this, Huang et al. (2022); Li et al. (2022); Wang et al. (2022) used the LLM to generate text. If the generated sequence of characters corresponds to one of the possible actions (i.e. $s \in \mathcal{A}$), this action is chosen by the agent. Otherwise, an ad-hoc mapping must be performed to select an action $a_i \in \mathcal{A}$ given s. As an alternative method, one could also use more standard RL practices by adding action heads - a Multi-Layer Perceptron (MLP) with $|\mathcal{A}|$ outputs - on top of the LLM. Finally, Ahn et al. (2022) proposed to directly use the LLM to compute the (log) probability of each action $a_i \in \mathcal{A}$ by computing the conditional probability of each token in action $a_i = \{w_0, ..., w_{|a_i|}\}$ given the prompt p:

$$\mathbb{P}_{LLM}(a_i|p) = \prod_{j=0}^{|a_i|} \mathbb{P}_{LLM}(w_j|p, w_{<j}) \tag{3.1}$$

with $\mathbb{P}_{LLM}(w_j|p, w_{<j})$ the probability computed by the LLM of token w_j given prompt p and previous tokens $w_{<j}$ (see (c) from Figure 3.2). This method suffers from requiring a forward pass on the LLM for each action to compute the probability of its sequence of tokens (especially in comparison to new action heads that require a single forward pass on the prompt to compute all actions' probability). However, it also has several advantages, in particular, 1) there is no need of potential ad-hoc mapping as when text is generated, 2) we use only pre-trained operations from the LLM and leverage language modeling heads' prior and 3) this method is robust to any action space and can thus be used on any textual environment with no change.

For these reasons, we chose the latter method. We first use log probabilities instead of normalized probabilities using $\mathbb{LP}_{LLM}(a_i|p) = \sum_{j=0}^{|a_i|} log\ \mathbb{P}_{LLM}(w_j|p, w_{<j})$ in replacement of $\mathbb{P}_{LLM}(a_i|p)$ to avoid multiple normalization operations. We eventually normalize all the log probabilities to obtain a distribution over $\mathcal{A}$:

$$\mathbb{P}(a_i|p) = \frac{e^{\mathbb{LP}_{LLM}(a_i|p)}}{\sum_{a_j \in \mathcal{A}} e^{\mathbb{LP}_{LLM}(a_j|p)}}. \tag{3.2}$$

PPO fine-tuning

We now propose to leverage experiences gathered by the LLM to perform functional grounding. More formally, we aim to learn a policy $\pi : O \times \mathcal{G} \mapsto \mathbb{P}(\mathcal{A})$ that maximizes the expected discounted sum of rewards for any given goal $g \in \mathcal{G}$. We use for this the PPO

algorithm (Schulman et al., 2017) that both learns a policy $\hat{\pi} : O \times G \mapsto \mathbb{P}(A)$ and a value function $\hat{V} : \mathcal{O} \times \mathcal{G} \mapsto \mathbb{R}$ approximating the true value $V(s, g) = \mathbb{E}_{a \sim \hat{\pi}(\mathcal{O}(s), g)} \big[R(s, g, a) + \gamma V(\mathcal{T}(s, a), g) \big]$.

As mentioned in Section 3.2.2, we compute the probability of each action $a_i \in \mathcal{A}$ using the likelihood computed by the LLM as $\hat{\pi}(a_i|o, g) = \mathbb{P}(a_i|p)$.

For value approximation, we we add an MLP with a single output for the value on top of the last layer of the first Decoder block (i.e. in place of the language modeling heads) in order to compute $\hat{V}(o|g) = \hat{V}(p)$ (see (d) from Figure 3.2). This position is explained by the fact that we use Encoder-Decoder LLMs in our experiments but our method could easily be used with Decoder-only models (as performed in further contributions of this chapter) by attaching the value head to the Decoder block encoding the last token of the prompt.

Distributed LLM policies using *Lamorel*

Using LLMs to compute probabilities over action space is computationally expensive as it requires computing $\prod_{j=0}^{|a_i|} \mathbb{P}_{LLM}(w_j|p, w_{<j})$ for each action $a_i = \{w_0, ..., w_{|a_i|}\}$. When one uses very large LLMs (i.e. more than hundreds of million parameters), computing the probability of a single action already means performing a long forward pass over the whole network. As a result, computing the probability of each possible action at every step becomes very slow. Considering the number of interactions usually required to solve tasks in BabyAI (and by extension BabyAI-Text), performing online RL fine-tuning of LLMs easily became intractable with a single LLM distributed over multiple GPUs. To overcome this, we deployed N LLM workers each handling a subset of actions to score in parallel (allowing a quasi-linear time decrease with N). We add to this distributed inference the possibility to also perform distributed training (i.e. compute the gradient of minibatches in parallel and gather gradients before updating models). We wrap all this in a Python library named *Lamorel* designed for RL practitioners eager to use LLMs. It allows one to use LLMs as black-box but also to perform more advanced methods such as adding new heads on top of them. See Appendix A.5 for more details.

3.2.3 Experiments

We design a set of experiments in BabyAI-Text aiming to provide answers for the scientific questions previously introduced:

• **Q1. Sample efficiency** How fast can an LLM adapt and learn to solve various spatial and navigation problems specified in natural language? How does the use of pre-trained knowledge from LLM boosts sample efficiency?

• **Q2. Generalization to new objects**: Once functionally grounded, how can an LLM generalize to various kinds of changes about objects, yet staying in trained tasks?

• **Q3. Generalization to new tasks**: How can such an interactively trained LLM perform zero-shot generalization to new tasks? How does generalization depend on the kind of new tasks?

• **Q4. Impact of online interventions**: What is the empirical impact of grounding using online RL with incremental interactions in comparison with offline BC from a dataset of expert trajectories?

In these experiments, we use Flan-T5 780M (Rae et al., 2022) for 1) the close link between its training corpus (containing instruction-following documents) and our language-conditioned interactive environment, and 2) its simple open-source access through the Hugging Face tools[3]. We apply our GLAM method to Flan-T5 (which we name GFlan-T5 in experiments below for Grounded Flan-T5) and compare it with three baselines. First, we also train a non-pre-trained Flan-T5 where we only reuse the pre-trained embedding layer and add action heads on top of it (see Figure A.6 in appendices). As for GFlan-T5, we propagate the gradient through the entire graph (including the action heads here). We call this baseline NPAE-Flan-T5 (Non-pre-trained with Action heads and Embedding Flan-T5). We show in Appendix A.2.4 that using a non-pre-trained Flan-T5 while keeping the scoring method fails. We also provide as a more classic RL baseline a DRRN (He et al., 2016) agent of approximately 1M parameters which is often used for TextWorlds. We especially reuse the implementation from Wang et al. (2022), which gave SOTA results and outperformed LLMs. At each step, we feed our 3 agents above with the following prompt template filled using the information returned by BabyAI-Text (see Appendix A.3 for examples):

• A header listing what actions are accessible (but not necessarily useful) in the environment in the form of: *Possible action of the agent: <list of actions>*

• The goal of the agent: *Goal of the agent: <goal>*

• The 3 previous observations and last 2 actions, used as a short-term memory required to complete BabyAI-Text tasks (in comparison, the DRRN uses recurrent layers to deal with short-term memory requirements):
Obs. 0: <description from BabyAI-Text at step $t-2$ *>*
Action 0:<action chosen by the agent at step $t-2$ *>*
Obs. 1: <description from BabyAI-Text at step $t-1$ *>*
Action 1: <action chosen by the agent at step $t-1$ *>*
Obs. 2: <description from BabyAI-Text at step t *>*
Action 2: <the next action to be chosen by the agent>

Finally, as BabyAI-Text simply provides an alternative mapping of observations, we add as an indication the performance of the PPO agent used in (Chevalier-Boisvert et al., 2019) that runs on BabyAI rather than BabyAI-Text (i.e., using symbolic observations instead of textual descriptions) and name this agent Symbolic-PPO in results below. In Appendix A.2.3, we show that symbolic observations provided by BabyAI encode biases that ease learning compared to textual descriptions. However, even with this advantage, GFlan-T5 outperforms Symbolic-PPO in all our setups.

We first study Q1 by training the different agents in a multi-task setting assessing their efficiency at learning the different tasks. We then address questions Q2, Q3 and Q4 using a set of generalization experiments (Figure 3.5) on the zero-shot abilities of the

[3] https://huggingface.co/docs/transformers/model_doc/flan-t5

resulted trained agents mostly inspired from (Colas et al., 2020) and (Valmeekam et al., 2022). We report their average success rate as well as standard deviation. We compare the results of GFlan-T5, DRRN as well as Flan-T5 (i.e. the LLM used in GFlan-T5 but before our fine-tuning) to show how our grounding method impacted it. All results below are given with their 99% confidence interval (mathematical details are given in Appendix A.7).

3.2.4 How fast can an LLM adapt and learn to solve tasks? (Q1)

In order to study question Q1, we train our agents for 1.5 million steps in BabyAI-Text where each episode is a task randomly sampled from the following:

- **Go to <object>**, a simple navigation task that requires reasoning abilities to choose the right plan given objects' position;
- **Pick up <object>**, a reasoning task that combines navigation tasks;
- **Put <object A> next to <object B>**, which requires first reaching <object A>, picking it up, reaching <object B> and finally dropping <object A> next to <object B>;
- **Pick up <object A> then go to <object B>** and **Go to <object B> after pick up <object A>**, both serving to test reasoning abilities on temporal sequences;
- **Unlock <door>**, a task that includes inferring that a key is needed to unlock the door, finding the right key (i.e. the one colored as the door) and eventually using the toggle action with the key on the door.

In each task, the agent must navigate in one procedurally generated room with 8 distractors (i.e. useless objects for the completion of the task).

We plot the mean and standard deviation of the success rate (i.e. 1 if the goal has been reached, 0 otherwise) over 4 seeds of GFlan-T5, NPAE-Flan-T5, DRRN and Symbolic-PPO in Figure 3.3. In addition, we also monitor the evolution of the probability of each possible action on a set of 11 evaluation prompts to assess agents' abilities to solve each task in Appendix A.3. By plotting the evolution of the distribution over possible actions in Figure A.14, we better grasp how and when the agents learn skills (e.g. navigation skills).

Looking at the evolution of the average success rate, GFlan-T5 quickly reaches 0.8 after only 250.000 steps (and 0.9 after approximately 600.000 steps). In comparison, both DRRN and NPAE-Flan-T5 are still under 0.2 after 1.5 million steps. Even when compared to Symbolic-PPO, which uses symbolic observations (easier to process than language as shown in Appendix A.3), GFlan-T5 exhibits a drastically better sample efficiency with Symbolic-PPO almost reaching 0.4 after 1.5 million steps. Figure A.14 and Table A.1 highlight how GFlan-T5 leverages its knowledge about the relationships between entities to learn navigation tasks in less than a hundred updates.

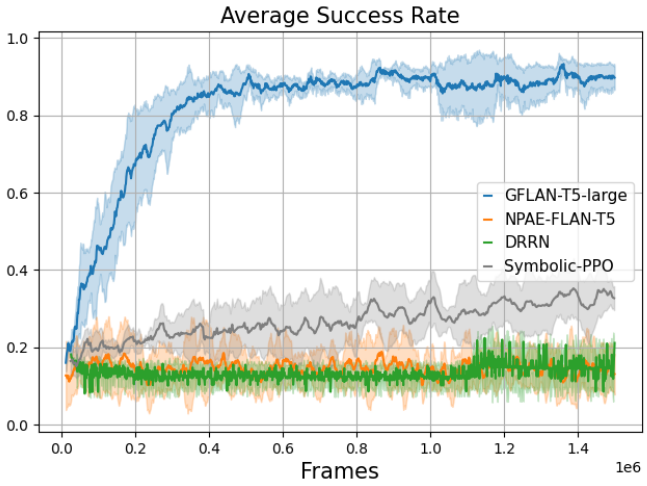


Figure 3.3: **Q1. Sample efficiency**: Evolution over 4 seeds of the average success rate and standard deviation on all Q1 tasks. The details of average success rate calculation over the goals is given in Appendix A.2.2.

The failure of NPAE-Flan-T5 both highlights how GFlan-T5 leverages the LLM's pre-trained knowledge to deal with the proposed tasks and how the fine-tuning method helps achieve the grounding objective. Furthermore, the fact that GFlan-T5 strongly outperforms Symbolic-PPO and the latter is better than NPAE demonstrates how language can be used as a tool to scaffold learning if already acquired. It also explains how counterproductive it can be if one asks an agent to both learn a task and language at the same time (see Appendix A.2.3 for further results).

We now perform a deeper analysis of this sample efficiency by studying the impact of varying the action space and the number of distractors. We provide both the evolution of success rate and a sample efficiency measure SE:

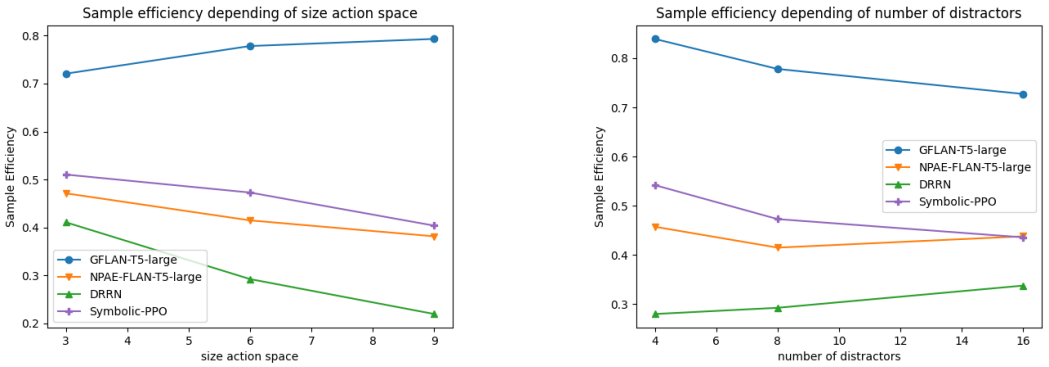
$$SE = \frac{1}{T}\sum_{t=0}^{T} SR_t \tag{3.3}$$

where T is the number of steps or frames seen and SR the success rate at frame t.

Impact of the dimension of the action space

In this experiment, we test the sensitivity of LLMs to the size of the action space by using 3 different action spaces when trained on the *Go to <object>* task:

- The **restricted** action space composed of the only 3 useful actions: `turn left`, `turn right`, `go forward`.
- The **canonical** action space composed of the 6 actions that can be performed in the environment with 3 useful and 3 useless actions that are `pick up`, `drop` and, `toggle` (they are useless here as the agent is only navigating).



(a) Impact of the action space size (3, 6 or 9 actions with always only 3 useful actions).

(b) Impact of the number of distractors.

Figure 3.4: Impact of the action space size and number of distractors on the sample efficiency measure (Equation (3.3)). We report results averaged over 2 seeds for training on the *Go To* task.

- The **augmented** action space composed of 9 actions (3 useful and 6 useless with `pick up`, `drop`, `toggle`, `sleep`, `do nothing` and `think`). The last three actions have been chosen such that they clearly have no use for the *Go To* <*object*> task and consequently should not impact an agent that has knowledge about the world.

We conduct our tests in an environment with 1 room, 8 distractors, and report results in Figure 3.4a. Results show no impact on GFlan-T5, while the performance of other agents decreases with larger action spaces. We hypothesize that this is due to the LLM's ability to discard useless actions quickly at the beginning of fine-tuning.

Impact of the number of distractors

Similarly, we expect LLMs to be less sensitive to variations in task complexity. We assess this by plotting the evolution of sample efficiency (Equation (3.3)) for 4, 8 and 16 distractors. We conduct these tests in an environment with 1 room and observe a slight performance loss from GFlan-T5 when the number of distractors increases (Figure 3.4b). In comparison, Symbolic-PPO degrades as the number of distractors increases with a success rate decreasing by 38% from 4 to 16 distractors whereas the GFlan-T5 success rate only decreases by 14%. We hypothesize that the LLM manages to focus on the relevant aspect of the environment quickly.

Thus, GFlan-T5 seems robust with similar learning curves when one increases the action space size (from 3 to 9 actions with only 3 useful ones) or the number of distractors (from 4 to 16) We also provide in Appendix A.2.5 an additional ablation analyzing the impact of the LLM's size. Results highlight that the number of parameters has a high impact on the learning process. Indeed, we observe a strong difference on sample efficiency and asymptotic performance between a small LLM (80 million parameters) and the 780 million parameters we used here. We also plot the full learning curves for the ablation on the action space size and the number of distractors in appendices A.2.6 and A.2.6, respectively.

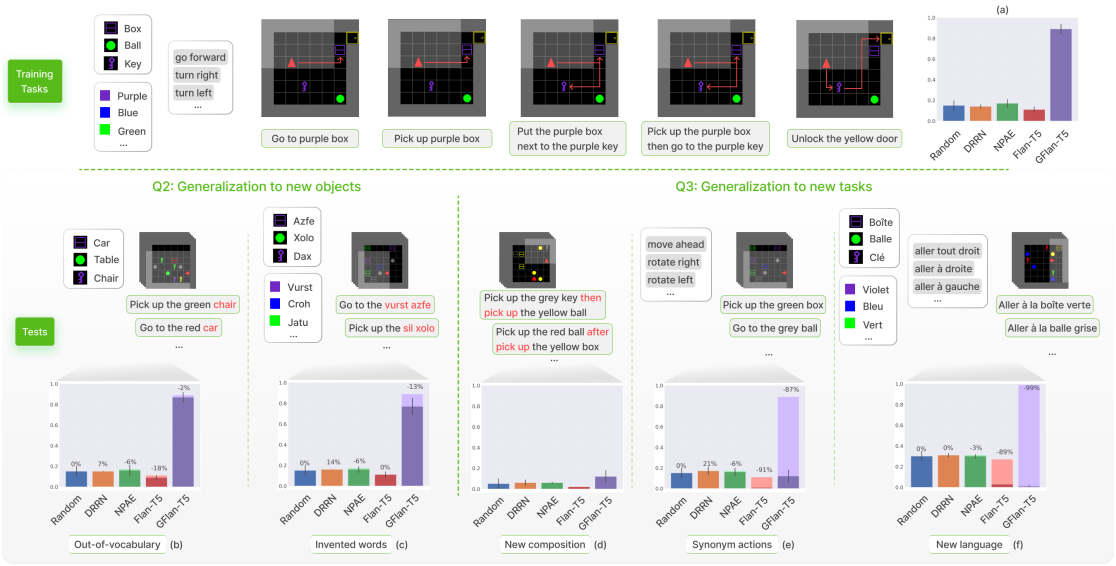


Figure 3.5: **Generalization tests**: We train all agents on a mix of 5 different tasks and evaluate their generalization abilities on 1000 test episodes (also containing a mix of these 5 tasks) (a). We compare them to two baselines: an agent choosing actions randomly (Random) and the zero-shot Flan-T5 (without any fine-tuning). We then perform several generalization studies to answer Q2 and Q3 by (b) substituting object names out-of-vocabulary names, (c) substituting objects and colors by invented words, (d) testing a new composition of tasks, (e) substituting actions by synonyms and (f) translating the whole environment to French for the *Go To* task. For each agent, we plot its mean success rate over 2 seeds along with the confidence interval and the delta with performance on the same task without any change (except for (d), on which no baseline result can be provided as this task is completely new). The details of average success rate calculation over the goals is given in Appendix A.2.2.

3.2.5 Q2. Generalization to new objects

In this section, we analyze how a functionally grounded agent can generalize its skills to new objects. Indeed, we expect our agents to focus on the geometry of the environment (how objects are positioned and how their positioning is described), but not on the identity of the objects themselves (e.g. *Go to <object>* should be achieved even if the object has not been seen during training). We test if this property is present in our trained agents by measuring their zero-shot performance in two environments. First, an environment with nouns not in the training vocabulary (e.g. "tree")[4] and second, an environment with invented objects (made of an invented adjectives and an invented nouns such as *faze dax*).[5] We use the environment the agent has been fine-tuned on (i.e. without any word substitutions) as a control environment. Results in Figure 3.5 (Q2 part) indicate that GFlan-T5 is not affected when tasks contain out-of-vocabulary nouns. Moreover, even if the GFlan-T5's success rate decreases by 13% when it is in an environment with invented

[4]The out-of-vocabulary nouns are given in Appendix A.8.1.
[5]The invented objects are given in Appendix A.8.2.

objects, it still retains strong performances compared to baselines. These results support the hypothesis that GFlan-T5 has functionally grounded the symbols that describe the geometry of the environment and the instructions (e.g. words such as "in front", or the meaning of "steps" as a distance measure)[6].

3.2.6 Q3. Generalization to new tasks

In this Section, we perform generalization tests as in Section 3.2.5, but with new unseen tasks. Using these, we verify to what extent an agent is able to compose and generalize over the symbols it has grounded during fine-tuning.

Table 3.1: Generalization tests for behavioral cloning

	Environments	GFlan-T5	BC-GFlan-T5	BC-Bot	Random
Q4	Go To task no change	0.82 ± 0.02	0.69 ± 0.08	0.73 ± 0.07	0.30 ± 0.05
	Go To task with invented words	0.74 ± 0.004	0.7 ± 0.07	0.63 ± 0.08	

New composition of learned tasks

During fine-tuning, agents learn to do both 1) *Pick up <object A>* and 2) *Pick up <object A> then go to <object B>* or *Go to <object B> after pick up <object A>* tasks. We test in this experiment if an agent can compose grounded symbols to solve the new tasks *Pick up <object A> <then/after> pick up <object B>*. Results in Figure 3.5 (Q3 part) hint that, while all agents fail to solve these new tasks, GFlan-T5 outperforms other baselines by reaching an 0.12 success rate compared to Flan-T5 (0.07) or Random (0.05). These low results can be explained by the fact that none of the agents managed to master the *Pick up <object A> then go to <object B>* or *Go to <object B> after pick up <object A>* tasks during training (see Appendix A.3). More details about the grounding of "then" and "after" are given in the Appendix A.4.4.

Seen tasks with synonym actions

In this task, we test the robustness of our agents to actions by replacing the actions used during training by synonyms. For instance, "go forward" is replaced with "move ahead"[7]. We expect LLMs, which already learned to map words to an embedding space, to also ground synonyms as they ground words of the environment. In this environment (see Figure 3.5 Q3 part), the success rate of GFlan-T5 is 0.12 vs 0.01 for Flan-T5. Thus the grounding of some words (here the actions) also improves the grounding of their synonyms. However, we observe an 87% drop in performance compared to the original settings, which we assume is due to an over-fitting of the actions' vocabulary.

[6] See Appendix A.1.2 for more details on the geometry.
[7] A table giving all the used synonyms is given in Appendix A.8.3

New language

In order to understand how far agents can generalize, we test them with a language not seen during training (French). Knowing that Flan-T5 has been pre-trained with a multilingual corpus and is able to translate simple sentences, we test whether grounding in GFlan-T5 has also impacted its manipulation of other languages. However, we observe that even only for a simple navigation task (i.e. *Go To*), the model fails to generalize to a new language with a success rate (0.02) worse than random (0.30). We hypothesize that when too many grounded symbols are modified at once, functional grounding fails to be transferred to this new subsystem of symbols. Complementary experiments that confirm and reinforce this result are presented in appendices A.4.2 and A.4.3.

3.2.7 What is the impact of using RL vs BC for grounding? (Q4)

Finally, we study how online interactions with an environment, enabling learning through interventions and trial-and-error, improves grounding in comparison to pure BC. We compare a GFlan-T5 trained on the **Go To** task over 400000 steps with two baselines trained with behavioral cloning using 400000 transitions (see Appendix A.6.2). For the baseline called BC-GFlan-T5, transitions are collected from GFlan-T5 fine-tuned on the **Go To** task. For BC-Bot, transitions are collected using the BabyAI procedural bot achieving a success rate of 1.

In Table 3.1, we measure the success rate of GFlan-T5 and the baselines on two tasks: *Go To* and *Go To* with invented nouns and adjectives. First, once can see that GFlan-T5 outperforms all baselines in both tasks. Second, as GFlan-T5 does not achieve a success rate of 1 on the **Go To** task, its collected trajectories for BC can contain deceptive transitions in comparison to the ones collected by the bot. Hence, we obtain the expected result that BC-Bot outperforms BC-GFlan-T5. Finally, we expect our agents not to be affected by an environment where nouns and adjectives are replaced by invented ones in such navigation tasks. Experiments show that GFlan-T5 is less affected ($0.82 \rightarrow 0.74$) than the BC-Bot ($0.73 \rightarrow 0.63$). GFlan-T5 also performs better in the *invented words* task than the BC-GFlan-T5 (success rate of 0.7). These results align with the limits of passive learning discussed in Chapter 2. Moreover, they also align with theoretical and empirical evidence discussed throughout this manuscript about the importance of embodied and active learning in cognitive development and language acquisition.

3.2.8 Conclusion

In this work, we proposed the GLAM method for functional grounding (i.e., aligning internal symbols to external dynamics so that the agent can use them to solve tasks in the environment) of LLMs in interactive textual environments based on online RL. Using our new BabyAI-Text environment, we performed several experiments studying 4 scientific questions. We showed how GLAM, which requires almost no environment-specific modifications on the LLM, enables to drastically improve performances to solve RL tasks in this environment as compared to zero-shot use of the LLM, to supervised fine-tuning, and to RL fine-tuning of non-pre-trained LLMs. We showed how it boosts both sample

efficiency and generalization abilities in zero-shot tests (both to new objects and several new tasks). In addition to these key results, we provided in-depth ablations showing the effect of several parameters (e.g., size) on grounding. We believe this method can act as a milestone towards grounding and using LLMs in interaction with our world. Moreover, from an RL perspective, these results hint that using LLMs as agent policies opens an avenue for escaping the *Tabula-Rasa* RL setting and creating much more sample-efficient RL agents.

However, this study still suffers several limitations. In particular, current experiments are limited to a textual environment, as it allowed us to focus on functional grounding without entangling any other form of grounding. Nonetheless, we believe that studying the functional grounding of VLMs is of high interest and would permit the use of more complex environments. Another limitation is the use of small action spaces and arguably small LLMs. This is notably explained by the computational cost of performing scoring over larger action spaces and fine-tuning larger LLMs. Directions, discussed in this chapter's discussion but not studied in this manuscript, could be explored for the former. One way to reduce the latter (i.e., computational burden of computing gradients over the entire LLM) is to use efficient fine-tuning techniques such as LoRA (Hu et al., 2022). Tan et al. (2024)—as well as the empirical contributions presented in the remainder of this manuscript—use LoRA to apply GLAM to larger LLMs.

Parallel to this, our study did not investigate how functionally grounding an LLM in a specific environment impacts its general knowledge—whether about that environment or more broadly—or its wider capabilities. In the next contribution, we aim to better understand the broader effects of GLAM on LLMs. We begin by examining GLAM's impact on the representational abilities of LLMs, particularly their capacity to generalize across diverse prompt formulations when extracting information from observations. We then assess the extent of environmental knowledge acquired through GLAM.

3.3 An analysis of functionally grounded representations and knowledge in LLMs

In the previous contribution, we demonstrated that GLAM—i.e., fine-tuning LLMs through online RL—can effectively functionally ground language models. We assessed gains in task-solving abilities as well as generalization to environmental changes (e.g., novel objects or tasks). However, it remains unclear how GLAM affects the internal knowledge and representations of the LLM itself. In this work, we introduce an experimental framework designed to explore this question. We apply GLAM to multiple LLMs of varying architectures and sizes across two distinct textual environments. Focusing on the representational effects of functional grounding, we evaluate LLM agents using several prompt templates. For each model, we report performance in four settings: (1) the zero-shot baseline (i.e., before any functional grounding), (2) after grounding with GLAM using a single prompt template, (3) after grounding with GLAM using multiple prompt templates, and (4) after grounding with GLAM using a single prompt template combined with a contrastive regularization loss designed to enforce invariance in the LLM's latent representations across different prompt formulations.

Using this experimental setup, we assess several key capacities of functionally grounded LLMs. We begin by examining the impact of prompt formulation on the task-solving abilities acquired through functional grounding. Next, we propose a set of visualizations aimed at shedding light on how grounded LLMs make decisions—specifically, how they extract and utilize information from their prompts. Finally, we investigate how functional grounding influences the LLMs' broader understanding of their environment. To this end, we conduct question-answering evaluations, where models are asked to respond to queries about their environment—for example, identifying which objects are present or which ones are most useful for completing tasks.

Overall, this work introduces multiple contributions:

- An experimental protocol scaling results presented in the previous section by using multiple LLMs, environments, and varying prompts.
- Two changes to GLAM to deal with diversity in prompts: applying PPO's loss on each prompt or using a contrastive regularization.
- An in-depth study of the impact of functional grounding on representational abilities, namely task-solving abilities and broader comprehension of the environment.

Our findings reveal that the performance of LLMs is highly sensitive to prompt variations, suggesting that fine-tuning with a single prompt template only induces superficial updates and fails to improve the acquisition of new knowledge about the environment. By analogy with observational overfitting in RL (Song et al., 2020b), we refer to this phenomenon as *prompt overfitting*. We also show that the contrastive regularization we propose significantly improves performance and the robustness to prompt variations, as well as the acquisition of new knowledge about the environment (and performs better than applying PPO's loss on all prompt formulations). Altogether, our work[8] contributes to a better understanding of RL-based functional grounding of LLMs in interactive environments.

3.3.1 Related work

LLMs have shown remarkable capabilities to generate text and solve tasks in zero-shot and few-shot scenarios. To enhance their performance and consistency, various prompting methods have been developed Liu et al. (2023a). In addition, multiple studies have highlighted the sensitivity of LLMs to minor perturbations in the prompt, leading to substantially different outputs (Zhao et al., 2021; Sclar et al., 2024; Salinas & Morstatter, 2024). This sensitivity impairs the reliability and robustness of these models. Indeed, certain input-agnostic sequences could trigger specific outputs, further illustrating the brittleness of LLMs to prompt modifications (Wallace et al., 2019). The sensitivity of LLMs persists regardless of model size, the number of examples, or the type of instruction provided (Sclar et al., 2024; Zhao et al., 2021; Loya et al., 2023). These studies also reveal poor performance consistency across models using the same prompt. To improve

[8]Code can be found here.

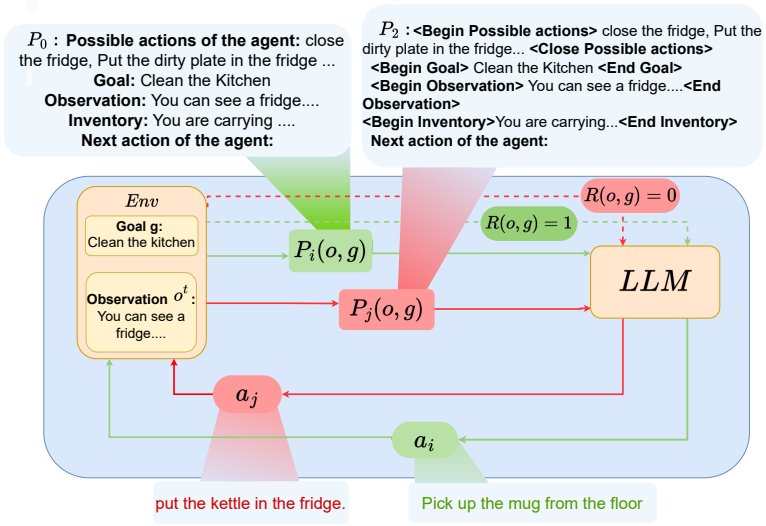


Figure 3.6: We use an LLM as agent policy in a textual environment Env. The Env provides a fixed goal description g for the current episode, a description of the agent's observation o, and a scalar reward $R(o,g)$ for the current step. The goal and observation are formatted using a prompt formulation P_i.

the LLM outputs, a recalibration can estimate and adjust for the model's biases with additional parameters, which mitigates the effects of prompt sensitivity (Zhao et al., 2021). Our work extends this research by evaluating the performance of LLM agents across various prompt formulations and by mitigating prompt overfitting to preserve semantic consistency in interactive environments.

3.3.2 Problem statement and methods

Problem statement

We operate, as in GLAM, in a textual RL setting. Given a vocabulary of tokens V, our experimental framework can be conceptualized as a partially observable Markov decision process $M = (S, A, T, R, G, O, \gamma)$ with S the state space, $A \subset V^N$ the action space, $G \subset V^N$ the goal space, $T : S \times A \mapsto S$ the transition function, $R : S \times G \mapsto S$ the goal-conditioned reward function, $Obs : S \mapsto V^N \equiv O$ the observation function that maps a state s to a textual description and γ the discount factor. For a trajectory $\tau = (s_1, a_1, \ldots, s_H, a_H)$ of length H, we note $R_g(\tau) = \sum_{t=1}^{H} \gamma^t R(o,g)$ its cumulative discounted reward given a goal $g \in G$. The optimal policy is $\pi^* = \arg\max_\pi \mathbb{E}_{g\in G, \tau\sim\pi(\tau|g)}[R_g(\tau)]$, with $\pi(\tau|g)$ the probability of τ following π.

On top of this framework, we introduce multiple prompt formulations $P = \{P_i\}_{i\in\{1,\cdots,n\}}$, where $P_i : O \times G \mapsto \mathcal{P} \subset V^N$ formats text entries from observations and goals as prompt inputs (see Figure 3.6). We assume that all formulations from P preserve information, i.e., any optimal policy π_i^* using the prompt formulation $P_i \in P$ can obtain the same amount of rewards as an optimal policy π^* acting on original observations and goals: $\forall g \in G, \forall i \in \{1, \cdots, n\}, \mathbb{E}_{\tau\sim\pi_i^*(\tau|g)}[R_g(\tau)] = \mathbb{E}_{\tau\sim\pi^*(\tau|g)}[R_g(\tau)]$.

Applying GLAM

First, we define a policy π, based on an LLM for solving tasks in M, given any prompt formulation $P_i \in P$. For any goal $g \in G$ and observation $o \in O$, we note $\mathbb{P}_{LLM}(w_k|p_i^{o,g}, w_{<k})$ the probability of token w_k generated by the prompt formulation P_i and the decoding history $w_{<k}$. We also define $\mathbb{P}_{LLM}(a|p_i^{o,g}) = \Pi_{k=0}^{|a|}\mathbb{P}_{LLM}(w_k|p_i^{o,g}, w_{<k})$ the decoding probability of the sequence corresponding to action $a \in A$ given prompt $p_i^{o,g}$. Following Tan et al. (2024), we use a normalized decoding probability $\mathbb{P}_{LLM}^{norm}(a|p_i^{o,g}) = \mathbb{P}_{LLM}(a|p_i^{o,g})^{\frac{1}{|a|}}$ to better balance actions of various sizes. From these quantities, we build the policy as $\pi(a|p_i^{o,g}) = \mathbb{P}_{LLM}^{norm}(a|p_i^{o,g})/Z_i^{o,g}$, where $Z_i^{o,g} = \sum_{a\in A}\mathbb{P}_{LLM}^{norm}(a|p_i^{o,g})$ is the partition function. Then, we follow GLAM to train our LLM agents using PPO (Schulman et al., 2017). Given a subset of prompt formulations $P_{ppo} \subseteq P$, we note $\pi_{P_{ppo}}$ an LLM agent of parameters θ trained by minimizing PPO's loss $PPO_{loss}(\theta, P_{ppo})$ from trajectories using prompt formulations P_{ppo}.

3.3.3 Contrastive learning

We propose an approach to dealing with multiple prompt formulations. We add a contrastive loss to bring closer the latent representations $z_\theta(p_i^{o,g})$ and $z_\theta(p_j^{o,g})$ of the same observation-goal pair (o, g) across prompts $p_i^{o,g}$ and $p_j^{o,g}$, compared to latent representations $z_\theta(p_i^{o',g'})$ of other observations and goals (o', g') of prompt $p_i^{o',g'}$. That is, we aim to minimize:

$$C^{(i,j)}(\theta) = \mathop{\mathbb{E}}_{\substack{(o,g)\sim d^{\pi_{P_{ppo}}} \\ (o',g')\sim d^{\pi_{P_{ppo}}}}} \max\big(\Delta(z_\theta(p_i^{o,g}), z_\theta(p_j^{o,g})) - \Delta(z_\theta(p_i^{o,g}), z_\theta(p_i^{o',g'})) + 1), 0\big) \tag{3.4}$$

where $z_\theta(p_i^{o,g})$ is the latent representation of textual prompt $p_i^{o,g}$ produced by prompt formulation P_i applied to the observation-goal pair (o, g), $\Delta(z, z') = ||(z) - (z')||_2^2$ is the Euclidean distance between two latent representations z and z', and $d^{\pi_{P_{ppo}}}$ is the joint stationary observation-goal distribution of a policy using P_{ppo}. In practice, we sample pairs (o, g) from the collected trajectories, supposed to follow $d^{\pi_{P_{ppo}}}$.

Our final loss L jointly optimizes PPO_{loss} with the contrastive loss $C^{(i,j)}(\theta)$ as follows:

$$\begin{aligned} L(\theta, P_{ppo}, P_c) &= PPO_{loss}(\theta, P_{ppo}) \\ &+ \frac{\alpha}{|P_c|^2}\sum_{P_i,P_j\in P_c^2} C^{(i,j)}(\theta) \end{aligned} \tag{3.5}$$

where α is a parameter regulating the impact of $C^{(i,j)}$, P_{ppo} represents the set of prompts used for fine-tuning the LLM policy with PPO's loss, and P_c represents the set of prompts used for the contrastive regularization $C^{(i,j)}$. Following Ni et al. (2022), $C^{(i,j)}$ is only applied to the representation of the first token of the prompt. Appendix B.1.4 provides implementation details about contrastive regularization.

3.3.4 Experimental Protocol

In this section, we introduce our experimental protocol. In particular, we detail our two environments, prompt templates, and training approaches.

Environments

We conduct our experiments in two textual environments: 1) BabyAI-Text, and 2) the Medium difficulty TWC Environment, proposed in Murugesan et al. (2021) (noted TWC-Medium). In TWC, the agent must solve household tasks using textual observation containing the description of the current room and a list of possible actions. These two environments assess complementary skills in terms of semantics: BabyAI-Text requires exploring and navigating between objects, whereas TWC-Medium requires the use of commonsense knowledge and reasoning. See Appendix B.1.1 for more details.

Prompt Design

We define four distinct prompt formulations that all contain the same information: the goal, the possible actions, the inventory, and textual observation. The first prompt formulation P_0 follows a format where pieces of information are separated by line breaks in the following order: possible actions, goal, observation, and inventory. P_1 is similar to P_0 but switches the order of the information. P_2 employs a more rigid syntax with delimiter tags, similar to an XML file. Finally, P_3 removes all rigidity in syntax and follows a natural language format, paraphrased by a prompt writer. Examples of P_0 and P_2 in TWC-Medium can be found in Figure 3.6, and detailed descriptions of all prompt formulations are provided in Appendix B.1.2.

Training

We consider several LLMs, including encoder-decoder architectures (Flan-T5 78M and 780M (Rae et al., 2022)) and decoder-only architectures (GPT-Neo 1.3B (Black et al., 2022)). We also propose partial results for Flan-T5 2.7B and Llama 7B (Touvron et al., 2023) in Appendix B.3.4. We consider three different fine-tuning scenarios denoted as $\sigma^{P_c}_{P_{ppo}}$: 1) the one prompt scenario σ_0 where the LLM is fine-tuned with PPO on prompt P_0 only, 2) the multiple prompt scenario $\sigma_{0:3}$ where the LLM is fine-tuned with PPO on prompts P_0 to P_3, and 3) the contrastive scenario $\sigma^{0:3}_0$ where the LLM is fine-tuned with PPO on P_0 only, but using our additional contrastive loss considering prompts P_0 to P_3. Fine-tuning is performed in both environments for 500k steps across 5 seeds. Importantly, each experiment below uses a single formulation for collecting trajectories, and potentially multiple formulations when computing the loss.

Evaluation

We evaluate the zero-shot scenario σ_{zs}, that corresponds to the pre-trained LLM agent without any fine-tuning, and the fine-tuned scenarios $\sigma^{P_c}_{P_{ppo}}$ on a set of test tasks.

Regardless of the scenario used, we evaluate the agent's performance on every single prompt formulation. In addition, for the scenarios $\sigma_{0:3}$ and $\sigma_0^{0:3}$, we define a new prompt formulation P_4 to analyze generalization to unseen prompts. P_4 follows a different template with changes to the sections' name, reordering, and an additional context tag (details about this prompt in Appendix B.1.2).

3.3.5 Task-solving abilities with respect to prompt formulation

We begin by training all our models in both environments and using the different scenarios. To evaluate the task-solving abilities, we define the success of an LLM agent in the environment as a case where a trajectory reaches the intended goal g. We deduce two metrics: 1) (**SR**): the success rate of the agent and 2) ($\overline{\textbf{SR}}$): the mean success rate across all prompt formulations and episodes.

In this section, all experiments use the formulation P_0 for collecting trajectories. We provide in Appendix B.3.4 the results when using other formulations. Figure 3.7 shows the SR values for Flan-T5 78M, 780M, and GPT-Neo 1.3B (also see Appendix B.3.4 for the remaining models) obtained according to the different evaluation scenarios: σ_{zs}, σ_0, and $\sigma_{0:3}$.

We start by discussing the results obtained with the same scenario as the one used in our section introducing GLAM (i.e., when the LLM is fine-tuned and evaluated using the same formulation). We observe that, regardless of the environment or model, our functional grounding significantly improves the agent's performance. For instance, in the TWC-Medium environment, both Flan-T5 78M and 780M achieve success rates exceeding 90% (compared to a maximum of 45% with σ_{zs}), and GPT-Neo 1.3B approaches a success rate of 85% (compared to at most 20% with σ_{zs}). However, a notable decline in performance is observed when using another prompt formulation at test time, with a decrease of over 30% from using the original P_0 to the P_3 variation in TWC-Medium and more than 30% from P_0 to P_2 in BabyAI-Text. *This drop outlines prompt overfitting in LLMs.* Importantly, *this trend is also observed on larger models (Flan-T5 2.7B and LLama 7B)* as detailed in Appendix B.3.4.

We also observe that, for both σ_{zs} and fine-tuned models, encoder-decoder models outperform decoder-only architectures, even when the decoder-only models are larger. One possible explanation for this is the superior representational abilities of encoder-only transformers, as shown by prior work discussed in Chapter 2. Regarding disparities between environments, we observe that the Flan-T5 78M model struggles in learning appropriate policies in BabyAI-Text.

We then compare the results of $\sigma_{0:3}$ and $\sigma_0^{0:3}$. In most cases, the $\overline{\textbf{SR}}$ of $\sigma_0^{0:3}$ surpasses the σ_{zs}, σ_0, and $\sigma_{0:3}$ (except for the 780M model on BabyAI-Text, where $\sigma_0^{0:3}$ underperforms by 3%). This is noticeable since $\sigma_0^{0:3}$ learns the matching between prompt formulations while $\sigma_{0:3}$ learns each prompt independently. In terms of homogeneity, both $\sigma_0^{0:3}$ and $\sigma_{0:3}$ scenarios yield consistent results (marked with * in Figure 3.7) for all TWC-medium evaluations. Flan-T5 78M struggles in BabyAI-Text with both task-solving and achieving homogeneity across prompts for $\sigma_0^{0:3}$ and $\sigma_{0:3}$. For GPT-Neo 1.3B on BabyAI-Text, $\sigma_0^{0:3}$ achieves homogeneity across prompts P_0 to P_2 but performance drops with P_3. Nevertheless, the $\overline{\textbf{SR}}$ for $\sigma_0^{0:3}$ remains higher than that of $\sigma_{0:3}$.

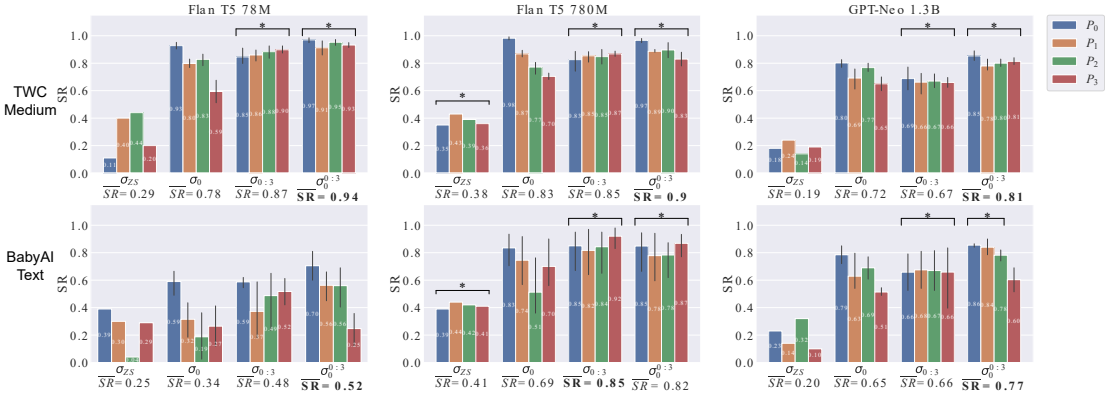


Figure 3.7: **Success Rate (SR)** in BabyAI-Text and TWC-Medium. The x-axis indicates the training scenario, while colors represent the prompt formulation used to format inputs during test episodes. The asterisk (*) indicates instances where the chi-squared test exceeds the homogeneity threshold (p-value > 0.05). Bold values for $\overline{\mathbf{SR}}$ represent the best result in the evaluation. Results show that the LLM exhibits heterogeneous performance when the prompt formulation used during training differs from the one used at evaluation, with a drop in success rate of up to 30% in certain scenarios.

	$\sigma_{0:3}$	$\sigma_0^{0:3}$
78 M	0.77 ± 0.11 (3%)	0.92 ± 0.02 (97%)
780 M	0.80 ± 0.06 (4.7%)	0.86 ± 0.05 (91%)
1.3 B	0.66 ± 0.02 (99%)	0.76 ± 0.03 (98%)

Table 3.2: **Success Rate (SR) in TWC-Medium** using prompt formulation P_4 unseen during training. The values in parentheses represent the χ^2 homogeneity test results, red indicates heterogeneity (p-value $< 5\%$) and green indicates homogeneity (p-value $> 5\%$). $\sigma_0^{0:3}$ scenario demonstrates superior performance in terms of SR and homogeneity compared to $\sigma_{0:3}$ across all models.

We also evaluate in Table 3.2 the generalization capabilities of $\sigma_0^{0:3}$ compared to $\sigma_{0:3}$ using the prompt formulation P_4, which was not seen during training in either scenario. The results indicate that the mean success rate on P_4 is higher for $\sigma_0^{0:3}$ than for $\sigma_{0:3}$, showing superior generalization capabilities across all three model sizes. To validate these findings, we conduct a χ^2 test to verify the homogeneity of P_4 results compared to the mean results obtained from the training prompts. The values in parentheses summarize the results, showing that, for all $\sigma_0^{0:3}$ models, the success rates on the unseen prompt P_4 follow the same distribution as the training success rates. This indicates robust generalization to unseen prompts, in contrast to $\sigma_{0:3}$, where the 78M and 780M models do not exhibit the same distribution of results as during training. This finding supports the conclusion that regularization helps mitigate overfitting.

3.3.6 Analyzing the functionally grounded latent representations

To further analyze prompt sensitivity observed in the previous section, we investigate the latent representation of states formatted with different prompt formulations on TWC-Medium. As in GLAM, we consider the representations outputted by the last layer for the token preceding the action (i.e., the last token of the prompt for decoder-only models

and the padding token for encoder-decoder models). This also corresponds to the latent representation given to our value head for PPO.

Measuring prompts similarity

We start by delineating the disparities between prompt formulations by computing the cosine similarity of their latent representations pre and post fine-tuning. Given a set of observation-goal pairs $\Gamma = \{(o, g)\}$, the similarity between representations using different inputs formatted by a single prompt formulation P_0 is defined as

$$Intra(P_i) = \frac{1}{|\Gamma|^2 - |\Gamma|} \sum_{\substack{(o,g)\in\Gamma \\ (o',g')\in\Gamma\setminus\{o,g\}}} cos\left(z_i^{o,g}, z_i^{o',g'}\right).$$

Besides, we assess the similarity between representations from different prompt formulations as:

$$Inter(P_i, P_j) = \frac{1}{|\Gamma|} \sum_{(o,s)\in\Gamma} cos\left(z_i^{o,g}, z_j^{o,g}\right).$$

We compare in Table 3.3 the intra-prompt and inter-prompt similarities to measure the topology of the latent representations of states according to our different scenarios, averaged over any pair of prompt formulations. The most striking result is that LLMs tend to cluster prompts formulated with the same prompt formulation ($Intra \simeq 1$), even when they correspond to different goal-observation pairs. In contrast, the same pair formulated with different prompt formulations exhibits low similarity ($Inter < 0.5$). The low similarity between prompts and the high similarity within prompts suggests that LLMs tend to focus on capturing syntax rather than semantics. Moreover, the last column of Table 3.3 shows a significant increase in the inter-prompt proximity for all models, confirming our results that applying the contrastive loss can help disregard the prompt formulations at the benefit of its content.

Visualizing prompts similarity

We visualize the topology of GPT-Neo's latent representation of our prompts using UMAP (McInnes et al., 2018) in Figure 3.8. This visualization illustrates prompt overfitting. In particular, we see no overlap between clusters of prompt formulations in σ_{zs}, nor σ_0. Importantly, using $\sigma_{0:3}$ does not mitigate this, highlighting the need for methods tackling overfitting in both task efficiency and state representation. However, results from $\sigma_0^{0:3}$ indicate that the topology of the latent space changed. The latent vectors are no longer clustered by formulation and now overlap, suggesting that the model has learned to match similar states together. Further details regarding distances and visualizations can be found in Appendix B.3.6.

Measuring prompt information use

We then go further and analyze which parts of the prompts (goal, possible actions, observation, and inventory) are used by Flan-T5 78M when selecting the action. We study the relationship between inputs and prediction actions using the Integrated Gradients

Models		σ_{zs}	σ_0	$\sigma_{0:3}$	$\sigma_0^{0:3}$
$78M$	*Intra*	0.992 ± 0.003	0.991 ± 0.003	0.991 ± 0.003	0.907 ± 0.017
	Inter	0.376 ± 0.019	0.382 ± 0.020	0.371 ± 0.020	0.806 ± 0.029
$780M$	*Intra*	0.998 ± 0.001	0.997 ± 0.001	0.998 ± 0.001	0.939 ± 0.017
	Inter	0.469 ± 0.462	0.458 ± 0.449	0.47 ± 0.461	0.812 ± 0.06
$1.3B$	*Intra*	0.995 ± 0.001	0.994 ± 0.001	0.997 ± 0.001	0.994 ± 0.017
	Inter	0.552 ± 0.03	0.539 ± 0.01	0.501 ± 0.05	0.94 ± 0.009

Table 3.3: **Inter and intra-similarity** comparison for Flan-T5 78M, 780M, and GPT-Neo 1.3B models on TWC-Medium on our different scenarios. For all models, we observe that *Intra* approaches 1, while *Inter* is consistently below 0.5. This indicates that the LLMs tend to cluster prompts based on their formulation rather than on content. A similar trend is observed for both $\sigma_{0:3}$ and σ_0 scenarios.

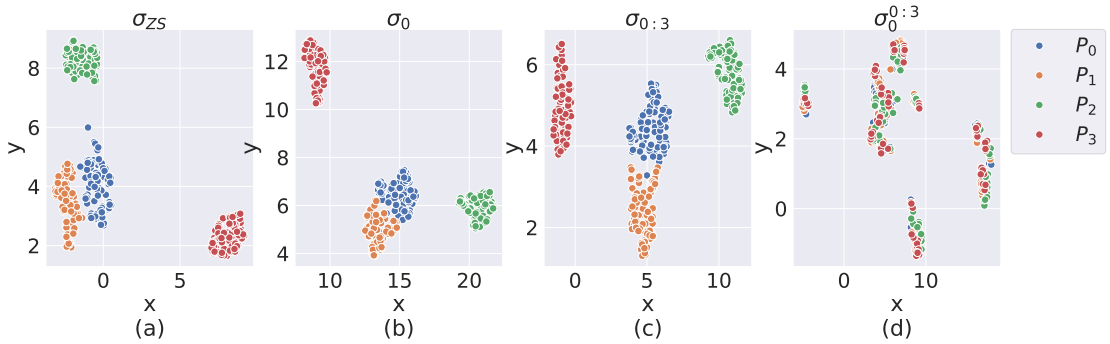


Figure 3.8: **UMAP visualization of prompts' hidden representation in GPT-Neo 1.3B across 100 states of TWC-Medium using four prompt formulations.** This illustrates a clustering based on prompt formulation over semantics in both σ_{zs} and fine-tuned models with σ_0 and $\sigma_{0:3}$. Additional results for other prompts used to collect trajectories during training can be found in Appendix B.3.7.

algorithm (Madsen et al., 2022) to generate saliency scores for each element of the input. For each scenario σ, and each prompt formulation P_i, we compute the averaged saliency of prompt tokens for every observation-goal pair (o, g) from Γ. Saliency scores are computed using the `Inseq` toolkit (Sarti et al., 2023), as the Integrated Gradient of the output probability of the policy $\pi(a^*|p_i^{o,g})$ with respect to the tokens of the prompt $p_i^{o,g}$, with a^* the most likely action regarding the observation-goal pair o, g. To enhance interpretability, saliency scores for all prompt tokens are finally aggregated depending on the part of the prompt they belong to, among {possible actions, goal, observation, inventory}. We do so after filtering out the top 5% most significant tokens from each section to eliminate perturbations caused by irrelevant tokens.

Figure 3.9 shows the saliency scores of prompt parts. We can observe that the parts used vary between prompt formulations and scenarios. For instance, in σ_{zs}, our LLM prioritizes the inventory when P_1 is used, whereas switching to P_0 and P_2 shifts the focus to the possible Actions. This can be explained by the fact that P_0 and P_2 follow the same order of information (see Appendix B.1.2), and therefore the LLM finds the possible actions in the same position. Fine-tuning our LLM alters the saliency scores compared to σ_{zs}, with a strategy prioritizing the goal over possible actions when using P_0. However, the LLM still focuses on different parts when changing the prompt formulation. Using $\sigma_{0:3}$ also results in heterogeneous saliency maps across prompt formulations, showing that our LLM continues to process the prompts differently despite being fine-tuned with multiple prompt formulations. However, $\sigma_0^{0:3}$ leads again to higher homogeneity, strengthening our results on the superior representational power of the contrastive regularization. Additional results can be found in Appendix B.3.7.

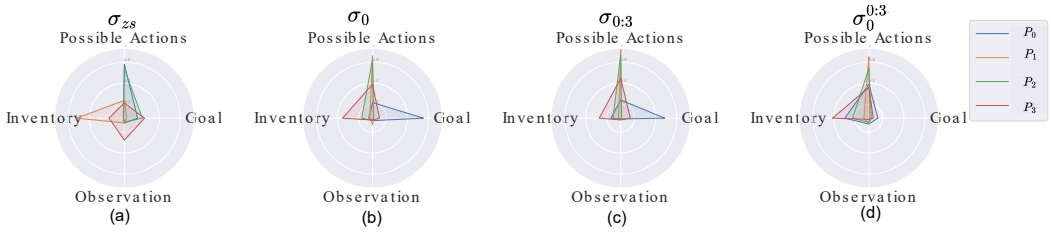


Figure 3.9: **Saliency maps of Flan-T5 78M across scenarios on TWC-Medium**. The maps show that the LLM focuses on different sections of the prompt depending on the prompt formulation. This variation is linked with performance changes when the prompt formulation is altered. This phenomenon is observed in σ_{zs}, σ_0, and $\sigma_{0:3}$ scenarios.

3.3.7 Environmental knowledge acquired through functional grounding

We end our analysis by assessing the environmental knowledge acquired by GPT-Neo 1.3B by measuring its accuracy in question-answering (QA) related to TWC-Medium, which includes object counting (TWC OC) and task-related questions (TWC TR). We follow the methodology from Xiang et al. (2023) for constructing a set of questions based on the optimal successful trajectory of TWC-Medium. First, we create task-related questions (TWC TR) by generating multiple-choice questions asking which object is the most useful for solving a task. Then, we generate object counting questions (TWC OC) in which we present to the LLM a trajectory in the environment and prompt it to count the number of objects in a specific location.

We measure the accuracy of GPT-Neo 1.3B in TWC TR and TWC OC datasets before and after fine-tuning over all training scenarios in Table 3.4. Before fine-tuning, our LLM generally struggles to answer environment-related questions, exhibiting poor performance across both QA datasets. Following fine-tuning with σ_0 or $\sigma_{0:3}$, the accuracy shows only superficial improvement for the TWC TR, and minor enhancements are observed in the TWC OC compared to the σ_{zs} setting. This indicates that fine-tuning in these scenarios leads to superficial updates. However, fine-tuning with the $\sigma_0^{0:3}$ scenario results in a more substantial improvement compared to other scenarios, with at least a 13% increase in

	TWC TR	**TWC OC**
σ_{zs}	0.4866 *	0.0876 ***
σ_0	0.4901 *	0.1340 ***
$\sigma_{0:3}$	0.5019 *	0.2526 *
$\sigma_0^{0:3}$	**0.6322**	**0.5155**

Table 3.4: **Environmental knowledge** of GPT-Neo 1.3B on TWC TR and TWC OC datasets. We plot the success rate over questions, with * and *** corresponding to the p-value (resp. < 0.05 and < 0.001) of Welch's t-test to compare the performance between $\sigma_0^{0:3}$ and other scenarios. We observe a significant improvement with $\sigma_0^{0:3}$ scenario compared to σ_{zs}, σ_0, and $\sigma_{0:3}$ scenarios across both datasets.

TWC TR accuracy, and respective gains of 25%, 35%, and 43% over the $\sigma_{0:3}$, σ_0, and σ_{zs} scenarios. These findings highlight that $\sigma_0^{0:3}$ not only enhances robustness to prompt variations but also improves the LLM's understanding of the environment's dynamics.

3.3.8 Conclusion

In this work, we proposed an in-depth study of how functional grounding, through GLAM, affects the internal representations and knowledge in LLMs. While Section 3.2 focused on the general task-solving performance (that is, the functional competence), this contribution focused on shedding light on how functionally grounded LLMs make decisions based on the observation they access through a prompt, as well as how functional grounding helps the LLM acquire knowledge about the environment's dynamics. For this, we introduced an experimental protocol along with a large-scale study over varying LLMs, environments, and prompts. Our results showed that fine-tuning LLMs with GLAM using a single prompt formulation leads to strong *prompt overfitting*, resulting in high prompt sensitivity for functional competence, prompt-specific information seeking in the observation, and little environment knowledge. We also showed that fine-tuning simultaneously on multiple prompt templates still results in prompt-specific strategies. Finally, our results indicate that our contrastive learning regularization is the only approach leading to robust strategies that rely on the same information from the observation, regardless of the prompt formulation, and increased abilities at answering questions about the environment's dynamics.

Overall, our study contributed to a better understanding of the impact of functional grounding of LLMs, from functional competence and beyond. Interestingly, our results show that functional grounding alone helps LLMs capture environments' dynamics and leverage this knowledge to answer questions. Nonetheless, we believe many questions remain open regarding how GLAM influences other core abilities of an LLM. For example, does fine-tuning an LLM to control an environment enhance its capacity to predict or explain that environment's dynamics explicitly? Conversely, does grounding in a specific environment constrain the model's plasticity or hinder its ability to acquire new skills in different settings?

3.4 Discussion

In this chapter, we introduced the notion of *functional grounding* and highlighted the role of embodied and active experience in developing functional competence, that is, the capacity to use language to achieve goals. In Section 3.2, we presented the first empirical contribution of this research: the GLAM approach, which uses online reinforcement learning to functionally ground LLMs. Our results showed that GLAM significantly enhances the task-solving abilities of LLMs in interactive environments and produces functional competence that generalizes to new tasks and unseen objects. Critically, we demonstrated that active learning via online interactions is essential for functional grounding: it outperforms passive imitation learning (behavioral cloning), even when the latter is trained on trajectories from an optimal policy.

We further investigated the internal impact of functional grounding in Section 3.3, focusing on the representational effects of GLAM across LLM architectures, environments, and prompt variations. Our analysis revealed a limitation: prompt overfitting, where functional grounding led to shallow adaptations tied to the specific language formulations seen during training. This limitation echoed earlier findings from Section 3.2, where LLMs trained on a single language failed to generalize to others. We found that grounded LLMs tended to rely on prompt-specific cues and lacked a deeper understanding of the environment's dynamics—struggling, for example, with answering comprehension questions about the environment. To address this, we introduced a contrastive learning regularization that encourages the LLM to learn representations invariant to prompt formulation. This significantly improved the robustness of the grounded competence across prompt variants and enhanced the model's ability to answer environment-related questions. Overall, this chapter demonstrated that online RL paired with interactive environments is a promising path for imbuing LLMs with functional competence—but also that the way in which language is grounded matters significantly.

Despite these advances, several limitations remain. First, we restricted our study to purely textual settings, where both observations and actions are expressed in natural language. While this allowed us to isolate and study functional grounding in controlled environments, it limits applicability to broader, multimodal scenarios. Extending functional grounding to other modalities—particularly vision—is a natural and necessary next step. Recent efforts have begun addressing this by applying GLAM-style fine-tuning to VLMs. For instance, Zhai et al. (2025) combined BC and PPO to train VLMs in visual environments, though they reported limited gains in functional competence. In contrast, Aissi et al. (2025) proposed a hybrid setup where a fine-tuned VLM provides textual descriptions of visual scenes to a language model grounded using GLAM. This led to stronger results but required supervised fine-tuning of the VLM to produce informative descriptions. Much work remains to fully integrate multimodal understanding and functional grounding in VLMs.

Another limitation stems from our observation that GLAM may induce only superficial changes in LLMs' internal representations—particularly when grounded using limited linguistic diversity. While our contrastive learning approach helped mitigate prompt overfitting, it still required manually designed prompt variations from the environment. An alternative and complementary strategy would be to ground similar forms of functional competence across diverse environments. In humans, functional skills are deployed across

varied contexts, which helps reinforce general strategies. This environmental diversity could also support broader generalization in action spaces—for example, enabling LLMs to use synonymous actions. As seen in Section 3.2, training with a fixed action space in a single environment offered no such generalization.

The next chapters build on these insights to further develop functional grounding through online RL and interactive environments. Chapter 4 will explore the predictive side of functional competence, in contrast to the control-focused perspective of this chapter. We will discuss how humans rely on intuitive theories to predict their world and propose a curiosity-driven RL framework for enabling LLMs to generate abstract, language-based models of environmental dynamics. Chapter 5 will then examine how to scale functional grounding to complex, open-ended environments. We will draw inspiration from human skill acquisition through autotelic learning and present methods that incorporate teacher-student scaffolding and metacognitive monitoring—two mechanisms critical to designing autotelic LLM agents with lifelong learning in rich environments.

Chapter 4

Moving beyond control: World modeling

Contents

4.1 World models 70
4.1.1 World models in RL agents 70
4.1.2 Mental models with theories 72
4.2 WorldLLM: Grounding LLMs' world modeling using curiosity-driven theory making 73
4.2.1 Improving LLMs' world model with WorldLLM 74
4.2.2 Experiments 78
4.2.3 Related work 84
4.3 Discussion 85

The previous chapter introduced the concept of functional grounding and presented GLAM, a method based on reinforcement learning for functional grounding of LLMs. Our analysis in Section 3.3 further explored how acquiring functional competence through interactions and RL influences the LLMs' internal representations of their environment. The results suggest that GLAM enables LLMs to capture some aspects of environmental dynamics, albeit in a limited way. When we introduced functional grounding in Section 3.1, we framed functional competence as the ability not only to control but also to model and *predict* external dynamics through the use of symbols. This stems from the close connection between action and modeling: making decisions requires anticipating how actions will affect the world. In this chapter, we shift focus to how LLMs might explicitly acquire world modeling abilities through functional grounding. Drawing inspiration from how humans—particularly children—progressively refine internal models of the world, we propose a framework in which an LLM autonomously explores an environment and generates natural language hypotheses describing its dynamics. By combining Bayesian inference to refine hypotheses with intrinsically motivated RL to guide evidence collection, we argue that this framework enables continual and autonomous improvement of the LLM's world modeling abilities. Before detailing this framework, we begin by examining the role and nature of world models.

Collaborations and scientific output – This chapter presents a contribution that originated as a follow-up project to the work on GLAM. The project was structured as the

Master's internship of Guillaume Levy (over the summer 2024) co-supervised by Thomas Carta and myself, with the objective of investigating how online interactions with an environment could improve the world modeling abilities of LLMs. While Guillaume was primarily responsible for implementing and running the experiments, all major decisions throughout the project were made jointly by the three of us. We were later joined by Cédric Colas, whose input significantly shaped the project—particularly by encouraging us to explore theory-based world models. Guillaume and I co-led the writing of a short version of the paper, which was presented at the RLDM conference: *Guillaume Levy, Cédric Colas, Pierre-Yves Oudeyer, Thomas Carta, Clément Romac. 2025. WorldLLM: Improving LLMs' world modeling using curiosity-driven theory-making. Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*. I also took the lead in writing a longer preprint version of the work, which is presented in Section 4.2 of this manuscript.

4.1 World models

The question of how humans create mental models of their world is vast and has long been a central concern in psychology (Craik, 1943; Johnson-Laird, 1983). Among the many facets explored, considerable attention has been devoted to understanding how children form mental models of objects in their environment—particularly through the acquisition of concepts (Piaget, 1954; Gelman & Legare, 2011). Parallel efforts have investigated how humans model other humans, especially through the development of Theory of Mind—an essential component of functional competence in social settings (Wimmer & Perner, 1983; Gopnik & Wellman, 1992). A further key focus has been on how the construction of such world models influences a child's exploration behavior (Piaget, 1954). In essence, children engage with their environment to gather evidence in support of—or in contrast to—their current internal model. These insights have naturally inspired mechanisms aimed at equipping artificial agents with internal world models that guide and structure their interactions with the environment.

4.1.1 World models in RL agents

> *"One way of understanding the predictive model inside of our brains is that it might not be about just predicting the future in general, but predicting future sensory data given our current motor actions."* – Ha & Schmidhuber (2018)

Building world models in artificial agents has long been a central research focus (Ha & Schmidhuber, 2018; Ding et al., 2025). In RL, *model-based* approaches have been studying how to learn components of the MDP, such as the reward and transition functions (Moerland et al., 2023). This contrasts with the *model-free* RL approaches primarily discussed throughout this manuscript, where agents do not construct an explicit model of the environment but are assumed to capture its dynamics implicitly through policy optimization. Much of the existing literature centers on learning neural models of the transition function, called *forward models*, that predict the next observation o_{t+1} the agent

will encounter after taking an action a given the current observation o_t. Such forward models can significantly simplify the search for an optimal policy, for example by enabling multi-step planning (Silver et al., 2016; Schrittwieser et al., 2020; Hansen et al., 2022), or by allowing agents to train entirely on imagined transitions, i.e., simulated interactions with the learned model rather than the real environment (Hafner et al., 2020, 2021, 2025). See Figure 4.1 for a schematic overview of these classic interactions between the policy, environment, and model in model-based RL.

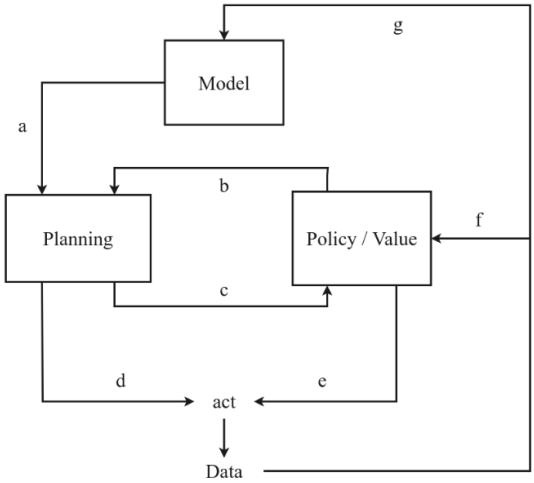


Figure 4.1: Overview of the classic interactions between a model and a policy in model-based RL from Moerland et al. (2023). While classic model-free RL only considers the direct interactions between the policy and the environment (the loop e-f), model-based RL introduces interactions with a model learned from environmental data (g). The interactions between the policy and the model (the loop b-c) can be used for multi-step planning before choosing an action (d) or for policy improvement through imagined transitions (c).

Moreover, evidence from developmental psychology indicating that children's exploration is driven by the acquisition of internal world models has inspired the design of various intrinsic motivation signals for RL agents. As discussed in Chapter 2, several approaches propose rewarding transitions that are poorly predicted by a forward model trained on data collected from the agent's policy. For instance, Schmidhuber (1991a); Pathak et al. (2017) introduced intrinsic rewards based on high prediction errors, while Schmidhuber (1991a); Oudeyer & Kaplan (2007); Lopes et al. (2012); Houthooft et al. (2016) instead rewarded transitions that led to improvement in the forward model itself. However, learning low-level, high-fidelity forward models through gradient descent does not fully reflect the nature of human world modeling as described in psychological research. In particular, human world models appear to be: 1) abstract and inherently imprecise; 2) susceptible to revision based on only a few critical pieces of counter-evidence; and 3) focused on more than just short-term outcome prediction—for example, they capture causal relationships and influence how information is interpreted and prioritized (Johnson-Laird, 1983; Tenenbaum et al., 2011).

4.1.2 Mental models with theories

Several theories have been proposed in psychology to formalize how humans represent, encode, and update their internal world models. Among these, the *theory theory* has gained significant attention. This framework posits that children—and adults (Gopnik, 1996)—rely on intuitive theories to both predict and explain their experiences (Johnson-Laird, 1983; Gopnik & Meltzoff, 1997; Gelman & Legare, 2011). Gopnik & Meltzoff (1997) argue that cognitive development in children can be viewed as a process of iterative theory refinement, shaped by the accumulation of experiential evidence throughout childhood. This perspective offers a compelling explanation for the sharp developmental transitions observed at specific ages (e.g., the sudden emergence of object permanence) (Piaget, 1954).

According to Gopnik & Meltzoff (1997), theories exhibit several key properties: they are tied to evidence, abstract in nature, causal—explaining relationships between events—and dynamic, evolving as new evidence is encountered. A critical strength of such theories lies in their capacity to support both predictive and counterfactual reasoning, allowing individuals not only to anticipate outcomes but also to reason about alternative possibilities. In essence, they reflect internal beliefs about the causal structure of the world (see Figure 4.2). Similar to what has been discussed for cognitive development by the embodied movement (see Chapter 2), Gopnik & Meltzoff (1997) further emphasize the deep interrelation between theory development and language acquisition. They suggest that, as children interact with caregivers and peers, language facilitates the communication and negotiation of shared theories, with semantic development progressing in tandem with theory refinement. Intuitive theories can then be communicated, ranging from simple utterances (e.g., "gravity makes the apple fall") to more formalized versions (e.g., through mathematical models).

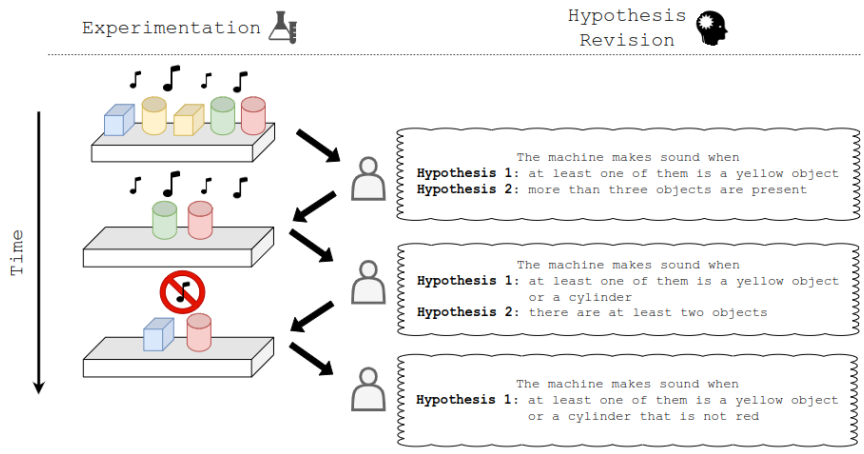


Figure 4.2: Example of the process of active intuitive theories refinement through interactions from Piriyakulkij et al. (2024).

These psychological insights have inspired various computational models of theory-based cognitive development and world modeling. A particularly influential line of work has modeled theory refinement as a (constrained) stochastic search process using Bayesian inference (Goodman et al., 2008; Ullman et al., 2010; Ullman & Tenenbaum, 2020). This

framework captures how theories can be generated and iteratively updated to increase the likelihood of observed evidence, guided by a prior over possible theories. Such priors can encode structural constraints or even innate biases, echoing the notion of inborn capacities discussed by Gopnik & Meltzoff (1997). More recently, several studies have explored how Bayesian inference can be leveraged to uncover an environment's internal dynamics by representing theories as executable computer programs (Tsividis et al., 2021; Tang et al., 2024; Wang et al., 2024b; Li et al., 2024; Piriyakulkij et al., 2024). These approaches demonstrate that LLMs are particularly well-suited to generating candidate theories—either directly as programs or as natural language hypotheses subsequently converted into programs—based on evidence presented in the prompt. In such settings, the LLM acts as a proposal distribution, generating hypotheses that are then evaluated and refined within a Bayesian inference loop.

We argue that, inspired by how humans explore their environment to refine their mental models, the world modeling abilities of LLMs should be functionally grounded through autonomous exploration. Specifically, we propose allowing an LLM to freely interact with its environment in order to improve its understanding of the environment's dynamics. To guide this process, we draw from the model-based RL literature, which has shown that rewarding exploration based on errors made by a forward model leads to efficient data collection. In our approach, the LLM itself serves as the forward model: a policy is trained to explore the environment in ways that expose the LLM's predictive failures. The resulting transitions—where the LLM's predictions are inaccurate—serve as learning signals. From this evidence, we build on theory-based world modeling approaches by iteratively generating and refining natural language hypotheses about the environment's dynamics. These hypotheses condition the LLM when acting as a forward model. We argue that LLMs' natural capacity to integrate prompt information makes them especially well-suited for leveraging such natural language theories. Intuitively, our framework aims to generate theories that help the LLM better predict and reason about its environment.

4.2 WorldLLM: Grounding LLMs' world modeling using curiosity-driven theory making

LLMs possess broad knowledge about the world, but leveraging this knowledge for precise dynamics modeling remains challenging (Vafa et al., 2024). In other words, they lack grounding of their world modeling abilities, and, in particular, functional grounding, which we argued involves grounding the symbols' processing to model and predict external dynamics to solve goals. Recent work has shown promise in using LLMs as forward models by fine-tuning them on embodied experiences (Xiang et al., 2023). However, such an approach is computationally costly given the large amount of interaction data required as well as the cost of gradient-based fine-tuning of LLMs. Contrary to this, LLMs are known to possess general knowledge that can be adapted using careful prompting (Zhang et al., 2024a).

In the previous section, we discussed theory-based world modeling methods. Current approaches either generate programs to capture environment dynamics directly (Tsividis et al., 2021; Tang et al., 2024; Li et al., 2024) or use natural language rules as an intermediate representation (Piriyakulkij et al., 2024; Wang et al., 2024b). Leveraging

the world model's structure as well as human-provided prior distributions over possible world models, theory-based methods are usually more sample-efficient than gradient-based ones. More importantly, the resulting world model often shows stronger generalization abilities. However, contrary to gradient-based approaches that usually do not require any expert-given knowledge, theory-based methods mostly rely on hand-crafted theory spaces. Finally, one major challenge in learning a world model lies in the data collection. Indeed, collecting transitions that cover the space of possible outcomes in the environment is key for learning an accurate world model. Notably, both classic model-based RL approach and theory-based methods agree on the importance of active data collection focusing on evidence with low likelihood under the current model (Schmidhuber, 1991a; Pathak et al., 2019; Burda et al., 2019b; Piriyakulkij et al., 2024).

In this work, we present **WorldLLM**, a framework for autonomous improvement of an LLM's world modeling abilities. Our approach combines 1) probabilistic theory induction to produce hypotheses that are given in our LLM's prompt to improve its predictions and 2) curiosity-driven RL to explore the environment and collect transitions poorly predicted by the current hypotheses. Formally, our LLM's world model is the conditional probability $P(s_{t+1}|s_t, a_t, H)$, where s_t represents a state, a_t an action, and H a set of natural language hypothesized theories. This probability is computed by the LLM by giving it s_t, a_t, and H in its prompt and taking the probability of s_{t+1} to follow this prompt. Our key insight is that natural language theories can help ground an LLM's broad knowledge into precise predictive power by providing domain-specific rules (Zhang et al., 2024a). Our approach consists of three interacting components: (1) our LLM that computes $P(s_{t+1}|s_t, a_t, H)$ by conditioning its predictions on both a state-action pair and the current hypotheses, (2) a theory generator that updates natural language hypotheses using Bayesian inference, and (3) a curiosity-driven reinforcement learning agent trained to collect evidence against the current hypotheses. Inspired by how humans, from children to scientists (Piaget, 1954; Gopnik & Meltzoff, 1997), actively update their internal world model by performing experiments, our agent's exploration provides new evidence for hypothesis refinement, creating a virtuous cycle of improvement.

We demonstrate our approach in a video game environment where agents manipulate and combine objects, showing that WorldLLM successfully learns accurate predictive models while generating human-interpretable theories about environment dynamics. This work contributes to a growing body of research on improving LLMs' world modeling capabilities and grounding their knowledge in specific domains. By combining ideas from theory-based RL, Bayesian inference, and active exploration, we provide a framework for learning structured, interpretable world models that leverage both the broad knowledge of LLMs and domain-specific experiences without any costly gradient-based learning.

4.2.1 Improving LLMs' world model with WorldLLM

Problem statement

We consider an environment framed as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$ with $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ the reward function, and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ the transition function. In particular, as our main objective is to

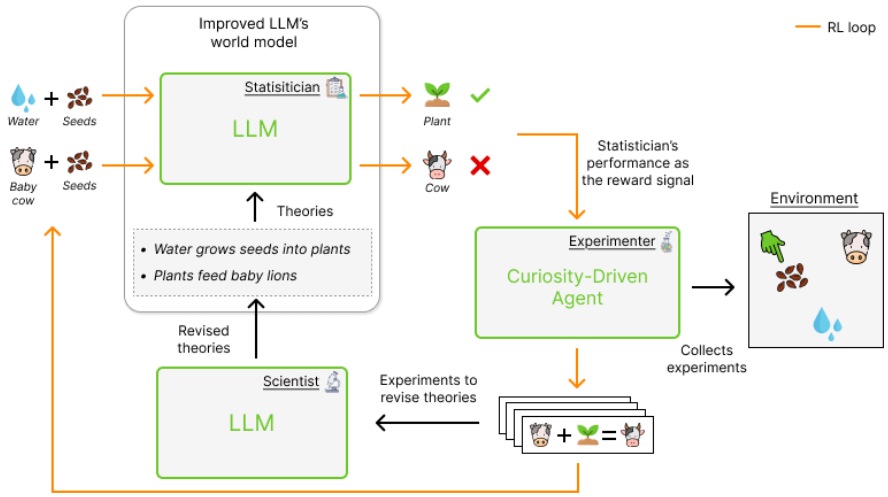


Figure 4.3: Our framework aims at improving an LLM's ability to predict the outcome of state-action pairs by augmenting it with natural language hypotheses about the world (*Statistician*). WorldLLM alternates between generating hypotheses (*Scientist*) and collecting evidence to update these hypotheses (*Experimenter*). The former uses Bayesian inference with an LLM as the proposal distribution. The latter uses curiosity-driven RL to train an agent to collect experiences that have a low likelihood under the *Statistician* with the current set of hypotheses.

study world modeling abilities, we consider reward-less environments where no particular task must be solved, i.e., the reward function always returns 0. We focus on textual environments where, given a vocabulary $\mathcal{V}$ and a maximum sentence length N, observations $o \in \mathcal{S} \subset \mathcal{V}^N$ and actions $a \in \mathcal{A} \subset \mathcal{V}^N$ are represented in natural language.

In such a setting, we study how to rapidly learn an LLM-based parametric model $\mathbb{P}^{LLM}$ of the transition function $\mathcal{T}$ (also called forward model) given a budget of experiences T allowed to an agent. This parametric model outputs a probability distribution over all possible next states given a state-action pair. The natural approach from the RL literature to improve this forward model uses gradient-based updates (Ha & Schmidhuber, 2018; Hafner et al., 2020; Schrittwieser et al., 2020). However, the scale of modern LLMs makes such an approach particularly costly. LLMs' natural in-context learning abilities open up an alternative approach: providing natural language hypotheses in the prompt to improve their predictions. This method is less costly, known as more sample efficient (Le Scao & Rush, 2021), and provides human-interpretable updates. Consequently, the parameter we seek to optimize is a set of hypotheses $\mathcal{H} \subseteq \mathcal{V}^N$ that, when given as input to the forward model (in addition to state-action pairs), maximizes the likelihood of the experienced next states.

We define the global optimization objective as finding the smallest set of hypotheses $H \subseteq \mathcal{H}$ that maximizes the forward model's probability of all possible transitions in the environment:

$$\begin{aligned} max \quad & \int_{S \times A \times S} \mathbb{P}^{LLM}(s'|s,a,H)\, ds'\, da\, ds \\ min \quad & |H| \\ \text{subject to} \quad & H \in \mathcal{H} \end{aligned} \tag{4.1}$$

As the full transition space is unknown to our agent, it can only use the collected transitions within the budget of T experiences. Note that it is essential for the agent to efficiently explore the environment within these T experiences, as collecting only similar transitions would lead to hypotheses that predict these transitions well but not the remainder of the transition space. For evaluation, as computing a forward model's performance on the whole transition space is usually intractable, we define a hidden test set $\mathcal{D}_{test}$ containing representative transitions.

WorldLLM

In this section, we present WorldLLM, our methodology for addressing the challenge outlined in the preceding section. Our framework comprises three interconnected components: the Statistician, the Scientist, and the Experimenter. The interactions among these components are illustrated in Figure 4.3. The Statistician represents our LLM-based forward model (i.e., $\mathbb{P}^{LLM}$) and is used to evaluate the current hypotheses. The set of hypotheses H the Statistician uses is produced by the Scientist, another LLM. These hypotheses are derived from trajectories collected from interactions with the environment using the Experimenter. WorldLLM alternates from during T iterations between evidence collection from the Experimenter and hypotheses refinement from the Scientist. In the subsequent sections, we elaborate on how these modules contribute to our objective's optimization.

Statistician – We first detail the Statistician module, whose role is to evaluate a set of hypotheses H_t on collected transitions. We start by renaming, for the sake of clarity, $\mathbb{P}^{LLM}$ by $\mathbb{P}^{St}$. As described above, the Statistician is a forward model computing the likelihood $\mathbb{P}^{St}(s'|s, a, H_t)$ of the next state s' given a state-action pair (s, a) and a set of hypotheses H_t. This likelihood is obtained by the LLM by giving it s, a, and H_t in the prompt and taking the probability of s' to follow this prompt.

Scientist – To improve the Statistician's prediction as a forward model, the Scientist's role is to search through the space of possible sets of hypotheses. To efficiently explore this space, we take inspiration from (Wang et al., 2024b; Piriyakulkij et al., 2024) and use a Bayesian inference framework. In particular, we rely on another LLM $\mathbb{P}^{Sc}$ as the proposal distribution which generates a candidate set of hypotheses following the distribution $\mathbb{P}^{Sc}(\hat{H}_t^i|\mathcal{D}_t, \hat{H}_t)$ given a data set of collected transitions $\mathcal{D}_t$ and the current best hypotheses candidate $\hat{H}_t$. We assume a uniform prior (i.e., no particular prior) over the space of hypotheses. For this paper, we chose the Metropolis algorithm, a variant of the Metropolis-Hastings algorithm, which iteratively generates candidate hypotheses for n_steps. The full algorithm is displayed in Algorithm 1 and more details are provided in Appendix C.2.

Experimenter – Given a set of hypotheses produced by the Scientist, the Experimenter must collect transitions that will help assess and refine the hypotheses (i.e., transitions that have a low likelihood under the Statistician). We propose to explore two types of approaches. First, we design oracle agents, allowing us to disentangle the efficiency of data collection and hypotheses refinement in WorldLLM.

Algorithm 1 WorldLLM

Input: T total number of iterations, n_{steps} number of Metropolis steps at each iteration, $\mathbb{P}^{St}$ the Statistician's distribution, $\mathbb{P}^{Sc}$ the Scientist's proposal distribution, π the Experimenter
Initialize $H_t \leftarrow null$
for $t = 0$ to T **do**
  Collect trajectories $\mathcal{D}_t \leftarrow$env(π)
  Evaluate current hypotheses $current\_score = \sum_{(s,a,s')\sim\mathcal{D}_t} \log \mathbb{P}^{St}(s'|s,a,H_t)$
  Set $\hat{H}_t = H_t$
  for $i = 0$ to n_{steps} **do** ▷ Metropolis step
    Generate hypotheses $\hat{H}_t^i \sim \mathbb{P}^{Sc}(\cdot|\mathcal{D}_t, \hat{H}_t)$
    Evaluate candidate hypotheses $score = \sum_{(s,a,s')\sim\mathcal{D}_t} \log \mathbb{P}^{St}(s'|s,a,\hat{H}_t^i)$
    Generate a random number $u \sim \mathcal{U}(0,1)$
    if $\log(u) < score - current\_score$ **then** ▷ Accept or reject
      Update hypothesis $\hat{H}_t = \hat{H}_t^i$
      Update score $current\_score = score$
  Set $H_t = \hat{H}_t$
  Update Experimenter π

For the oracles, we propose the following hard-coded policies:

- A policy that collects the same distribution of transitions as the test set's distribution (O-Ideal)
- An optimal policy that progressively moves from collecting simple transitions to collecting the hardest ones (O-Curriculum)
- An optimal policy that produces the sequence of actions necessary to collect the hardest transitions (O-Hardest)

While O-Hardest will collect complex transitions, it may not cover the full transition space. In comparison, O-Ideal matches the test set distribution. However, the iterative process of WorldLLM may require the Experimenter to focus for several iterations on particular subspaces of the transition space for the Scientist to produce accurate hypotheses. While O-Hardest and O-Ideal lack such a developmental trajectory, O-Curriculum focuses on specific transitions for an arbitrary number of collections. We also study the performance of a policy acting randomly (O-Random), which is unlikely to collect complex transitions.

Then, we introduce RL-based Experimenters. We propose to study three curiosity-based intrinsic rewards derived from $\mathbb{P}^{St}$ (Oudeyer & Kaplan, 2007). First, we introduce *RL-LogP*, where the Experimenter uses the prediction error (Schmidhuber, 1991a; Pathak et al., 2017; Burda et al., 2019a) as reward by receiving $r = -log\mathbb{P}^{St}(s'|s,a,H_t)$ for each collected transition (s,a,s'). While usually efficient, this method is also known to suffer in stochastic environments where it fails to separate epistemic from aleatoric uncertainty (Burda et al., 2019b). Second, we propose *RL-ALP*, which leverages information gain over the prediction error (a form of Learning Progress) (Oudeyer & Kaplan, 2007; Lopes

et al., 2012; Schmidhuber, 1991a) and rewards transitions on which the forward model improves. In particular, we use the absolute value of Learning Progress, which tracks both progress and forgetting in the forward model (Oudeyer & Kaplan, 2007; Baranes & Oudeyer, 2009):

$$r_{alp} = |\log \mathbb{P}^{St}(s'|s, a, H_{t-1}) - \log \mathbb{P}^{St}(s'|s, a, H_t)| \tag{4.2}$$

Finally, directly tracking Learning Progress at the transition level often leads to noisy estimations. We also propose *RL-ALPEXP*, a variant of RL-ALP which partitions the transition space into subspaces and computes ALP over each subspace, stabilizing the estimations. While existing works studied how to automatically infer such partitions (Baranes & Oudeyer, 2013), we here rely on hand-designed ones, which correspond to the partitions used by oracles. Further details are provided in Appendix C.1.

4.2.2 Experiments

Experimental setup

Environment – To evaluate WorldLLM's efficiency, we use the Playground-Text

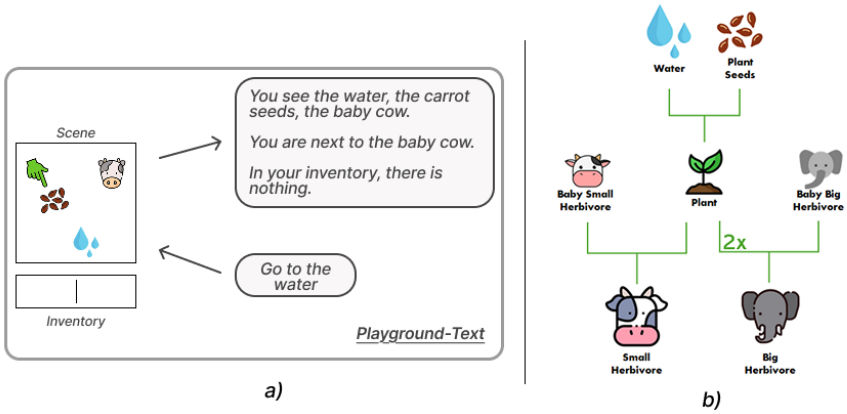


Figure 4.4: *Our experimental setup.* We use the Playground-Text environment that features objects the agent can interact with, along with textual observations and actions (a). We focus on 4 object types from Playground-Text: *Water*, *Plants*, *Small Herbivores*, and *Big Herbivores*. We show in (b) the technology tree for combining objects.

environment introduced in Appendix D.1, a textual environment which produces textual observations and receives textual commands. This environment generates a room that contains multiple objects the agent can interact with by grasping or releasing them. The agent can combine two objects, if an interaction is possible, by releasing one onto the other, as illustrated in Figure 4.4 b. For instance, releasing *water* on a *carrot seed* transforms them into *grown carrot*. In our experiments, we only consider 4 types of objects: *Water*, *Plants* (seed or grown) that require *water* to grow, *Small Herbivores* (baby or grown) that require a grown *plant* to grow, and *Big Herbivores* (baby or grown) that require 2

grown *plants* to grow. We evaluate WorldLLM's ability to discover, predict, and explain how to grow *Plants*, *Small Herbivores*, and *Big Herbivores*. We provide more details on the environment in Appendix C.1. For analysis, we gather the possible transitions into 6 different types: *Standing* when moving on an object, *Holding 1* when grasping an object, *Holding 2* when grasping two objects at the same time, *Grow Plant* when releasing water on a seed, *Grow S. Herbivore* when releasing a grown plant on a baby small herbivore and *Grow B. Herbivore* when releasing two grown plants on a baby big herbivore. These types are the ones used by RL-ALPEXP.

Optimization – We use Phi-3-mini-4k-Instruct[1] for both the Statistician and the Scientist. For our Experimenter, we either use one of our hard-coded oracles or an RL agent. Regarding oracles, O-Ideal collects transitions that match the distribution of the above test set. O-Hardest performs the optimal policy to grow the *Small Herbivore* and the *Big Herbivore* accessible. O-Curriculum plays the optimal policy to grow a *Plant* up to 133 iterations, then plays the optimal policy to grow the accessible *Small Herbivore* up to 266 iterations, and finally grows the *Small Herbivore* and the *Big Herbivore* until the end. See Appendix C.2.3 for more details on the oracles. Concerning the RL-based Experimenters, while it would appear natural to use the LLM already leveraged in the Statistician and Scientist as the policy (e.g., with GLAM), we propose a simpler setup to focus on world modeling abilities. We use a Multi-Layer Perceptron with 2 hidden layers of 64 units as the policy, which uses a symbolic representation of Playground-Text's state as observation. In all experiments below, unless specified otherwise, we use 8 different random seeds. For each seed, we perform 400 iterations of the framework (i.e., T) where 150 transitions are collected per iteration and $n_{steps} = 5$ steps of the Metropolis algorithm are performed by the Scientist at each iteration. When the Experimenter is an RL agent, we collect 3600 transitions per iteration to train it and use PPO to update the policy, while only keeping the last 150 transitions for the Scientist to match the oracles' hyperparameters. Our test set $\mathcal{D}_{test}$ is composed of 120 *Standing*, 20 *Holding 1*, 7 *Holding 2*, 12 *Grow Plant*, 6 *Grow S. Herbivore* and 3 *Grow B. Herbivore*.

WorldLLM with an oracle Experimenter

We initiate the experiments by examining the performance of WorldLLM when using our oracles as Experimenters. In Figure 4.5, we show the evolution of log-likelihoods outputted by the Statistician using the last accepted set of hypotheses H_t throughout WorldLLMs' iterations when using different oracles as Experimenters. These log-likelihoods are computed on the test set and averaged over transition types (and seeds).

One can observe that, regardless of the oracle used or transition type, giving the Statistician hypotheses always improves its performance compared to when no hypotheses are given (i.e., the initial point of each curve). When comparing the oracles, one can observe that the data collection strategy of O-Ideal strongly influences WorldLLM's abilities at producing efficient hypotheses. Indeed, using the same distribution as the one we perform evaluation on, O-Ideal achieves the best overall performance. On the opposite,

[1] https://huggingface.co/microsoft/Phi-3-mini-4k-instruct

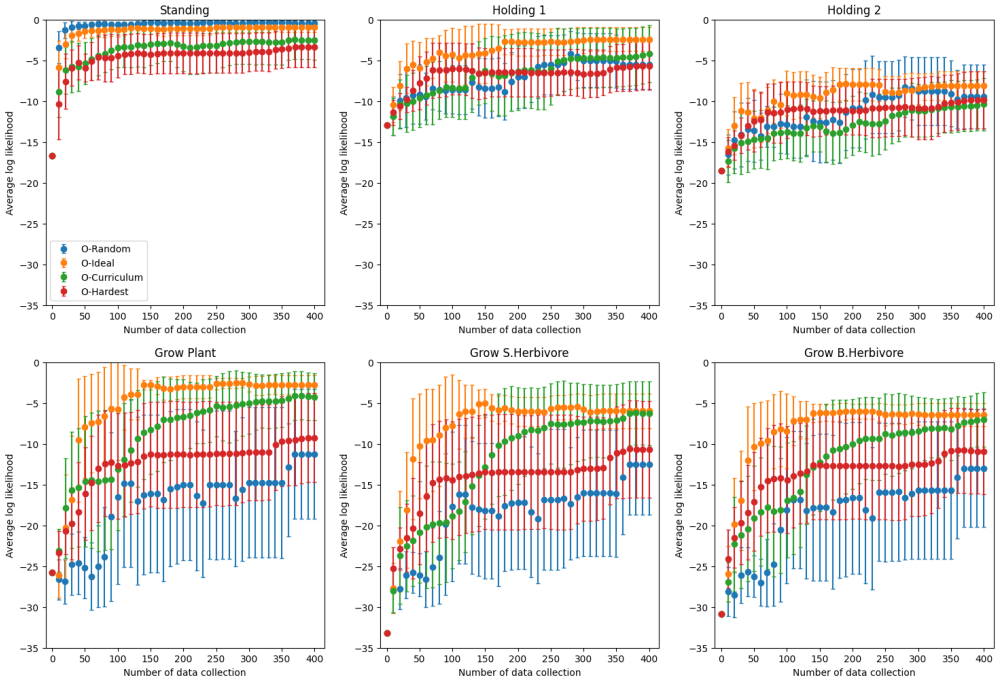


Figure 4.5: Evolution of log-likelihoods computed by the Statistician (using the Scientist's last accepted hypotheses) on the test set throughout WorldLLM's iterations when using different oracles as Experimenters. We average per transition type over 8 random seeds with the standard deviation represented by the error bars. We show at iteration 0 the log-likelihood from the Statistician without any hypotheses given.

O-Random obtains the poorest performance on complex transitions (i.e., growing an object) while being the best method on *Standing* transitions. This can easily be explained by the fact that moving actions are predominant in the action space. O-Hardest did not lead to strong performance, highlighting the need of diversity in collected transitions for the Scientist. Finally, O-Curriculum obtains results that are on par with those of O-Ideal. Results from Appendix C.4.2 indicate that O-Curriculum particularly helps the Scientist generate improved hypotheses.

Overall, these initial experiments demonstrate that WorldLLM is able to produce hypotheses that improve the predictive capabilities of the Statistician's LLM. Moreover, the different performance obtained by our oracles highlighted the importance of 1) diversity in the collected transitions and 2) a developmental collection strategy allowing the Scientist to progressively update the hypotheses.

WorldLLM with a curiosity-driven RL agent as Experimenter

We now move to curiosity-driven RL agents as Experimenters. Compared to the hard coded policies, we rely instead on one of our three intrinsic reward signals (RL-ALPEXP, RL-LogP, and RL-ALP), to collect transitions that drive the Scientist towards generating efficient hypotheses.

As in the previous section, we analyze the evolution of the Statistician's log-likelihoods over the test set. In order to ease the comparison of our three RL-based Experimenters to

oracles, we introduce a coverage metric as the area under the evolution of log-likelihoods' curve of each method. We normalize this area using the area between the Statistician's log-likelihood without any hypotheses and the best performance achievable (i.e., 0). The complete formula is

$$\text{area} = 1 - \int_0^T \frac{\log \mathbb{P}^{S_t}(\mathcal{D}_{test}|H_t)}{\log \mathbb{P}^{S_t}(\mathcal{D}_{test}|\{\emptyset\})T} dt. \tag{4.3}$$

The best performing method possible is expected to be very close to 1 while an inept method's result will be near 0. A method whose result is below 0 means that the hypotheses degrade the initial performance of the Statistician.

Methods	Standing	Holding 1	Holding 2	Grow Plant	Grow S.H.	Grow B.H.
O-Random	0.97 ± 0.01	0.45 ± 0.16	0.40 ± 0.15	0.34 ± 0.24	0.44 ± 0.16	0.40 ± 0.17
O-Ideal	0.92 ± 0.03	0.72 ± 0.12	0.50 ± 0.06	**0.81 ± 0.06**	**0.77 ± 0.06**	**0.74 ± 0.04**
O-Curriculum	0.79 ± 0.15	0.48 ± 0.25	0.31 ± 0.15	0.67 ± 0.15	0.63 ± 0.13	0.60 ± 0.14
O-Hardest	0.74 ± 0.13	0.48 ± 0.20	0.39 ± 0.16	0.52 ± 0.21	0.57 ± 0.18	0.56 ± 0.18
RL-ALPEXP	0.95 ± 0.02	0.44 ± 0.16	0.40 ± 0.14	0.42 ± 0.25	0.52 ± 0.19	0.49 ± 0.19
RL-LogP	0.93 ± 0.01	**0.52 ± 0.07**	**0.53 ± 0.07**	0.62 ± 0.09	0.61 ± 0.07	0.60 ± 0.07
RL-ALP	**0.98 ± 0.01**	0.18 ± 0.08	0.17 ± 0.06	-0.07 ± 0.05	0.16 ± 0.04	0.08 ± 0.07

Table 4.1: This table presents the normalized area under the training curve for each algorithm and transition type. The normalization is performed by dividing the computed area by the area formed by the initial log-likelihoods, i.e., obtained without using hypotheses.

From the results shown in Table 4.1, one can observe that RL-ALPEXP is able to leverage its environment knowledge to reach performance significantly better than O-Random but remaining worse than O-Ideal. When given no expert knowledge, RL-ALP achieves poor performance while RL-LogP obtains results close to or even better than oracle baselines.

To better understand the effect of the RL-based Experimenters on the Statistician's predictions, we studied the evolving distribution of collected transitions. We grouped *Holding 1* and *Holding 2* for simplicity and ignored *Standing* transitions, which are predominant by nature. We show these results in Figure 4.6.

They show that RL-LogP manages to explore the environment and cover the whole transition space. The resulting collected distribution is near O-Hardest's one (which is shown in dashed lines). This performance is notably explained by the syntax used by Playground-Text for observations and actions that make complex transitions naturally harder to predict for an LLM (e.g., moving to a *Big Herbivore* involves only this entity whereas growing a *Big Herbivore* implies an action where two *Plants* are released on a baby *Big Herbivore*). This natural ordering appears in the rewards obtained by the RL policy and favors complex transitions. RL-ALPEXP, which does not leverage such a reward ordering, progressively converges to collecting more complex transitions. When looking at RL-ALP, most of the collected transitions are *Standing* ones. As we have seen in the previous section, the diversity of the transitions given to the Scientist is key in WorldLLM. Consequently, the hypotheses produced with RL-ALP focus solely on explaining *Standing* transitions. A more detailed analysis of why RL-ALP failed is available in Appendix C.4.4.

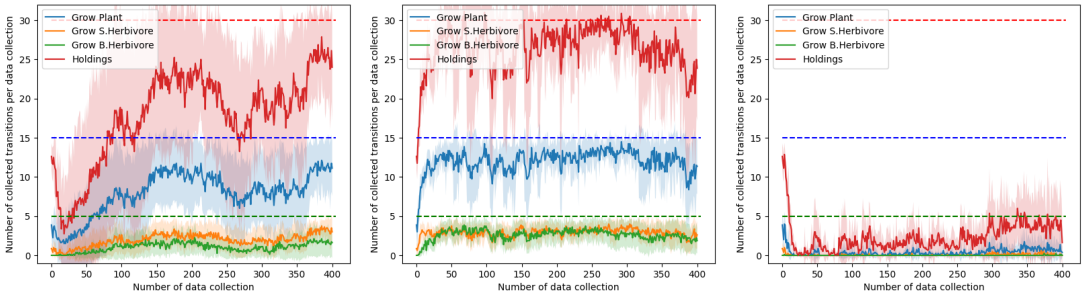


Figure 4.6: Proportion of collected transitions for RL-ALPEXP (left), RL-LogP (middle), and RL-ALP (right). The colored dashed lines correspond to the amount of transitions collected by O-Hardest.

This section's results show that WorldLLM successfully improves the LLM's forward modeling abilities when using curiosity-driven RL-based Experimenters. Moreover, we show that rewarding Experimenters for collecting transitions that help refine the hypotheses not only leads to efficient hypotheses but also leads to policies that solve our environment's full technology tree.

Evaluating the produced world model

We now focus on evaluating the usefulness of the produced world model (i.e., the Statistician equipped with the last accepted hypotheses). While previous sections indicate that WorldLLM succeeds in increasing the likelihood of the observed next states, these results do not guarantee that the observed next states receive the highest likelihood compared to other possible next states. We thus examine our LLM-based world model's capacity to correctly predict (i.e., generate) the next state on our test. In particular, for each state-action pair in $\mathcal{D}_{test}$, we used our Statistician's LLM and constrained decoding to only generate the valid next states, given the current state, in Playground-Text. We then report in Table 4.2 the percentage of next states that belonged to the top-3 generated sequences.

Methods	Standing	Holding 1	Holding 2	Grow Plant	Grow S.H.	Grow B.H.
No Hypo.	0.82 ± 0.00	**1.00 ± 0.00**	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Random	**1.00 ± 0.00**	0.48 ± 0.26	0.07 ± 0.19	0.18 ± 0.31	0.00 ± 0.00	0.04 ± 0.11
O-Ideal	**1.00 ± 0.00**	0.94 ± 0.15	0.00 ± 0.00	**0.94 ± 0.17**	0.25 ± 0.26	0.46 ± 0.37
O-Curriculum	**1.00 ± 0.00**	0.88 ± 0.31	0.00 ± 0.00	0.85 ± 0.28	**0.60 ± 0.41**	**0.54 ± 0.44**
O-Hardest	**1.00 ± 0.00**	0.86 ± 0.21	0.11 ± 0.28	0.29 ± 0.34	0.27 ± 0.37	0.21 ± 0.37
RL-ALPEXP	**1.00 ± 0.00**	0.71 ± 0.29	**0.34 ± 0.36**	0.40 ± 0.43	0.02 ± 0.06	0.08 ± 0.14
RL-LogP	**1.00 ± 0.00**	0.67 ± 0.30	**0.34 ± 0.25**	0.74 ± 0.32	0.02 ± 0.06	0.29 ± 0.42
RL-ALP	**1.00 ± 0.00**	0.20 ± 0.26	0.00 ± 0.00	0.03 ± 0.08	0.00 ± 0.00	0.00 ± 0.00

Table 4.2: Top 3 performance of Constrained Decoding for each algorithm by transition type. We also report the performance without any hypotheses given (No Hypo.).

The results show that no method performs best on all transition types. However, similarly to what was observed in previous sections, *O-Ideal*, *O-Curriculum*, *RL-ALPEXP*

and *RL-LogP* appear to be the best performing Experimenters. More importantly, these results show a significant improvement over the performance without any hypotheses.

Generating hypotheses with WorldLLM vs fine-tuning

WorldLLM improves an LLM's forward modeling abilities by providing it natural language hypotheses about the world. In this section, we study how our approach compares to a more classic gradient-based approach where the LLM is directly fine-tuned on collected transitions. In addition to avoiding gradient computation through the LLM and being human interpretable, the theories produced by WorldLLM are expected to better generalize to changes in the environment.

We use the transitions collected by RL-LogP and fine-tune our Statistician's LLM on them using a causal language modeling objective (i.e., we maximize the likelihood of the next state's tokens given the state-action pair). We name this new approach *F-LogP*. When evaluated on its top-3 constrained decoding accuracy over the test set as in Section 4.2.2, F-LogP obtains perfect results (i.e., accuracy of 1) over all transition types. In the following sections, we study the generalization abilities of both methods and provide a deeper analysis of their impact on log-likelihoods.

Generalization to changes in the environment – We now assess whether using natural language hypotheses in the prompt produces better generalization abilities for the LLM-based world model than fine-tuning it on collected transitions. For this, we create a generalization environment in which the syntax of observations is changed (see Figure C.5). We use this new environment to report the top-3 constrained decoding accuracy on our test set, as in previous sections.

Methods	Standing	Holding 1	Holding 2	Grow Plant	Grow S.H.	Grow B.H.
No Hypo.	0.28 ± 0.00	**0.85 ± 0.00**	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Random	0.69 ± 0.16	0.45 ± 0.26	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Ideal	0.73 ± 0.23	0.46 ± 0.19	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Hardest	0.33 ± 0.33	0.71 ± 0.33	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Curriculum	0.46 ± 0.34	0.72 ± 0.29	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
RL-ALPEXP	0.65 ± 0.16	0.42 ± 0.30	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
RL-LogP	0.43 ± 0.16	0.30 ± 0.23	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
RL-ALP	0.76 ± 0.18	0.61 ± 0.31	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
F-LogP	**0.88 ± 0.17**	0.24 ± 0.25	0.00 ± 0.00	0.07 ± 0.13	**0.21 ± 0.32**	**0.67 ± 0.44**

Table 4.3: Top 3 performance on Constrained Decoding for each algorithm by transition type on the test set using the generalization environment. We also report the performance without any hypotheses given (No Hypo.).

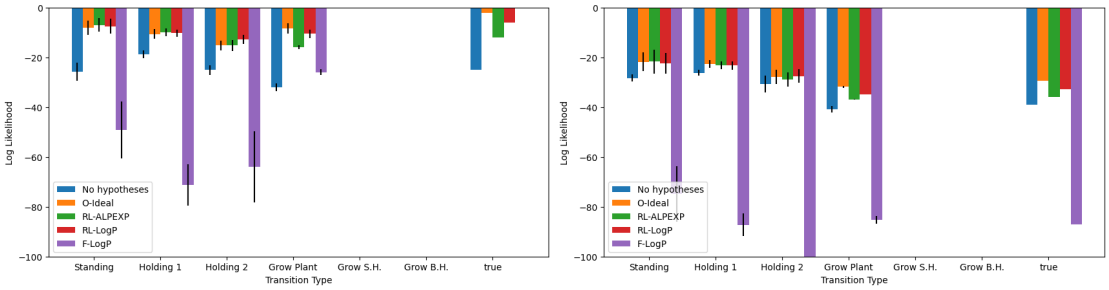
Results from Table 4.3 show an important deterioration of the performance for all methods. F-LogP manages to maintain a non-zero accuracy on complex transition types while WorldLLM-based world models do not perform better than the Statistician without any hypotheses. These results hint that WorldLLM may struggle in finding abstract hypotheses that generalize. The next section provides additional insights on these results.

Effect on log-likelihoods – We now propose to study how both methods affect the Statistician's log-likelihoods. We show in Figure 4.7 the log-likelihoods (averaged by transition type) produced by the Statistician when facing an instance of a *Grow Plant* transition.

Looking at results when using the classic environment (a), WorldLLM increases the log-likelihood of all the possible transitions (no matter the Experimenter) while maintaining the correct transition more likely. In comparison, the causal language modeling objective used for F-LogP increases the log-likelihood of the correct transition while significantly decreasing the log-likelihood of all the other transitions. These results illustrate the difference in the two optimization processes: while fine-tuning aims to increase the distance in log-likelihoods between the correct transition and the other ones, WorldLLM focuses on increasing the log-likelihood of the correct transition, regardless of the others.

When focusing on performance with the generalization environment, WorldLLM leads to very few changes compared to when no hypotheses are given. On its side, F-LogP produces very low log-likelihoods for all transitions and keeps the correct one slightly above the others. Overall, these results show that both approaches generalize poorly.

One explanation for WorldLLM's poor generalization performance lies in the hypotheses produced. We provided samples in Appendix C.4.3 which show that our Scientist notably used examples of transitions in the hypotheses (even though some hypotheses did mention abstract categories or even mentioned objects not in our environment). Such examples no longer align with the generalization environment's syntax. Multiple explanations can be given for this. First, our current experiments rely on a Scientist's LLM with limited capabilities. Second, while our objective in 4.1 minimizes the number of hypotheses, the current implementation of WorldLLM only focuses on maximizing the Statistician's log-likelihood. We argue that favoring short hypotheses could help abstract theories emerge. We leave these investigations for future work.



(a) No changes to the environment.

(b) Generalization environment.

Figure 4.7: Log-likelihoods (averaged by transition type) produced by the Statistician when facing an instance of a *Grow Plant* transition. We show how using the generalization environment from Section 4.2.2 affects the log-likelihoods (b).

4.2.3 Related work

Building forward models

Learning a forward model has been used as an intrinsic reward to foster RL policies' exploration (Schmidhuber, 1991b; Pathak et al., 2017; Burda et al., 2019a; Oudeyer & Kaplan, 2007). WorldLLM employs a similar curiosity-driven scheme where our RL-based Experimenters use reward signals from prior work to promote exploration and collect transitions that help improve the forward model. However, our method differs in the way the model is improved: instead of using a gradient-based approach to fine-tune the LLM-based forward model, as done in (Xiang et al., 2023), WorldLLM searches for natural language theories that improve the forward model when given in the prompt. While recent work studied how a forward model could leverage external textual information such as a manual explaining the environment (Zhang et al., 2024a; Dainese et al., 2023), our approach iteratively generates and refines its own explanations.

Inductive reasoning for world modeling

We discussed in Section 4.1 the importance of intuitive theories in shaping one's representations about the world. Building on this, the literature of theory-based RL has been advocating for augmenting artificial agents with inductive reasoning (Tsividis et al., 2021). In particular, (Tsividis et al., 2021; Tang et al., 2024; Li et al., 2024) proposed approaches based on LLMs that generate programs representing the theories about an environment's dynamics. The use of programs allowed the authors to execute and compare how the proposed theories match empirical evidence. Parallel to this, (Piriyakulkij et al., 2024; Wang et al., 2024b) proposed to generate theories using natural language. While the former constrained the generated theories to programs that can be easily evaluated in their environment, the latter used the generated theories to condition a program generator to simulate the environment's dynamics. WorldLLM also builds on the theory-based RL literature and closely relates to (Piriyakulkij et al., 2024). Notably, WorldLLM also leverages Bayesian inference with an LLM as the proposal distribution to generate the theories. However, the major difference lies in how the produced theories are used: while (Piriyakulkij et al., 2024) constrain the format of theories such that they can be directly tested in the environment, our work focuses on giving these theories in another LLM's prompt to improve its forward modeling capabilities. *In other words, our work strongly differs in its perspective of producing theories that improve our LLM's world modeling abilities instead of producing correct theories.* Nonetheless, the Bayesian inference framework we used in our Scientist is largely inspired by (Piriyakulkij et al., 2024). Both our work and theirs also consider an active learning paradigm to choose which experiments to perform for theories' refinement. While their approach computes exact Information Gain of all possible experiments, ours relies on curiosity-driven RL agents and Information Gain approximates as rewards to overcome the intractability of exact Information Gain in most environments.

4.3 Discussion

In this chapter, we explored the predictive and modeling dimensions of functional grounding in LLMs. Drawing inspiration from cognitive development, we highlighted

how children iteratively refine intuitive theories to make sense of their environment. We connected these insights to the field of model-based RL, particularly approaches that reward exploration based on forward model errors—mirroring how humans seek informative experiences to update their mental models.

Building on this foundation, we introduced WorldLLM, a framework for functionally grounding world modeling abilities in LLMs. WorldLLM generates natural language hypotheses—interpretable, theory-like statements—which condition the LLM when predicting state-action outcomes. This approach enables world modeling without expensive fine-tuning of the LLM itself and avoids the need for a handcrafted theory space. Our experiments demonstrated that the adversarial dynamic between a Scientist (hypothesis generator) and simple curiosity-driven Experimenters (data collectors) supports broad exploration and the discovery of environment dynamics.

Despite promising results, our initial implementation of WorldLLM presents several limitations. First, the Playground-Text environment is structurally simple, and the natural ordering created by the most challenging transitions being longer and harder to predict is not expected in more complex environments—potentially reducing the effectiveness of RL-LogP. Evaluating WorldLLM in more complex environments is therefore a critical direction for future work (Ying et al., 2025).

Second, our current Experimenters use lightweight neural networks and symbolic inputs. A more compelling alternative would be to integrate LLM-based Experimenters, possibly unifying all WorldLLM components with a single GLAM-equipped LLM that learns both to control and to improve its forward model via online RL. Furthermore, our experiments only used one LLM model, which showed limitations in the Scientist role—particularly in generating abstract, high-level hypotheses.

Yet, as we argued in Section 4.1, abstract reasoning is a key characteristic of human theories. Its emergence in LLMs may require more than scaling the base model. A key advantage of Bayesian inference in this context is the ability to incorporate *priors*, which constrain the theory space. While a length-based prior might encourage abstract hypotheses, we believe richer priors—such as "error maps" that represent structured patterns of predictive failure, as described by Schulz (2012)—may be necessary to better replicate human-like theory refinement.

Additionally, our current Metropolis-based approach for updating hypotheses only considers the most recent transitions, which can lead to forgetting, especially with non-stationary Experimenters. This was primarily motivated by practical constraints: the LLM's limited context window and the cost of evaluating likelihoods on large transition sets. These issues could be mitigated by storing past transitions in a replay buffer or employing more sophisticated Bayesian inference methods (e.g., particle filters), which would also allow tracking multiple particles to avoid local optima—a current limitation evidenced in Appendix C.4.

Finally, our exploration strategy relied solely on simple curiosity-driven signals focused on poorly predicted transitions. As discussed in Chapter 2, other intrinsic motivation mechanisms have been studied in both humans and artificial agents. A particularly important one is *autotelic learning*, where learners generate, select, and attempt to master self-defined goals (Steels, 2004; Colas et al., 2022b). In the next chapter, we extend our investigation of curiosity-driven exploration to more complex and open-ended

environments. In particular, we explore how functional grounding in such settings can be supported by key mechanisms underlying autotelic learning—such as scaffolding, hindsight relabeling, and metacognitive monitoring.

Chapter 5

Towards functional grounding in complex goal spaces

Contents

5.1 Social interactions for autotelic functional grounding: hindsight relabeling and off-policy RL with SAC-GLAM 91
5.1.1 SAC-GLAM 91
5.1.2 Experiments 94
5.1.3 Conclusion 97
5.2 MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces 98
5.2.1 Related Work 100
5.2.2 Methods 101
5.2.3 Experiments 106
5.2.4 Conclusion 111
5.3 When goals are beyond reach: Metacognitive monitoring guides autonomous discovery of frugal assistance-seeking in LLMs 112
5.3.1 Related Work 114
5.3.2 Methods 116
5.3.3 Experiments 118
5.3.4 Conclusion 124
5.4 Discussion 124

In Section 2.2.3, we discussed how curiosity-driven learning plays a central role in human cognitive development. In particular, humans are open-ended learners, continuously exploring and acquiring new skills throughout their lives. Crucially, humans are *autotelic learners*—intrinsically motivated to represent, invent, select, and pursue their own goals (Steels, 2004; Colas et al., 2022a). To navigate a potentially infinite space of goals—without the time or capacity to exhaustively explore it—they rely on intrinsic motivation signals (Baldassarre & Mirolli, 2013; Gottlieb & Oudeyer, 2018). In this chapter, we argue that functionally grounding LLMs presents a similar challenge: there exists a vast space of possible skills that an LLM could acquire through functional grounding, yet time and interactions are inherently limited.

Importantly, autotelic learning in humans is not purely an individual endeavor. Developmental psychology, beginning with Lev Vygotsky and followed by others, emphasizes the role of social guidance—especially from caregivers—in shaping cognitive and linguistic development (Vygotsky, 1962). This guidance takes various forms. For instance, adults often provide linguistic descriptions of events, which help children focus attention and facilitate language acquisition (Smith & Gasser, 2005; Golinkoff et al., 2015). Drawing inspiration from this, Colas et al. (2020) proposed an autotelic RL agent that uses language to invent its own goals and learns to evaluate goal success through descriptions provided by a social partner. By combining this social feedback with hindsight relabeling and off-policy RL, they showed that the agent could autonomously learn diverse skills and generalize effectively—without relying on any extrinsic reward.

We begin this chapter's contributions by extending the functional grounding framework to environments involving social interaction. Specifically, we consider scenarios in which a social partner provides behavioral descriptions to the LLM agent. In Section 5.1, we introduce a new method—SAC-GLAM—which incorporates hindsight relabeling and off-policy RL into the GLAM framework. This work constitutes an important contribution to designing autotelic LLM agents able to leverage social interactions to efficiently improve their functional competence. Additionally, by providing an in-depth analysis of SAC's integration in LLM agents, we also contribute to improving the efficiency of RL-based fine-tuning of LLMs.

Another important form of guidance is what has been described as *scaffolding* by Bruner (1985); Wood et al. (1976): caregivers tailor and adapt tasks based on a child's current abilities. This plays a major role as it constrains the learning opportunities for children who face a rich and possibly infinite environment to explore. Effectively identifying such tasks requires the caregiver to estimate the child's current competence—that is, to assess what the child knows and does not know. These insights have inspired various works in ML aimed at organizing the learning experiences of artificial agents. While curricula can be hand-crafted by human experts (Bengio et al., 2009), the field of automatic curriculum learning (ACL) focuses on designing teacher algorithms that autonomously sequence learning tasks for an artificial learner. Although ACL has shown mixed success in supervised learning contexts, it has proven particularly effective in RL, where adjusting the difficulty of tasks presented to an agent is crucial for efficient policy discovery (Portelas et al., 2020b; Narvekar et al., 2020). In Appendix H, I provide a benchmark of ACL methods used in RL contexts, assessing their robustness to various task spaces (e.g., mostly infeasible or with a rugged difficulty landscape). These methods notably differ in the motivation signal they use to estimate which goals are most beneficial for the learner at a given time (e.g., Learning Progress or intermediate difficulty).

But beyond external guidance, humans also develop internal mechanisms for task prioritization. As discussed earlier and in Chapter 2, such mechanisms are particularly crucial in autotelic learning, where individuals must navigate large and complex goal spaces without predefined curricula. A key component of effective goal selection is the emergence of metacognitive abilities—allowing learners to monitor their own competence and prioritize tasks with the highest learning potential (Ten et al., 2021). In Section 5.2, we extend this idea by exploring how to endow LLMs with metacognitive monitoring abilities—specifically, the ability to estimate what they already know versus what they have yet to learn. We investigate how LLM agents can develop competence estimation

mechanisms over vast task spaces and use these estimates to autonomously prioritize their learning trajectories. Our proposed method, MAGELLAN, enables LLM agents to track Learning Progress across large language-defined task spaces by leveraging semantic relationships between tasks. Experiments show that MAGELLAN effectively prioritizes tasks and that, through metacognitive estimation, it shapes the LLM's internal task representations to better reflect the structure of the external environment.

This chapter concludes by addressing scenarios in which tasks exceed the LLM's capabilities but assistance (e.g., calling another LLM) is available. In particular, we demonstrate that augmenting LLMs' metacognitive skills is not only useful for scaffolding their training but also enables LLMs to track when it is best to rely on external help. Modeling this as a multi-objective bandit problem in which the LLM must balance maximizing task performance with minimizing the cost of external help, we show that LLMs can learns an optimal strategy for this trade-off through online interaction alone.

Collaborations and scientific output – Sections 5.1 and 5.2 stem from the same Master's internship conducted during the summer of 2024 with Loris Gaven, under the co-supervision of Thomas Carta and myself. The initial goal of the internship was to transition a GLAM-based LLM agent toward an autotelic agent by integrating off-policy RL and hindsight relabelling into GLAM. Loris, Thomas, and I jointly led the project and were involved in all scientific and technical decisions throughout the internship, with Loris handling the implementation and execution of the experiments. This first phase of the work resulted in a workshop paper (presented in Section 5.1): *Loris Gaven, Clément Romac, Thomas Carta, Sylvain Lamprier, Olivier Sigaud, Pierre-Yves Oudeyer. 2024. SAC-GLAM: Improving Online RL for LLM agents with Soft Actor-Critic and Hindsight Relabeling. Intrinsically Motivated Open-ended Learning (IMOL) Workshop, Neural Information Processing Systems (NeurIPS).* To support our investigation of hindsight relabelling, we designed an environment featuring a complex language-specified goal space. However, early experiments using uniform random goal selection quickly revealed limitations, prompting us to explore goal prioritization strategies based on automatic curriculum learning. This second phase extended into late 2024 and culminated in a conference paper, presented in Section 5.2: *Loris Gaven, Thomas Carta, Clément Romac, Cédric Colas, Sylvain Lamprier, Olivier Sigaud, Pierre-Yves Oudeyer. 2025. MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces, Proceedings of the 42nd International Conference on Machine Learning (ICML). Vol. 267.* In parallel, beginning in late 2023, I initiated a project applying GLAM-style fine-tuning to help LLMs discover tool-use strategies. After several iterations and redirections, this work matured into the contribution presented in Section 5.3, which I carried out as sole first author. The project's direction was notably shaped by my discussions with Thomas Carta, Loris Gaven, Cédric Colas, as well as Nicolas Yax. Nicolas played a key role in shaping both the project presented in Section 5.3 and MAGELLAN, thanks to his insightful feedback on metacognition for LLMs.

5.1 Social interactions for autotelic functional grounding: hindsight relabeling and off-policy RL with SAC-GLAM

While Chapter 3 demonstrated how GLAM can functionally ground LLMs through online interaction, scaling this process to support the open-ended acquisition of functional competence across infinite task spaces requires granting LLM agents greater autonomy—specifically, the ability to freely explore their environment and define their own goals (Colas et al., 2022b). As discussed in Chapter 2, language—and, more recently, LLMs—has played a central role in the design of such agents. For example, LLMs have been used to generate novel goals (Colas et al., 2020; Wang et al., 2023a; Zhang et al., 2024b; Colas et al., 2022a) or to produce task-specific reward functions (Colas et al., 2020; Fan et al., 2022; Du et al., 2023). However, the construction of fully autotelic LLM agents remains an underexplored challenge. This chapter seeks to address that gap by adapting mechanisms from classic autotelic agents—such as those introduced in (Colas et al., 2020)—to the context of LLMs.

In particular, autotelic architectures usually involve experience replay and hindsight relabeling to best utilize all trajectories, even unsuccessful ones, which can be predominant when agents set their own goals. However, most current approaches to fine-tune LLM agents with RL rely on on-policy algorithms that cannot handle experience replay and trajectory relabeling mechanisms. While a handful of attempts (e.g., (Wen et al., 2024a; Putta et al., 2024)) to use off-policy RL on LLMs exist, they consider token-level actions (better suited for text generation tasks) and complex architectures that balance token-level actions and environment-level actions (usually sequences of tokens). As a step towards autotelic LLM agents, this work introduces a version of Soft Actor-Critic (SAC) (Haarnoja et al., 2018) explicitly designed for LLMs as RL agents combining the simplicity of prior on-policy methods focusing on environment-level actions (e.g., GLAM) and the advantages of off-policy learning. We then extend it with Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) and show our new method (**SAC-GLAM**) exhibits higher sample efficiency than GLAM (called PPO-GLAM in our experiments) in a classic multi-goal environment while staying on par in time efficiency in spite of the stability and efficiency challenges of using an actor-critic approach with a pre-trained LLM-based policy and a randomly initialized critic.

5.1.1 SAC-GLAM

We consider a textual RL setting where, given a language vocabulary $\mathcal{V}$, the environment returns an observation $o \in \mathcal{V}^N$ and a reward $r \in \mathbb{R}$ after an action $a \in \mathcal{A} \subseteq \mathcal{V}^M$ (i.e., actions are sequences of tokens). The task or goal description $g \in \mathcal{G} \subseteq \mathcal{V}^K$ conditions the reward. This environment can be modeled as a goal-augmented partially observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G}, \mathcal{U}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{G}$ is the goal space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ is the goal-conditioned reward function, $\mathcal{U} : \mathcal{S} \to \mathcal{V}^N$ is the observation function that maps a state to a textual description, and γ is the discount factor.

In such settings, our previous empirical contributions showed that PPO shines for its efficiency and simplicity when fine-tuning stochastic pre-trained policies such as LLM

agents. However, on-policy RL approaches such as PPO cannot theoretically leverage HER, which, by relabeling trajectories, introduces off-policy transitions. We propose in the sections below an off-policy alternative to PPO that keeps GLAM's simplicity in how the LLM is used as a stochastic policy over discrete environment-level actions (i.e., sequences of tokens).

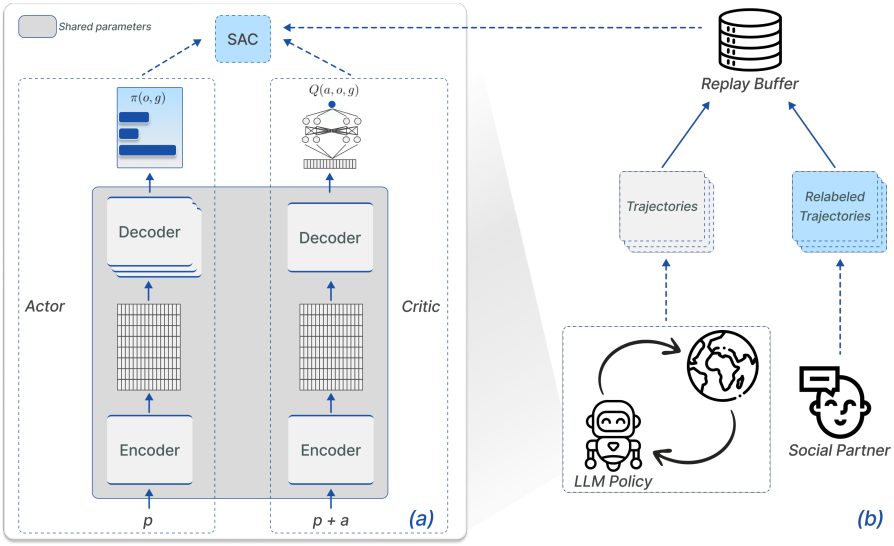


Figure 5.1: **The SAC-GLAM method.** (a) depicts the agent's architecture when an encoder-decoder LLM is used: the actor computes an action probability as the probability computed by the LLM of the action's tokens to follow the observation and goal concatenated in the prompt p, while the critic computes the Q-value for each action a and the prompt p with an MLP attached to the decoder's last hidden state. (b) illustrates the agent-environment interaction, where trajectories are generated and added to the replay buffer. We used an environment where a social partner relabels these trajectories with hindsight goals.

Soft Actor Critic with an LLM actor

We adapt the discrete version of SAC (Christodoulou, 2019) to our stochastic LLM-based policy. We first follow GLAM's approach to obtaining a stochastic policy with an LLM by computing the probability of each action $a_i \in \mathcal{A}$ from our environment as the probability of its token sequence $a_i = \{w_1, w_2, ..., w_{|a_i|}\}$ (with $w_i \in \mathcal{V}$) to follow a prompt containing both o and g:

$$\mathbb{LP}_{LLM}(a_i|o,g) = \sum_{j=0}^{|a_i|} \log \mathbb{P}_{LLM}(w_j|o,g,w_{<j})\,. \tag{5.1}$$

Subsequently, a softmax function is applied to generate a probability distribution over all possible actions and obtain our stochastic policy:

$$\pi(a_i|o,g) = \frac{e^{\mathbb{LP}_{LLM}(a_i|o,g)}}{\sum_{a_j \in \mathcal{A}} e^{\mathbb{LP}_{LLM}(a_j|o,g)}}\,. \tag{5.2}$$

Then, we follow SAC and simultaneously learn both this policy and a Q-function, which we implement as an MLP added on top of the final decoder block of our LLM (as in GLAM).

Pre-trained policy and randomly initialized critic

When running SAC with a pre-trained LLM as the policy, several unique challenges arise. In SAC, the actor's updates rely on feedback from the critic. While poor estimates from a randomly initialized critic may not significantly impact a randomly initialized policy, our situation is different. We aim to protect the pre-trained LLM actor from harmful updates that could degrade its performance.

For this, we introduce a warm-up period where only the critic is trained. To make this warmup period as short as possible, we speed up the critic's convergence (1) by backpropagating gradients through both the MLP and the LLM (even though it might affect the actor) and (2) by having a critic with a single head that outputs the Q-value of a state-action pair by concatenating them in the prompt. While the latter requires $|\mathcal{A}|$ forward passes through the LLM and MLP to obtain all the actions' Q-value, it significantly improves convergence.

Our experiments from Section 5.1.2 also use n-step returns with a target expressed as:

$$y_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \mathbb{E}_{a_{t+n} \sim \pi(\cdot|o_{t+n}),g} \left[Q(s_{t+n}, a_{t+n}) - \beta \log \pi(a_{t+n} \mid o_{t+n}, g) \right] . \tag{5.3}$$

In the standard SAC algorithm, a single-step return is used, which corresponds to setting $n = 1$. Using n-step returns reduces the reliance on bootstrapping by accumulating rewards over multiple steps before bootstrapping, often leading to faster critic convergence. However, n-step returns introduce the risk of older transitions becoming highly off-policy, as intermediate rewards and actions depend on the policy in place when the transitions were collected. To mitigate this, we reduce the size of our replay buffer (see hyperparameters in Appendix D.4).

We provide a study of these architectural choices in our ablations in Section 5.1.2.

Hindsight Experience Replay

We augment our SAC-GLAM agent with HER, enabling the algorithm to learn from failed attempts by relabeling portions of the failed trajectories with accidentally reached goals. At the end of an episode, the trajectory with its initial goal and the relabeled sections of the trajectory are added to the replay buffer. We use the *future* strategy proposed in Andrychowicz et al. (2017) (i.e., a trajectory is relabeled with every achieved goal and all these relabeled trajectories are added to the replay buffer). When training, we randomly sample a batch of transitions from this buffer such that the batch contains 50% of relabeled transitions. This is inspired by how Colas et al. (2020) sampled their batch, but we show in Appendix D.3 that this only has little impact in our case.

5.1.2 Experiments

We evaluated our method in the Playground-text environment, a text-based adaptation (goals, observations, and actions are represented as text) of the original Playground environment. This environment was initially introduced by Colas et al. (2020) to study autotelic RL agents that interact with a social partner describing goals reached during a trajectory. In this environment, the agent receives a textual goal and must interact with different objects (plants, animals, food, and water) to complete the task. As this work focuses on augmenting LLM agents with HER and SAC, we do not study how such agents can learn a reward function based on the partner's feedback. Consequently, we define an extrinsic reward where the agent receives 1 if the task is completed and 0 otherwise. Moreover, in the original Playground environment, the agent moves along the x and y axes to reach objects. In the text-based version, we simplified this by introducing moving actions: "*Go to {object}*", which directly moves the agent to the selected object. The environment features two types of tasks: "*Grasp {object}*" and "*Grow {object}*". Animals can be grown using either water or food, while plants can only be grown using water. In Playground-Text, we also introduce sequential tasks, where the agent must complete two tasks in the correct order, significantly expanding the task space to 8,470 tasks. A detailed description of the environment is given in Appendix D.1.

We compared SAC-GLAM—with and without HER—to PPO-GLAM using Flan-T5 250M (Rae et al., 2022). As shown in Figure 5.2, SAC-GLAM alone achieves higher sample efficiency than PPO-GLAM for a fixed budget of 400K environment steps, although it lags behind in time efficiency. This outcome stems from the nature of off-policy methods like SAC, which can sample large batches from the replay buffer, exceeding the number of new environment steps since the last update. While this typically yields better sample efficiency (see ablations below), it also incurs higher computational costs—particularly when computing gradients for large models such as LLMs. Consequently, SAC-GLAM is slower than PPO-GLAM when considering wall-clock time to reach 400K environment steps. However, when augmented with HER, SAC-GLAM not only improves sample efficiency further but also closes the gap in time efficiency, achieving performance comparable to PPO-GLAM in terms of total training time.

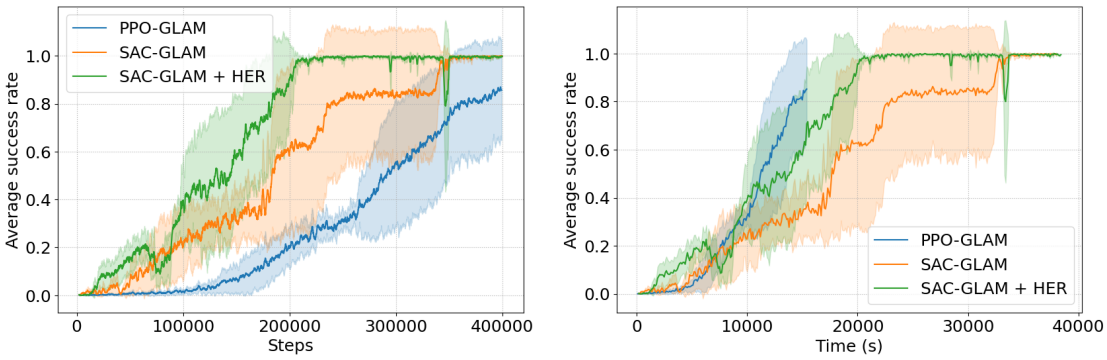


Figure 5.2: **Performance comparison of SAC-GLAM and PPO-GLAM in the Playground-Text environment.** We show the average success rate as a function of the number of steps (left) and the average success rate over time in seconds (right). The mean and standard deviation are calculated across 4 seeds.

Ablations

We now present a series of ablation studies exploring architectural choices and hyperparameters in SAC-GLAM (without HER), using a simplified environment setup that excludes sequential goals and limits the object set to $N = 3$. We begin by examining alternative architectures and learning schemes for the critic, aiming to improve convergence speed, as previously discussed in Section 5.1.1. In our configuration, the critic is implemented as an MLP appended to the final decoder block of the LLM, resulting in shared weights between the actor and critic. We investigate two distinct approaches for computing Q-values:

1. **Observation input:** The critic takes the observation as input and outputs Q-values for each possible action. This method is commonly used in discrete action spaces.
2. **Observation-action input:** The critic takes the concatenation of the observation and action as input, producing a single Q-value. This approach is more typical in continuous action spaces.

As well as two ways to fine-tune our critic:

1. **Backpropagation through the LLM:** The critic shares parameters with the actor and allows gradient backpropagation through these shared layers, enabling joint learning of the underlying representations.
2. **No changes to the LLM:** In this case, the critic shares parameters with the actor, but gradients are backpropagated only through the critic's MLP head, keeping the shared layers only affected by policy improvement.

As shown in Figure 5.3, the critic architecture that conditions on both the observation and the action outperforms the variant that relies solely on the observation. Furthermore, the configuration in which gradients are propagated through both the MLP and the shared LLM parameters achieves better performance than the version where gradient flow is restricted to the MLP head. We hypothesize that this increased stability arises from the richer latent representations passed to the MLP—representations that more effectively encode both the prompt and the candidate action.

We then examine the influence of hyperparameters controlling the updates. In RL, the update frequency plays a crucial role in balancing sample efficiency against the risk of overfitting to collected data. This is especially critical in on-policy settings, where updates rely exclusively on recent samples. However, in the context of large-scale policies such as LLMs, tuning the update frequency is equally important in off-policy settings, since gradient updates can become slower than data collection itself. In addition, both PPO and SAC include a hyperparameter that determines how many gradient updates are performed per batch of collected data—a factor that also significantly affects sample efficiency.

We study different values for these parameters for both SAC and PPO. In the case of PPO, we observed that updating the model too frequently (every 1024 steps) leads to the method plateauing at a suboptimal policy. Additionally, performing too many epochs

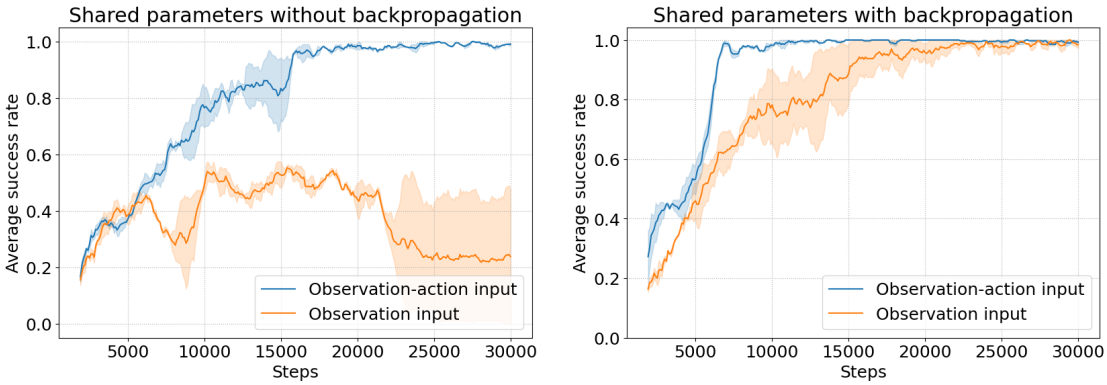


Figure 5.3: **Comparison of critic architectures.** The blue curve represents the architecture where both the observation and action are inputs, producing a single Q-value. The orange curve corresponds to the architecture where only the observation is input, producing Q-values for each action. The left plot illustrates the case where gradients are backpropagated only through the MLP head, while the right plot shows the case where gradients propagate through both the MLP and shared LLM parameters. Mean and standard deviation are computed across two seeds.

(32) at each update introduces instability in the training process (Figure 5.4). Concerning SAC, the method benefits from frequent updates with multiple epochs, inducing a balance between sample and time efficiency (frequent updates and multiple epochs imply slower training).

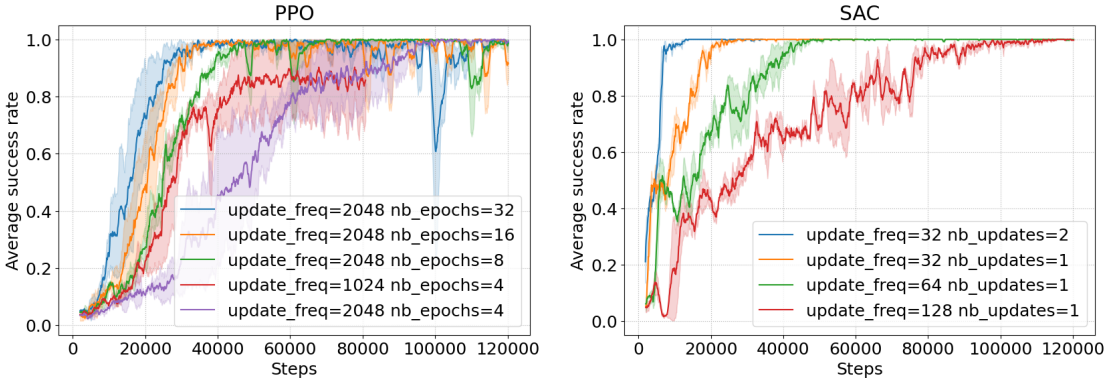


Figure 5.4: **Comparison of different update frequencies and epochs for PPO and SAC.** In PPO, *nb_ epochs* refers to the number of times the model processes each transition during an update, while, for SAC, it represents the number of batches sampled from the replay buffer during each update.

We conclude our ablation studies by examining the impact of the Temporal Difference (TD) loss on sample efficiency. Increasing the n parameter in the n-step return enhances the magnitude of the bootstrap estimate, allowing reward signals to propagate more effectively to earlier steps in the episode. This often accelerates convergence. However, larger values of n also introduce greater off-policy bias, which can destabilize learning. To mitigate this effect, we experimented with reducing the size of the replay buffer,

thereby limiting the age of sampled trajectories. Figure 5.5 compares the performance of SAC-GLAM under varying n values and replay buffer sizes.

Based on these results, we chose to use 3-step returns in our experiments, as 5-step returns introduced instability. Additionally, we opted for a replay buffer with a capacity of 100000, which provided greater stability compared to using a larger buffer.

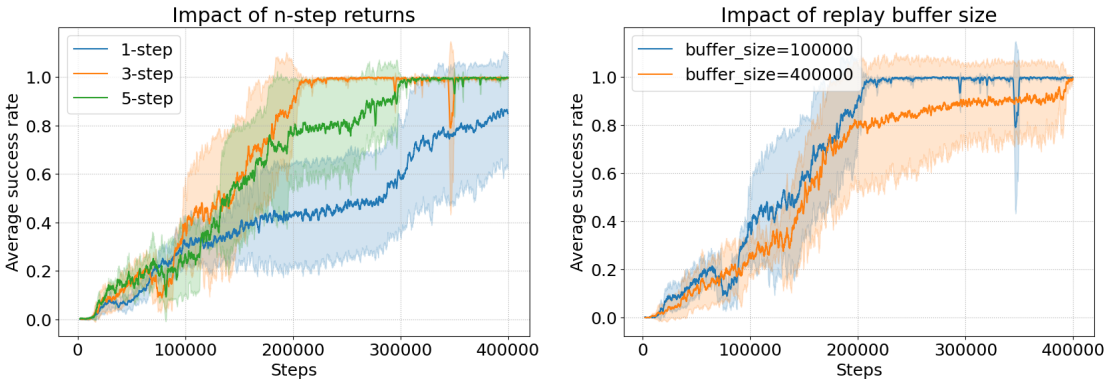


Figure 5.5: **Comparison of n-step parameters and replay buffer size.** The left plot illustrates the impact of the n parameter (using a replay buffer with a capacity of 100000), while the right plot shows the effect of the replay buffer capacity parameter (when using 3-step returns).

5.1.3 Conclusion

In this work, we introduced SAC-GLAM, an off-policy RL approach that adapts SAC and HER for the functional grounding of LLMs. We highlighted the unique challenges that actor-critic methods pose when applied to pre-trained policies such as LLMs, particularly in balancing policy and critic convergence. Our ablations showed that design choices such as introducing a warmup period, sharing weights between actor and critic, using a single Q-value head, and employing n-step returns allow SAC-GLAM to achieve comparable time efficiency to PPO-GLAM while exceeding it in sample efficiency. Beyond outperforming PPO-GLAM in a standard multi-goal RL setting, SAC-GLAM also lays the groundwork for autotelic functional grounding of LLMs—where hindsight experience replay and off-policy RL are essential mechanisms.

Nonetheless, SAC-GLAM has several limitations. While appending the action to the prompt in the critic architecture improves convergence speed, it becomes computationally expensive as the action space grows. Moreover, the broader challenge of stabilizing actor-critic training when the actor is a large, pre-trained model remains an open question deserving further investigation. Lastly, although HER enables autotelic agents to extract learning signals from unsuccessful trajectories—which naturally arise when goals are self-selected—such trajectories are still costly. A promising way to reduce their frequency is to improve the goal-selection strategy by selecting goals that match the agent's current abilities.

In the next contribution, we shift our focus to the goal-selection aspect when turning LLMs into autotelic RL agents. In particular, we introduce MAGELLAN, an ACL

approach for functional grounding of LLMs. MAGELLAN augments LLM agents with metacognitive capabilities, allowing them to estimate their own competence and select goals accordingly.

5.2 MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces

We previously discussed how humans are open-ended learners, continuously exploring and acquiring new skills throughout their lifetime through curiosity-driven learning (Berlyne, 1954; Kidd & Hayden, 2015). This exploration is often goal-directed: caregivers initially guide learners by selecting appropriate tasks (i.e., scaffolding), and over time, this process becomes internalized, giving rise to *autotelic learning*. To navigate an effectively infinite space of potential goals—without the time or resources to explore it exhaustively—humans rely on intrinsic motivation signals (Baldassarre & Mirolli, 2013; Gottlieb & Oudeyer, 2018). Among these, research has highlighted the central role of *Learning Progress* (LP)—that is, the improvement in one's ability to achieve a goal—as a key signal driving exploration and guiding developmental learning (Kaplan & Oudeyer, 2007).

Computational modeling work showed both how it enables efficient automatic curriculum learning (Lopes et al., 2012; Poli et al., 2024) and how it generates developmental trajectories that simulate key properties in the development of human infants (Oudeyer & Smith, 2016). Recently, several experimental paradigms where humans were free to explore various learning activities confirmed that humans use metacognitive LP monitoring to explore and prioritize goals (Ten et al., 2021; Leonard et al., 2023; Sayalı et al., 2023; Poli et al., 2024). Autotelic artificial agents selecting goals that maximize LP were shown to efficiently allocate the agent's learning time by avoiding goals that are either too easy or too difficult (e.g., see the survey from Portelas et al. (2020b) as well as the benchmark from Appendix H), enabling even physical robots to acquire complex skills like tool use in just a few dozen hours (Forestier et al., 2022). However, while these methods show promise in constrained settings, scaling them to open-ended learning remains challenging. The key difficulty lies in efficiently estimating an agent's current competence and expected LP across potentially infinite, evolving, and high-dimensional goal spaces — a fundamental challenge we address in this work.

In particular, current approaches leveraging LP fall short at handling discrete, high-dimensional and structured goal space, such as language-specified goals that LLMs face when being functionally grounded. Indeed, existing methods either work only on small low-dimensional goal spaces (Baranes & Oudeyer, 2013; Portelas et al., 2020a; Kanitscheider et al., 2021; Zhang et al., 2024b) or rely on expert-defined goal groupings to reduce the number of goals (Colas et al., 2019; Akakzia et al., 2021; Kumar et al., 2024). In particular, none of them are able to capture the semantic relationships between goals to efficiently estimate an LLM agent's generalization abilities.

In this work, we study how to estimate LP over natural language goals such that an LLM agent learning with online RL in an interactive environment could increase its overall functional competence as efficiently as possible. For this, we introduce **MAGELLAN**, for **M**et**A**cognitive **GE**neralization of **L**earning progress in **LAN**guage model

agents. MAGELLAN leverages the LLM inside the agent to learn an LP estimator that automatically learns semantic relationships and tracks competence transfer between goals in a sample-efficient manner (see Figure 5.6). We evaluate MAGELLAN in the Little-Zoo environment specifically designed as a carefully controlled experimental setup for commonsense-based generalization of agents in a textual environment. In particular, we study the following scientific questions:

• **Q1.** Given an initial set of language goals, how does MAGELLAN's estimation of a learner's competence compare to more classic methods? How does this estimation scale with the size of the goal space?

• **Q2.** Can MAGELLAN be used by an online RL LLM agent to self-organize an efficient learning curriculum over these goals?

• **Q3.** How well can MAGELLAN's estimation generalize to predict the agent's competence on unseen goals?

• **Q4.** When these new unseen goals are introduced throughout training, can MAGELLAN leverage its generalization abilities to integrate new goals into the curriculum seamlessly?

We show that MAGELLAN 1) accurately and efficiently approximates LP, 2) allows an LLM agent to master all goals from Little-Zoo while prior methods fail when not provided extensive expert knowledge, and 3) generalizes its LP estimation to never-seen goals, enabling faster adaptation to evolving goal spaces. Moreover, we show MAGELLAN learns to cluster goals and achieves results comparable to an LP estimator with expert-defined groups.

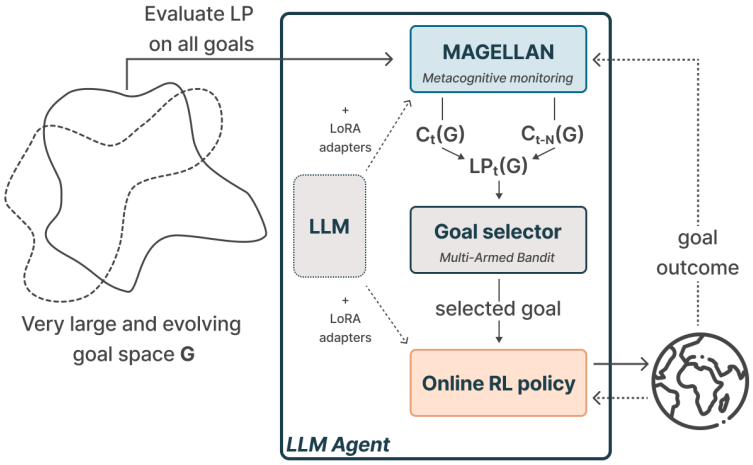


Figure 5.6: **Navigating large goal spaces with MAGELLAN**: During training, our LLM agent uses MAGELLAN to estimate its past and current competence to compute absolute LP (ALP) on each goal. Given the per-goal ALP, the LLM agent's goal selector chooses the next goal to practice proportionally to their ALP. The LLM agent then performs a trajectory to achieve this goal and the outcome is used to update both the LLM agent with online RL and MAGELLAN's competence estimation.

5.2.1 Related Work

Goal selection in autotelic agents

Autotelic agents exploring vast goal spaces face a critical challenge: they must prioritize which goals to pursue to efficiently develop general competence (Colas et al., 2022b). The ACL community has developed various approaches to address this challenge (Portelas et al., 2020b), leveraging different forms of intrinsic motivation: pursuing goals of intermediate difficulty (Florensa et al., 2018; Racaniere et al., 2020; Castanet et al., 2023; Rutherford et al., 2024), seeking novelty or uncertainty (Warde-Farley et al., 2019; Pong et al., 2020; Pitis et al., 2020), maximizing regret w.r.t. an optimal policy's performance — mostly used in Unsupervised Environment Design (UED) methods (Dennis et al., 2020; Jiang et al., 2021a; Parker-Holder et al., 2022) — or maximizing LP (Stout & Barto, 2010; Matiisen et al., 2017; Fournier et al., 2018; Portelas et al., 2020a; Colas et al., 2019; Kanitscheider et al., 2021; Kovač et al., 2023; Zhang et al., 2024b). ACL methods also differ in their use of the goal space: while some methods assume access to all goals in advance (e.g., (Matiisen et al., 2017; Kanitscheider et al., 2021; Portelas et al., 2020a)), others generate goals (e.g., (Florensa et al., 2018; Castanet et al., 2023; Dennis et al., 2020)). Another important difference lies in the final objective one seeks. While regret-based approaches seek to produce robust agents that can handle new, unseen goals, focusing on the worst-case performance (Rutherford et al., 2024), other methods seek to obtain agents maximizing their performance, either over all the goals or over a pre-defined subset of goals (Klink et al., 2020, 2024). When considering the objective of performance maximization over the complete goal space, LP-based methods have proven particularly robust, especially upon a limited training budget in which it is not possible to learn all goals (Lopes & Oudeyer, 2012). They adapt to the agent's capabilities *without requiring environment knowledge* and avoid common pitfalls like getting stuck on goals where progress plateaus or chasing uncontrollable novelty (see benchmark from Appendix H for empirical comparisons). The key challenge with LP approaches lies in efficiently estimating progress over large goal spaces, which is the focus of our work.

Computing LP over goals

LP measures the expected future improvement in achieving a goal through practice (Oudeyer & Kaplan, 2007). Since future progress cannot be directly measured, most approaches use past progress as a proxy, with the recent exception of Kumar et al. (2024)'s Bayesian prediction model. The most direct approach to estimate LP is to regularly reevaluate the agent's competence for each goal (Kanitscheider et al., 2021; Zhang et al., 2024b), which accurately captures *competence transfer* — the phenomenon where practicing one goal affects performance on other goals. However, this becomes computationally prohibitive for large discrete goal spaces and is just impossible for continuous ones. Une way to address this is to only rely on online estimations, where a goal's estimated competence is only updated when this goal is practiced. Online estimations nonetheless fail to capture competence transfer, and existing methods addressed this by grouping goals with similar competence together. For continuous spaces, approaches either learn to partition the space directly when dimensionality is low (Oudeyer & Kaplan, 2007; Baranes

& Oudeyer, 2013; Portelas et al., 2020a), or first embed high-dimensional goals into a lower-dimensional space before partitioning (Laversanne-Finot et al., 2018; Kovač et al., 2023). For discrete spaces, methods typically rely on expert-defined groupings (Stout & Barto, 2010; Matiisen et al., 2017). However, these grouping approaches are inherently brittle: they assume no transfer between groups while potentially masking competence variations within groups. This limitation is particularly acute for high-dimensional structured spaces like natural language, where competence transfer naturally occurs between semantically similar goals regardless of predefined groupings. Instead, MAGELLAN leverages an LLM's semantic understanding to dynamically model competence transfer between goals, enabling efficient and adaptive LP estimation without requiring predefined groupings or exhaustive evaluation.

5.2.2 Methods

We now detail how MAGELLAN learns a metacognitive module that estimates and generalizes an agent's LP over language goal spaces. We then explain classic LP baselines against which MAGELLAN is compared. Finally, we introduce the Little-Zoo environment, specifically designed to study commonsense-based generalization abilities of LLM agents when facing large language goal space.

Problem statement

Let $\mathcal{M} = (S, A, \mathcal{T}, R)$ be an MDP, with S a set of states, $\mathcal{T}$ the transition function, A the action space and R the reward function. Let G be a goal space and Π the policy space. We define a competence function $C_{\mathcal{M},\pi} : G \to \mathbb{R}$ that indicates the competence of a policy $\pi \in \Pi$ for a goal in $\mathcal{M}$.[1] The final aim is to find the optimal policy π^* that maximizes

$$J_{\mathcal{M}}(\pi) = \mathbb{E}_{g\sim\mathcal{U}(G)}[C_{\mathcal{M},\pi}(g)],$$

where $U(X)$ is the uniform distribution over a set X.

In this work, we focus on episodic online goal-conditioned RL with sparse and binary rewards, defined on a goal-augmented MDP $(S, A, \mathcal{T}, G, R)$, with $R : S \times G \to \{0; 1\}$ a binary success function indicating whether a state s satisfies a goal g. Here, we define $G = \{S_0 \times I | S_0 \subseteq S\}$, with I an instruction space and S_0 the set of initial states. We consider a textual environment where a prompting function $\phi : S \times I \to \mathcal{V}^K$ is given to transform any pair (state, instruction) into a textual prompt of K tokens in a given vocabulary $\mathcal{V}$. Thus, from the agent side, the policy π selects any action $a_h \in A$ by sampling from a categorical distribution $\pi(.|\phi(s_h, i))$ at any step h of the episode.

For our competence function we use the success probability $\mathbb{P}_\pi(s_0, i)$ defined as the probability for π, starting from s_0, to fulfill i within H steps: $\mathbb{P}_\pi(s_0, i) = \mathbb{E}_{\tau\sim\pi(\tau|\phi(s_0,i))}[r_{\tau,i}]$, with $r_{\tau,i} = \mathbb{1}(\exists s_h \in \tau, R(s_h, i) = 1)$ the goal outcome of episode τ for instruction i, $\mathbb{1}$ the indicator function and $\pi(\tau|\phi(s_0, i))$ the distribution of episodes of H steps induced by π

[1] In all generality a competence function does not assume the goal to be inside the MDP. For instance, g could ask for a maximum number of steps.

to fulfilling i from s_0. In this setting our objective becomes:

$$J(\pi) = \mathbb{E}_{s_0 \sim \mathcal{U}(S_0), i \sim \mathcal{U}(I)} \left[\mathbb{P}_\pi(s_0, i) \right].$$

However, given the possibly huge number of goals (s_0, i), the direct maximization of the problem becomes particularly inefficient. Our aim is to leverage transfer of competence between goals and focus during training on the ones maximizing LP. We denote as π^t the policy obtained after t episodes using an RL algorithm, with Γ^t the set of all trajectories collected during training using $\{\pi^{k-1} | k \in [|1, t|]\}$. The goal of each episode is sampled using a task selector η_G that selects a goal based on collected trajectories $\eta_G(\Gamma^t) = g$. Given a budget of T training episodes, we thus consider the problem of approaching the optimal selector $\eta_G^* = \arg\max_{\eta_G} J(\pi^T)$. In particular, we build on prior work to construct η_G on a proxy of the LP at each training episode t. We define the LP for any goal $g = (s_0, i)$ as the improvement of the policy at episode t after k episodes on goal g: $LP^k_{\pi^t}(g) = \mathbb{P}_{\pi^{t+k}}(g) - \mathbb{P}_{\pi^t}(g)$. As highlighted in Section 5.2.1, since computing the future competence on all goals is intractable, prior approaches approximate future progress with past progress ($LP^k_{\pi^t}(g) \approx \mathbb{P}_{\pi^t}(g) - \mathbb{P}_{\pi^{t-k}}(g)$). Nonetheless, accurately estimating past and current competence remains a challenge in large discrete goal spaces.

Metacognitive generalization of learning progress in language model agents

With MAGELLAN, we propose to learn estimators of the current and past policy's competence for any goal. As opposed to prior works, which either consider all goals independently or use goal groupings, we argue that learning goal-conditioned estimators would allow generalization between similar goals without defining any clear group. We propose to leverage the LLM used by our agent to learn the parameters θ_t of a competence estimator $C_{\theta_t}(g)$ for a policy π_t on a goal g. We compute $C_{\theta_t}(g)$ by giving g in the LLM's prompt, which produces a latent representation on top of its final decoder block for the last token. We use a Multi-Layer Perceptron (MLP) to output the estimated competence based on this representation. We train both the LLM and the MLP, leveraging the LLM's ability to project goals into a latent space where semantically similar goals are close. By updating the estimated competence of one goal, this allows MAGELLAN to also update close goals.

In practice, we maintain a buffer $\mathcal{D}_t$ which contains, for the M most recent training episodes at t (i.e., $\tau^{t-M} \dots \tau^t$), their corresponding pair of goal and outcome (i.e., $(g = (s_0, i), r_{\tau,i})$ for each τ). As this work focuses on success probability (i.e., we want $C_{\theta_t}(g) \approx \mathbb{P}_{\pi^t}(g)$), we train C_{θ_t} using stochastic gradient descent to minimize the binary cross-entropy: $\mathcal{L}(\theta_t) = \mathbb{E}_{(g,r)\sim\mathcal{D}_t} \left[BCE(r, C_{\theta_t}(g)) \right]$.

We maintain another buffer $\mathcal{B}_t$ storing the last N weights of our competence estimator: $\mathcal{B}_t = [\theta_{t-N}, \theta_{t+1-N}, \dots, \theta_t]$. Weights are added to the buffer every time the competence estimator is updated, enabling access to estimations of the policy's competence from time t to $t - N$. Using this information, we estimate the absolute LP (ALP) (Baranes & Oudeyer, 2013; Kanitscheider et al., 2021), tracking both progress and forgetting, as follows:

$$\hat{ALP}_{\pi_t}(g) = |C_{\theta_t}(g) - C_{\theta_{t-N}}(g)|. \tag{5.4}$$

This ALP estimation can subsequently be used to structure the agent's curriculum. We apply the multi-armed bandit goal selection scheme introduced by Lopes & Oudeyer (2012) where each arm is a goal, and its utility is MAGELLAN's estimate of this goal's ALP. Goals are then sampled proportionally to their estimated ALP (also called Boltzmann sampling) with an annealing exploration probability ϵ (ϵ decreasing from 1 to 0.2). In practice, we train two separate versions of the same initial LLM (using LoRA adapters (Hu et al., 2022)): one for the policy and one for MAGELLAN's current competence estimator. We show in Appendix E.4.1 ablations on architectural choices indicating that 1) keeping the LLM frozen leads to poor results, highlighting the need for a dynamic representation space (see also Figure 5.11), and 2) training separate LoRA adapters for the policy and MAGELLAN leads to more stability.

Classic ALP baselines

Following the literature on ALP in Section 5.2.1, we implement classic approaches, focusing on two dimensions. First, we consider Online (Baranes & Oudeyer, 2013; Matiisen et al., 2017) vs Evaluation-based ALP (Kanitscheider et al., 2021; Zhang et al., 2024b) estimation. Then, we consider directly using the goal space (Portelas et al., 2020a; Kanitscheider et al., 2021) or using expert-defined groups of goals with assumed competence transfer (Stout & Barto, 2010; Colas et al., 2019). The latter requires extensive expert knowledge (EK) given the absence of automatic approaches for discrete goal spaces. As expert-defined groups are created beforehand, no competence transfer is assumed across groups, which is likely to happen in spaces like natural language, where transfer occurs between semantically close goals regardless of groups.

We thus implement four baselines (see all details in Appendix E.3.3):

- **Eval-ALP**: Every N episodes, training stops and the agent is separately evaluated on each goal to obtain a competence estimate. The per-goal ALP is the absolute difference between estimates at t and $t - N$. The same goal selection scheme as in MAGELLAN is used according to the per-goal ALP estimations.
- **EK-Eval-ALP**: Every N episodes, training stops and the agent is evaluated on multiple goals randomly sampled in each expert-defined group to obtain a per-group averaged competence. The per-group ALP is computed using the absolute difference between the competence at t and $t - N$. The goal selection process first selects a group using the same selection scheme as MAGELLAN for goals. Given a selected group, a goal from this group is randomly selected.
- **Online-ALP**: At each goal practiced, the observed policy's competence is added to a buffer of $2M$ past experiences for this goal. The ALP is computed using the absolute difference between the average competence over the last and first M experiences in the buffer. The same goal selection scheme as MAGELLAN and Eval-ALP is used.
- **EK-Online-ALP**: At each goal practiced, the observed policy's competence is added to a buffer of $2M$ past experiences for the goal's expert-defined group. The per-group ALP is computed using the absolute difference between the average competence over the last and first M experiences in the group's buffer. The same goal selection scheme as EK-Eval-ALP is used.

We summarize the four methods above and MAGELLAN in Table 5.1 based on their *Efficiency* (i.e., computational cost introduced by additional evaluations), *Competence Transfer tracking* and *no Expert Knowledge requirement*. We consider a method's efficiency as "high" if it does not require any additional evaluation (i.e., it only uses the performance observed on goals sampled), and as "low" otherwise. We evaluate the competence transfer tracking using the following criteria:

- absence of +: the estimated competence is updated only on sampled goals.
- +: the estimated competence is updated on a predefined group the sampled goal belongs to.
- ++: the estimated competence is updated on a dynamically learned group the sampled goal belongs to.
- +++: the estimated competence is updated on all goals.

We provide in Appendix E.2 the same table for all prior works covered in Section 5.2.1.

	Eff.	**Transf.**	**No EK**
EK-Eval-ALP	low	+	×
Eval-ALP	low	+++	✓
EK-Online-ALP	high	+	×
Online-ALP	high		✓
MAGELLAN	high	+++	✓

Table 5.1: Comparison of ALP estimation methods. We use the following dimensions: computational Efficiency, competence Transfer tracking, and required Expert Knowledge.

The Little-Zoo environment as a testbed

Evaluating commonsense-based generalization of LLM agents in textual environments requires several key properties. The environment must be fully text-based, with all observations, actions, and goals expressed in natural language. It should feature a diverse set of goals with varying difficulty levels, enabling the assessment of the agent's ability to learn and generalize complex skills. Additionally, these goals should be organized into hidden families based on commonsense knowledge, allowing for targeted evaluation of generalization capabilities.

Existing environments for LLM agents do not have such requirements. Creative environments such as Minecraft (Johnson et al., 2016) or Crafter (Hafner, 2022) rely on image-based observations and require an image captioner to use LLM agents. Textual environments such as BabyAI-text focus on navigation skills without any commonsense-based generalization. Although WordCraft (Jiang et al., 2020) incorporates commonsense-based goals, no relationship between goals exists, limiting the analysis of the agent's generalization abilities. To address these gaps, we introduce Little-Zoo, a novel environment explicitly designed to meet these criteria.

Built upon the Playground environment (Colas et al., 2020), Little-Zoo is fully text-based, with observations, goals, and actions expressed in natural language. It features

objects that can be combined together and are grouped into the following hidden categories: furniture (which cannot be combined), plants, herbivores, and carnivores. Given the set of all objects and a set of instructions, Little-Zoo's goal space is the combination of all possible instructions and scene initializations. The feasibility of a goal thus depends on the objects available, making most combinations infeasible and not trivial to detect (see Figure 5.7 and Appendix E.1.3). For instance, these are respectively feasible and infeasible goals: *"Goal: Grow deer. You see: baby deer, bookshelf, water, tomato seed."*; *"Goal: Grow deer. You see: baby deer, bookshelf, baby lion, tomato seed."* (water is missing in the last one). Instructions are hierarchically structured, ranging from simple grasping tasks to more complex sequences involving object interactions (e.g., "growing" animals). The complete goal space contains approximately 20 million combinations. In our experiments, we subsample goals with the following proportions: 80% of the goals are impossible 16% involve grasping, 3.2% involve growing plants, 0.7% involve herbivores, and 0.1% involve carnivores. These proportions correspond to proportions in the complete goal space (see Appendix E.1.4).

Little-Zoo is a deterministic, fully-observable and episodic environment: the agent begins an episode by standing on nothing, with full visibility of the whole scene. The action space consists of 8 actions, including movement to objects, grasping, and releasing objects. Ubservations include the objects in the scene, the ones in the agent's inventory, as well as the object the agent is standing on. See Appendix E.1 for details on Little-Zoo.

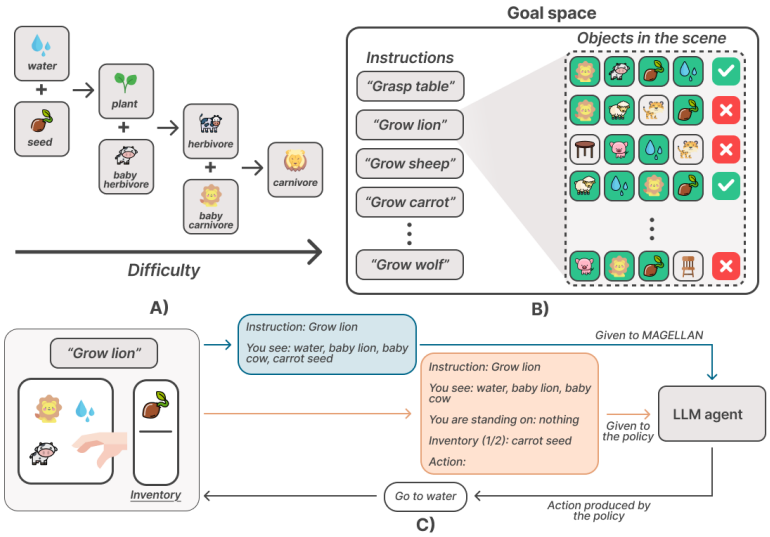


Figure 5.7: A) Little-Zoo's tech tree. B) Little-Zoo's goal space is composed of all the possible combinations between instructions and objects that can be in the scene. Most object configurations make an instruction infeasible (e.g., "grow lion" is impossible with the second configuration, as water, needed to obtain plants, is missing). C) Little-Zoo provides a textual description that is given in our LLM agent's prompt.

5.2.3 Experiments

We provide empirical answers to our scientific questions using experiments with 8 different random seeds in the Little-Zoo environment. For our LLM agent, we use SAC-GLAM (from Section 5.1) to fine-tune Flan-T5 248M (Raffel et al., 2020), as in SAC-GLAM's experiments. We compare MAGELLAN to the classic approaches presented in 5.2.2. For methods accessing external expert knowledge, we use Little-Zoo's hidden goal families (grasp any object, grow plant, grow herbivore, grow carnivore), but add only possible goals in these groups. Indeed, these baselines are based on groups predefined in advance by human experts with a strong assumption: the goals within a group share the same learning dynamics and therefore the agent's competence is the same over all goals in the group. *If impossible goals were included, these groups would lose their relevance.* Moreover, because of the large number of impossible goals, the average competence within each group will always be very close to 0. There will be no progress niche that the method can use to generate a curriculum, and performance will likely be close to the random baseline. We thus provide an additional group containing all impossible goals. In all our experiments, we use the success rate (i.e., average outcome over multiple trials for a goal), noted SR, as the observed competence.

We first study how MAGELLAN's competence estimation compares to baselines (**Q1**). Then, we study how the different methods (except Eval-ALP and EK-Eval-ALP, which are too costly to run, while EK-Online-ALP provides a good estimation of their performance) compare when scaffolding the LLM agent's curriculum (**Q2**). We show how these competence estimators also generalize to goals not seen during training (**Q3**). Finally, we study how all methods adapt as the goal space evolves (**Q4**) by replacing all goals at different points throughout training.

How well does MAGELLAN estimate competence (Q1)

To assess the ALP methods' ability to efficiently estimate competence, we designed an experimental setup in which our LLM agent was trained for 50k episodes on the Little-Zoo environment with varying goal space sizes (25k, 50k, 100k), while keeping the same repartition between goal types. As computing the expected ALP to train this agent is intractable, one could argue that Eval-ALP is the best approximation. However, it remains too computationally costly to run, even when performing only 50k training episodes with 25k goals. We thus chose to sample goals according to EK-Eval-ALP's estimations. To obtain an accurate estimate, we perform 2048 per-group evaluations every 1000 episodes. The per-group competence evaluated by EK-Eval-ALP is consequently our competence reference, and we compare the other methods against it. For Eval-ALP, we consider it to have zero error, and its computational cost can be estimated without running it. For MAGELLAN and Online-ALP, we average the per-goal competence over groups to compute the error w.r.t. EK-Eval-ALP. Figure 5.8 shows the average error on competence throughout training along and the cost of competence evaluation (i.e., the total number of episodes used only to evaluate competence).

As indicated in Table 5.1, MAGELLAN performs on a par with Eval-ALP, showing that it accurately estimates the transfer of competence while using online estimations. We

also observe similar competence errors to methods using expert-defined groups, hinting at MAGELLAN's abilities at learning semantical relationships between goals. We provide a more in-depth analysis of such relationships in Appendix E.4.4. Finally, MAGELLAN achieves this performance without an estimation cost.

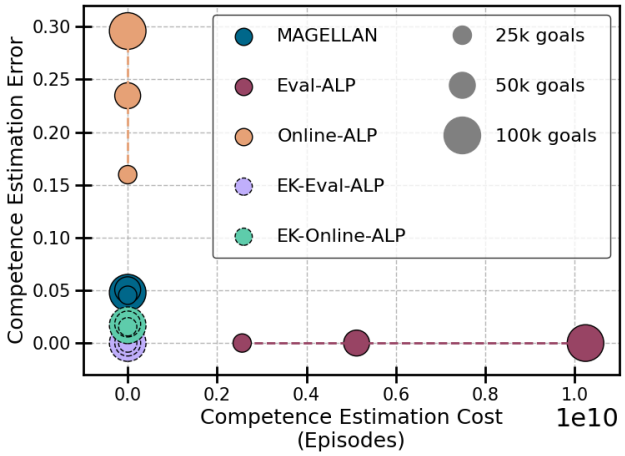


Figure 5.8: Scaling of competence estimation error and competence estimation cost (i.e., total number of additional evaluation episodes) when increasing the goal space size.

To demonstrate that our method generalizes beyond the Little-Zoo environment, we conducted an additional experiment using goals derived from the UpenR1-Math-220k dataset (Hugging Face, 2025). In this setup, instead of training an RL agent, we simulate the learning of an agent that progressively acquires skills in three categories: *Algebra*, then *Geometry*, and finally *Number Theory*. We compare the competence estimation from MAGELLAN and Online-ALP to the underlying true competence. Figure 5.9 shows that MAGELLAN accurately tracks the agent's competence and distinguishes between the different mathematical categories, clearly outperforming Online-ALP. Additional results, including experiments in the BabyAI-Text environment and an ablation study on the impact of LLM size on MAGELLAN's estimations, are provided in Appendix E.4.2.

Training an LLM agent with MAGELLAN (Q2)

As demonstrated in 5.2.3, MAGELLAN provides a superior competence estimation than Online-ALP. We further investigate whether this improvement translates into a better curriculum and improved overall goal mastery. We train our LLM agent on the goal space of Little-Zoo with 25k goals for 500k episodes using four methods: MAGELLAN, Online-ALP, EK-Online-ALP and "Uniform", where goals are sampled uniformly. We do not report EK-Eval-ALP, as we report EK-Online-ALP, which produces similar competence estimation with no cost. We report the agent's SR every 5000 training episodes by evaluating it on 64 goals uniformly sampled for each category. Figure 5.10 shows the evolution of SR averaged over all categories. Our results show that MAGELLAN is the only method without expert-defined grouping to obtain an SR of at least 90% in all categories. It also masters the categories significantly faster than baselines. Despite

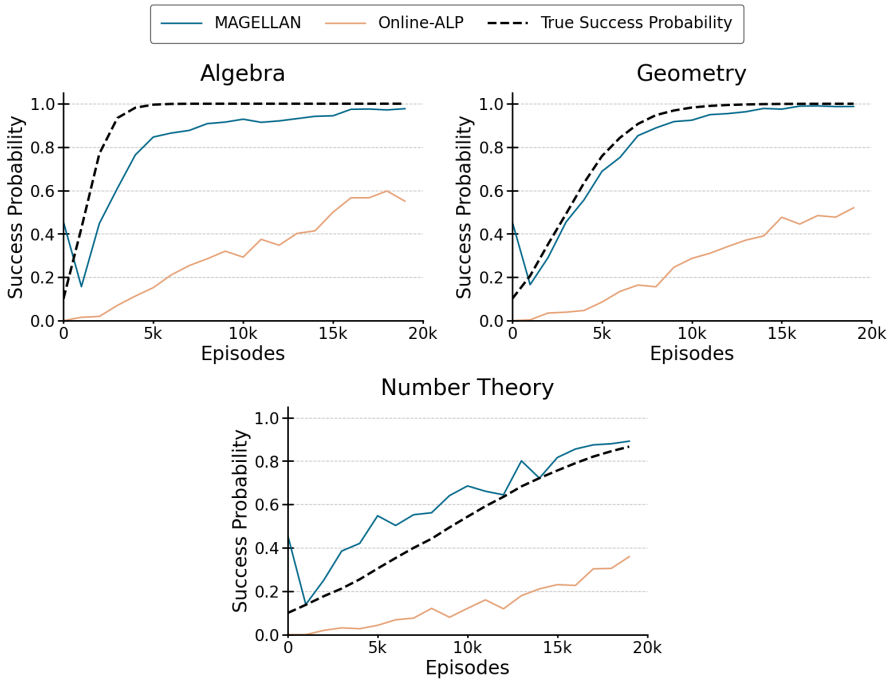


Figure 5.9: Competence estimation on UpenR1-Math-220k. MAGELLAN (blue) accurately tracks competence across Algebra, Geometry, and Number Theory, closely matching true success probabilities and outperforming Online-ALP (orange).

MAGELLAN's similar competence estimation as EK-Online-ALP, the latter learns faster by leveraging the expert-defined groups to better explore. This is because MAGELLAN explores by uniformly sampling goals whereas EK-Online-ALP uniformly samples groups, easily discarding impossible goals.

MAGELLAN's generalization abilities (Q3)

We move further and study the generalization abilities of both our LLM agent and competence estimators. While Section 5.2.3 evaluates each policy on 64 goals per category that belong to the training goal space, this section reports evaluation on a held-out test set composed of unseen goals. As in the previous section, evaluations were performed every 5000 training episodes. However, instead of reporting the policy's observed competence (SR) during evaluation, we report the difference between the observed competence and the competence estimated by the policy's ALP method. We show in Table 5.2 the average error over training.

By only tracking competence on goals practiced by the policy (i.e., the ones from the training goal space), Online-ALP cannot provide any estimation for new unseen goals and uses its default competence of 0. This leads to the largest error among methods except for "Grow carnivore" goals as the policies trained with Online-ALP never mastered these goals (see Figure 5.10). MAGELLAN successfully generalizes its competence estimation and obtains a small error. Finally, EK-Online-ALP produces accurate estimations based on expert knowledge of which group each test goal belongs to. Appendix E.4.4 provides

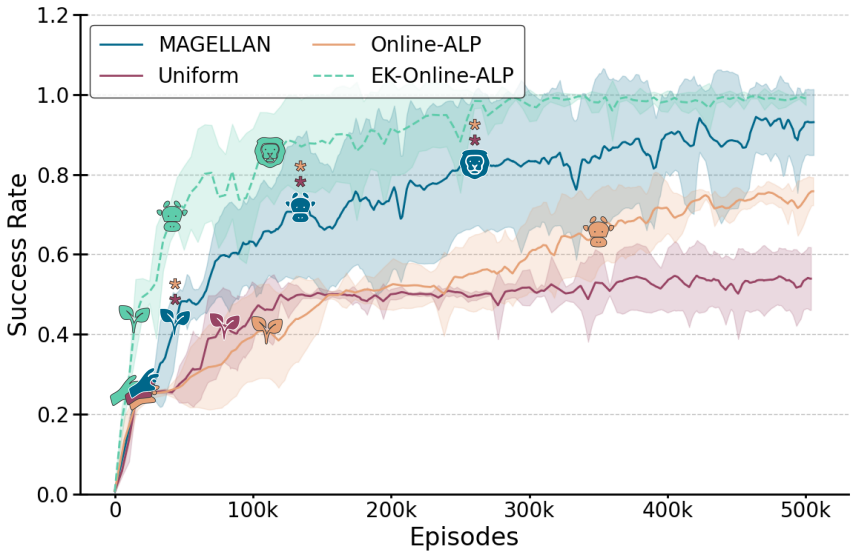


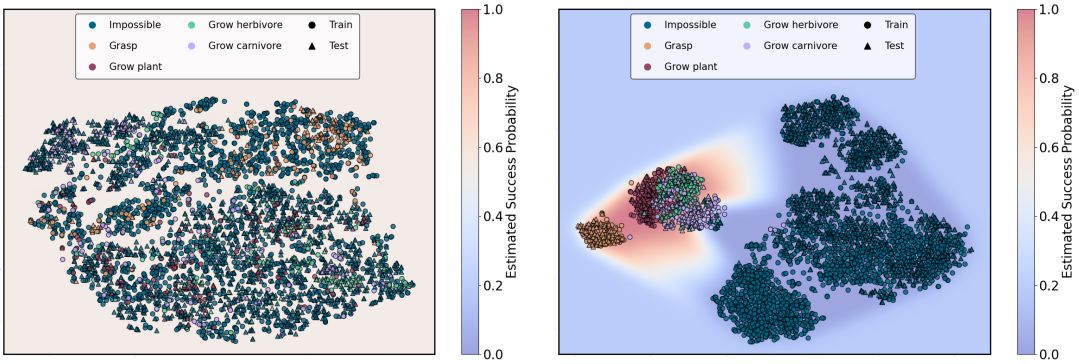
Figure 5.10: Evolution of the observed competence (SR) when evaluating policies on 64 training goals per category every 5000 episodes. We report the average SR over evaluated goals along with standard deviation (8 seeds). Icons indicate the average time step at which a method mastered a goal (i.e., SR $> 90\%$). We add stars to MAGELLAN, denoting significantly earlier mastery of a category compared to the method with the star's color (p-value $< 8 \times 10^{-4}$). The dotted line (EK-Online-ALP) indicates that the method relies on expert knowledge.

detailed results indicating our LLM agents do not perfectly generalize, which explains MAGELLAN and EK-Eval-ALP estimation error.

Table 5.2: We evaluate the policies trained with each ALP method on a held-out test set. We show the difference between the observed and predicted competence. Online-ALP's performance on "Grow carnivore" is simply explained by the fact that its policies never mastered this goal category.

Categories	**MAGELLAN** (Mean ± Std)	**Online-ALP** (Mean ± Std)	**EK-Online-ALP** (Mean ± Std)
Grasp	**0.01 ± 0.00**	0.98 ± 0.00	0.01 ± 0.00
Grow plant	**0.05 ± 0.03**	0.78 ± 0.07	0.03 ± 0.01
Grow herbivore	**0.08 ± 0.05**	0.34 ± 0.18	0.06 ± 0.02
Grow carnivore	0.30 ± 0.16	**0.00 ± 0.00**	0.28 ± 0.08
Mean	**0.11 ± 0.06**	0.53 ± 0.06	0.09 ± 0.03

We further investigate MAGELLAN's generalization abilities by projecting train (Q2) and test (Q3) goals from a single seed into the LLM embedding space MAGELLAN used. The embedding space is plotted both before and after training (Figure 5.11) with projections obtained via t-SNE (Maaten & Hinton, 2008). The initial embedding space lacks any discernible structure for goal classification. Post-training, the space exhibits significant restructuring, with similar goals accurately clustered. A small subset of "Grow carnivore" goals are misclassified as impossible, likely due to incomplete mastery of this category by the LLM agent.



(a) (b)

Figure 5.11: MAGELLAN's LLM embedding space displayed using t-SNE with goals used in Q2 (Train) and Q3 (Test), along with the estimated success probability and linear interpolation between goals. We show the embedding space for a single seed (a) before training and (b) at the end of the 500k training steps. We see that impossible goals have been left aside, and that the other goals with a high estimated success probability are clustered consistently.

Furthermore, the spatial arrangement of goals correlates with estimated success rates: newly learned goals tend to lie near the boundary with impossible goals. Additionally, goals from the test set are well clustered, demonstrating strong generalization. We show in Appendix E.4.4 that MAGELLAN also makes different clusters for impossible goals based on the infeasibility reason, hinting that metacognitive monitoring abilities help capture environment dynamics.

MAGELLAN's adaptation to evolving goal spaces (Q4)

Finally, we investigate how each ALP method can adapt when the goal space evolves. For this, we isolate the training of one seed using MAGELLAN in Section E.4.3. Every 50k episodes over the 500k training episodes, we stop training, replace the training goals with the ones in our held-out test set, and start four trainings (with 8 seeds each) with this new goal space: one with each ALP method for 50k steps. We expect MAGELLAN to quickly adapt to new goals leveraging semantical relationships between the new and old goals. Online-ALP starts with a competence estimation of 0 on the new goals as in Section 5.2.3. For EK-Online-ALP, up to the episode where we change the goal space, we make it track the policy's competence in parallel to MAGELLAN. It thus starts with a per-group competence estimation when the goals are replaced along with the information of which expert-defined group each new goal belongs to. We study all adaptation training in Appendix E.4.5 but isolate and study in Figure 5.12 two of them chosen as representative of the scenarios encountered:

- **Scenario zero LP (Figure 5.12a):** The agent has mastered the "Grasp" and "Grow plants" goals and has 0 ALP across all goals. In this scenario, all ALP estimations are equivalent. EK-Online-LP manages to discover new ALP niches faster as all impossible goals are in the same group.

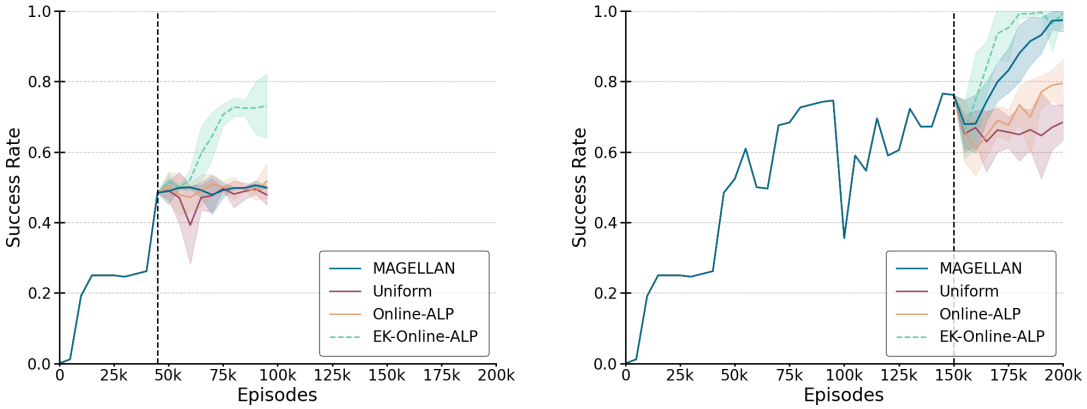


Figure 5.12: **Adaptation tests**: Using a single's seed training of 500k episodes, we stop and replace all goals with unseen ones every 50k episodes. We then resume training and sample goals using each method for 50k training episodes. We show two isolated and representative points of goal replacement: (a) there is no ALP on any goal (after 50k training episodes), and (b) some goals (here, "Grow carnivores" after 150k training episodes) have a high ALP. We report the evolution of SR when evaluating the policies on 64 goals per category from the new training set every 5000 episodes. Results show the average competence over evaluated goals along with standard deviation (8 seeds).

- **Scenario high LP (Figure 5.12b):** The agent is getting a high ALP as it is learning some "Grow carnivores" goals. Here, MAGELLAN outperforms baselines by generalizing its ALP estimation and continuing training on these goals. MAGELLAN even gets on par performance with EK-Online-LP.

5.2.4 Conclusion

In this work, we introduced MAGELLAN, a metacognitive module designed to efficiently estimate an LLM agent's LP across large language-defined goal spaces. By leveraging the LLM within the agent, MAGELLAN accurately and efficiently estimates LP by learning semantic relationships between goals. This approach fundamentally differs from prior methods, which either struggle to scale to large goal spaces or rely on expert-defined goal categories. Using MAGELLAN's LP estimations, the LLM agent can effectively structure its curriculum, enabling it to master all goals within an extensive goal space. In contrast, previous methods achieve only partial mastery in the absence of expert-defined categories. Moreover, MAGELLAN's ability to capture semantic relationships between goals allows it to assess the agent's competence on unseen goals and rapidly adapt as the goal space evolves.

Additionally, our analysis of MAGELLAN shapes the LLM's embedding space (including how impossible goals are clustered) hints that estimating helps capture the environment's dynamics. This evidence indicates that acquiring metacognitive monitoring abilities requires one to grasp how the environment functions. We argue that this constitutes another form of grounding, which might be complementary to functional grounding

(and how it also permits capturing environmental dynamics as evidenced by Section 3.3). While most experiments in this work used separate adapters for functional grounding (i.e., learning the policy) and MAGELLAN, we believe that future work could further investigate how combining the two using a single set of adapters (as done in architecture A in Appendix E.4.1) affects the LLM's internal representations.

While our study provides an in-depth analysis of MAGELLAN's capabilities in a controlled textual environment, the method itself is highly generalizable. It offers a goal prioritization strategy applicable to any learner operating in a large, high-dimensional, structured, and discrete goal space. Notably, goal spaces involving code appear particularly promising, given the proficiency of current LLMs in code generation (Wang et al., 2023a; Pourcel et al., 2024b). Additionally, traditional automatic curriculum learning settings—where goals are not language-defined—could also benefit from MAGELLAN's ability to uncover relationships between goals. Beyond artificial learners, MAGELLAN may also prove valuable for human learning, particularly in educational domains where learners must navigate a large space of language-defined problems, such as mathematical word problems (Doroudi et al., 2019). Classical LP measures have already been shown to enhance personalized curricula in educational technologies (Clement et al., 2015), suggesting MAGELLAN's potential impact in this area.

Finally, while estimating one's functional competence is necessary to prioritize what to learn next, it is also valuable for identifying which tasks are currently beyond reach. This ability allows one to find when external help is necessary. In the next contribution, we study how learning metacognitive monitoring abilities can help LLMs identify when their functional competence falls short and call for external assistance, for instance, from another larger or more specialized LLM.

5.3 When goals are beyond reach: Metacognitive monitoring guides autonomous discovery of frugal assistance-seeking in LLMs

In the previous section, we introduced MAGELLAN, an approach designed to augment LLMs with metacognitive abilities—specifically, the capacity to evaluate their own functional competence across a wide range of tasks. We demonstrated that this self-monitoring mechanism could be used to guide exploration over large goal spaces by prioritizing tasks with high estimated Learning Progress. In particular, we showed that MAGELLAN can capture semantic relationships between language-specified goals, allowing it to discover competence transfer but also generalize its estimations to novel, unseen goals.

More broadly, enhancing LLMs with metacognitive capabilities has been identified as a key challenge for improving the trustworthiness and interpretability of these models (Johnson, 2022; Johnson et al., 2025). In this section, we investigate how such metacognitive abilities can be leveraged to trigger external assistance when the model's own capabilities are insufficient. While previous contributions have focused primarily on improving LLMs' functional competence through interaction, it is equally critical that models learn to recognize their own limitations—and to seek or rely on external support in real-world settings where functional competence may be partial or underdeveloped. This ability forms

a crucial part of a broader learning loop: requesting help when needed, then internalizing the knowledge or skills acquired through that assistance. As such, while the present work may not directly extend functional grounding, it constitutes a necessary precursor. We argue that the capacity for help-seeking is foundational for building LLMs capable of becoming functionally grounded in open, interactive environments—particularly when these environments include human users as integral participants in the learning process.

Augmenting LLMs with external assistance and, in particular, what has been named "tools", has become a well-established practice (Nakano et al., 2022; Yao et al., 2022; Schick et al., 2023; Patil et al., 2023; Ge et al., 2023). These augmentations range from calculators (Schick et al., 2023; Kadlčík et al., 2023) and retrieval systems (Parisi et al., 2022; Nakano et al., 2022; Yao et al., 2022) to code interpreters (Gao et al., 2023; Wang et al., 2024c), and even other LLMs (Ge et al., 2023). This shift has led to a rethinking of the role of LLMs—not as general-purpose solvers, but as assistants (often referred to as action models (Wang et al., 2025)) that must learn to orchestrate the use of external resources and integrate their outputs into coherent, human-readable responses.

This reframing introduces a new class of decision-making problems: LLMs must determine when and which external assistance to invoke. However, the optimal assistance strategy is not known in advance. Some tasks may be solvable independently by the LLM, while others may require external help. Additionally, the tools themselves may be fallible—for instance, even large or specialized LLMs can return suboptimal results. To address this, most prior approaches rely on supervised learning, fine-tuning LLMs on curated datasets containing examples of effective tool use. More recently, several works have begun exploring how RL can be used to learn assistance-seeking strategies from scratch, without requiring predefined tool-use demonstrations (Wen et al., 2024b; Chen et al., 2025; Li et al., 2025; Feng et al., 2025).

While both RL and more conventional supervised learning approaches have shown promise, an important dimension of the assistance-seeking problem remains largely understudied: external assistance comes at a cost. This cost may take the form of increased latency in the LLM's response, financial charges for calling APIs, or computational overhead. Although early work on tool use—such as Schick et al. (2023)—acknowledged this issue, it has received limited attention since. A recent exception is Min et al. (2025), which introduced a first approach to this multi-objective problem: maximizing task performance while minimizing assistance costs. Their method involves a multi-stage learning pipeline: (1) an estimator of LLM performance is trained using interaction data between the LLM and the task space; (2) a separate model is trained to simulate the outputs of both the LLM and its assistance sources; and (3) given a predefined cost budget, Dynamic Programming is used to derive the optimal assistance strategy. While effective, this method is computationally intensive and requires extensive data collection and training across multiple stages.

In this work, we propose a fully online approach based on multi-objective contextual multi-armed bandits. Given a task, we frame the decision of whether to keep the task with the LLM or delegate it to external assistance as the selection of an arm. Given a task, we consider the dual objective of maximizing the answer's performance R while minimizing its cost C, and we adopt scalarization—i.e., combining the two objectives into a single weighted sum $U = \beta R + (1-\beta)C$ that our approach aims to maximize. Crucially,

our method naturally adapts to any specified user-specified budget by treating the budget as the scalarization weight that balances the two objectives. A central challenge of this approach lies in efficiently estimating the performance and cost associated with each option (i.e., the LLM and all available assistance sources), using as few interactions as possible. To address this, we draw inspiration from MAGELLAN and leverage the LLM itself to learn these estimations. As demonstrated in Section 5.2, such LLM-based estimators can generalize across semantically similar tasks, enabling faster and more sample-efficient convergence.

We first evaluate our method on a set of carefully designed math problems with calculator tools as assistance, for which the optimal strategy is known. This setting enables us to investigate how the strategy discovered by our method compares to the optimal one, as well as the sample efficiency of our approach (i.e., the number of interactions required to converge). We notably show that our LLM-based estimation of performance and cost reaches similar or even better performance than a classic moving average approach which has access to privileged information—namely, the problem category. Finally, we demonstrate the broader applicability of our method by applying it to real-world problems faced by LLMs. In particular, we apply it to a standard question-answering benchmark: MMLU-Pro (Wang et al., 2024d). The results show that our approach is scalable to complex natural language tasks without access to any external expert knowledge.

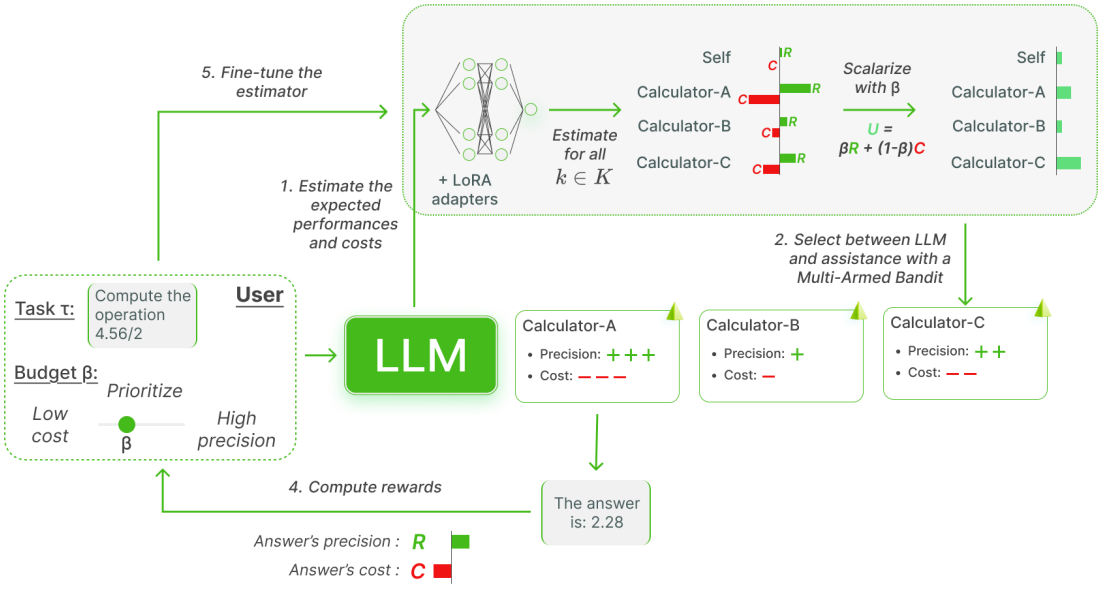


Figure 5.13: We frame the decision of whether an LLM should trigger external assistance as a multi-objective contextual multi-armed bandit problem. For each task, the LLM estimates the performance and cost of all available options. These estimates are combined into a single utility score using a user-defined scalarization weight, which specifies the trade-off between maximizing performance and minimizing assistance cost. The estimator is continuously updated through online interactions.

5.3.1 Related Work

External assistance for LLMs

Augmenting LLMs with external assistance—particularly in the form of *tools*—has been extensively explored in recent years. Much of the focus has centered on calculators to enhance mathematical reasoning (Parisi et al., 2022; Schick et al., 2023; Kadlčík et al., 2023; Hao et al., 2023b), information retrieval systems via databases or web search (Parisi et al., 2022; Nakano et al., 2022; Lazaridou et al., 2022; Yao et al., 2022; Hao et al., 2023b), and code interpreters to support multi-step reasoning (Gao et al., 2023; Wang et al., 2024c; Christianos et al., 2023; Feng et al., 2025; Gehring et al., 2025). More recently, another line of work has explored using other LLMs—typically larger or more specialized—as a form of external assistance. While these models can be accessed via API calls and integrated similarly to traditional tools, an emerging research direction directly addresses the multi-LLM setting, where a dedicated routing function selects which LLM to invoke for a given task (Yue et al., 2024; Ding et al., 2024). In this work, we propose a method whereby an LLM learns to estimate both its own performance and the performance of available assistance. This estimation guides the decision of whether to keep a task or delegate it, using a multi-armed bandit framework. Our approach is applicable to both tool-based assistance and multi-LLM scenarios (see Section 5.3.3).

Confidence estimations in LLMs

Recent work has increasingly focused on estimating the confidence of LLMs in their functional competence. In the context of question-answering, several studies have explored the relationship between token-level logits and model confidence (Mahaut et al., 2024; Zhou et al., 2023), as well as the alignment between verbalized confidence and actual correctness (Tian et al., 2023). In parallel, other approaches have proposed training LLMs to output special uncertainty tokens when they are unsure about their answers (Cohen et al., 2024; Chuang et al., 2025). Closer to our approach, Min et al. (2025) proposed training an LLM to estimate its own functional competence for navigation tasks. However, their framework separates the learning of the competence estimator from the learning of the assistance-asking policy. In contrast, we propose a fully online method inspired by MAGELLAN (introduced in Section 5.2), in which an LLM learns to estimate not only its own competence, but also the competence of each available assistance. These estimations are directly integrated into a multi-armed bandit framework to guide assistance-seeking decisions.

Multi-objective optimization of LLMs

Balancing task performance and the computational cost induced by reasoning or external assistance can be formulated as a multi-objective optimization problem. In the context of tool use, Schick et al. (2023) constructed a supervised fine-tuning dataset by retaining only tool calls that increased the likelihood of producing the correct answer beyond a threshold τ. Li et al. (2025) employed RL to train an LLM to use a code interpreter tool, introducing a fixed negative reward for tool calls that resulted in failed executions—thus penalizing only inappropriate tool use. They also enforced a maximum number of tool calls per query to avoid overuse. However, this approach has two limitations:

1) it assumes prior knowledge of the optimal number of tool calls, and 2) it does not prevent unnecessary tool use if the allowed maximum exceeds what is required. Min et al. (2025) also penalize tool calls via a negative reward, but derive this penalty through Dynamic Programming to ensure that the resulting strategy respects a pre-defined cost budget. In contrast, our approach introduces negative rewards for assistance calls (with several variants explored in our experiments) and directly balances performance and cost via scalarization. Scalarization—combining multiple objectives into a single weighted objective—has been used in multi-objective RLHF settings, where LLMs must trade off between different preference-based rewards (Rame et al., 2023; Bahlous-Boldi et al., 2025). Finally, recent research has explored reasoning as a cost-sensitive operation. For example, Aggarwal & Welleck (2025) and Arora & Zanette (2025) train LLMs via reinforcement learning to perform reasoning while minimizing the number of generated tokens, subject to either a fixed token budget or a target performance threshold.

Multi-objective contextual neural bandits

Multi-objective decision-making has been widely explored in the multi-armed bandit literature. Some methods directly optimize for Pareto efficiency using Pareto regret as an objective (e.g., (Turgay et al., 2018)), while others—closer to our approach—rely on scalarization, combining multiple objectives into a single one via weighted sums (e.g., (Drugan & Nowe, 2013; Qassimi & Rakrak, 2025)). In our case, however, the problem takes the form of a contextual bandit setting, where both the task and the scalarization weights must be jointly considered as context. Although contextual bandits have been extensively studied, including in multi-objective scenarios (Turgay et al., 2018), most classical methods operate under two key assumptions: 1) the context is represented as a fixed-dimensional vector, and 2) the utility of each arm is a linear function of the context features (Li et al., 2010; Valko et al., 2013). To move beyond these constraints, recent approaches have turned to neural approximators for modeling the utility of context-arm pairs (Zhou et al., 2020; Ban et al., 2023). Our work follows this line, where each assistance source is treated as an arm, and its expected return is estimated using a neural network. However, unlike prior work assuming structured or vectorized contexts, our tasks are natural language prompts. To handle this, we extend the approach proposed in MAGELLAN and use an LLM-based estimator capable of generalizing across semantically similar tasks.

5.3.2 Methods

In this section, we present our approach. We begin by formulating assistance-asking as a multi-objective contextual bandit problem, where both the task and the scalarization weight are treated as part of the context. We then introduce a neural estimator inspired by MAGELLAN, designed to predict the expected utility of different assistance options and guide the decision-making process accordingly.

Asking for assistance as a bandit problem

We consider a discrete space of tasks $\mathcal{T}$ that an LLM π must solve. These tasks can, for instance, come from a dataset of questions, but could also be embodied tasks, as in the remainder of this manuscript. We also assume a reward function (often called outcome-based reward) $R : \mathcal{T} \times \mathcal{A} \mapsto \mathbb{R}$ which associates a scalar performance to the output $a \in \mathcal{A}$ produced when executing π on a task $\tau \in \mathcal{T}$. In the case of embodied problems, $\mathcal{A}$ could be the set of possible trajectories in the environment. In this work, we will consider natural language questions for which an LLM is rewarded based on how close its natural language answer is compared to the expected answer. We therefore have $\mathcal{A} \subseteq \mathcal{V}^M$, with $\mathcal{V}$ a token vocabulary and M the maximum length of answers.

We then introduce a set of potential N assistance $\Phi = \{\phi_1, \phi_2, ..., \phi_N\}$. Each $\phi_i \in \Phi$ is also, as π, a function that produces an answer to a task (i.e., $\forall i \, \phi_i : \mathcal{T} \mapsto \mathcal{A}$)[2]. In this work, we assume both π and all assistance to be frozen (i.e., they do not change over time). In such a stationary case, choosing between π or any $\phi_i \in \Phi$ for a task $\tau \in \mathcal{T}$ so as to maximize R can be framed as a contextual multi-armed bandit problem with:

- $K = \Phi \cup \{\pi\}$, the set of arms.
- $\mathcal{T}$, the set of possible contexts.
- $U(\tau, k) = \mathbb{E}_{a \sim k(\tau)}[R(\tau, a)]$, the utility of arm $k \in K$ on task τ as the expected reward obtained by giving τ to k.

We consider a purely online setting where tasks are sequentially (and uniformly) sampled from $\mathcal{T}$. Given a set of collected task-outcome pairs and a new task, we look for a policy that would optimally choose which arm to give the new task to.

We now introduce a second reward function $C : \mathcal{T} \times K \mapsto \mathbb{R}^-$ which associates a per-task negative reward (i.e., a cost) to each arm. With two objectives, R and C, we now frame our problem as a multi-objective contextual bandit. We propose to integrate these two objectives in our bandit using scalarization and introduce the global utility objective $U(\tau, k, \beta) = \beta R(\tau, k(\tau)) + (1 - \beta)C(\tau, k)$ where $\beta \in [0; 1]$ balances the two objectives.

Performance and cost predictors

In this work, we consider natural language tasks as contexts for our bandit. Inspired by the recent literature on neural contextual bandits and the success of MAGELLAN, we propose to learn an LLM-based estimator $E_\theta(\tau, k, \beta) \approx U(\tau, k, \beta)$ which approximates U without needing to run $k(\tau)$. In particular, we leverage π on which, as in MAGELLAN, we attach one MLP per arm $k \in K$ (i.e., computing each arm's utility separately). For a given arm k, we use an MLP with two heads: one for estimating R and the other one for C. The global utility is obtained by computing $E_\theta(\tau, k, \beta) = \beta R_\theta(\tau, k) + (1 - \beta)C_\theta(\tau, k)$. In Appendix F.2, we discuss variations on this architecture. In particular, we look at how

[2]Note that, especially in the case where LLMs are used, such a function can be stochastic.

a single MLP can be used by inserting β and k in the LLM's prompt. See Figure 5.13 for a schema of our approach.

To train E_θ, we update the MLPs and π (using LoRA adapters (Hu et al., 2022) which are deactivated when using π to solve tasks, so that its performance is not affected). As in MAGELLAN, we keep a buffer $\mathcal{H}$ of the last L tasks along with the chosen arm and associated observed utility (i.e., a tuple $< \tau, \beta, k, U(\tau, k, \beta) >$). Every F tasks, we randomly sample a batch from $\mathcal{H}$ and train E_θ to approximate the sampled outcomes with a mean squared error loss.

Finally, for any given task τ and balance β, we select k using an epsilon-greedy scheme where, with a probability $\epsilon = 0.2$, k is randomly selected, and, with a probability $1 - \epsilon$, $k = \arg max_{k \in K} E_\theta(\tau, k, \beta)$.

5.3.3 Experiments

We now propose to evaluate our method using two experimental setups. First, we carefully design calculator tools useful for solving addition and division. In these experiments, the optimal strategy is known. We thus leverage this setup to evaluate (1) the accuracy of our estimator and (2) the sample efficiency of our online approach (i.e., the number of tasks required to converge to the optimal strategy). We then evaluate our method in a more ecological setting, where our LLM is presented with questions from MMLU-Pro (Wang et al., 2024d) and receives assistance from multiple larger or more specialized LLMs.

In all our experiments, we consider and provide results with two different models used as the "main" LLM π: Llama 3.2 1B (Grattafiori et al., 2024) and Qwen 2.5 0.5B (Qwen et al., 2025). Results are repeated with four different random seeds (see Appendix F.1 for more details).

Convergence analysis on maths problems

In this first set of experiments, we consider a task space $\mathcal{T}$ made of maths operations. In particular, we sample tasks from $\mathcal{T}$ by first selecting an operator among $\{+, /\}$, and then sampling the two operands uniformly between 1 and 50 with six fractional digits. At every new task, β is also randomly sampled in $[0; 1]$. We define R to be a function of how close an answer is to the expected operation's answer up to eight fractional digits (see Appendix F.1.1).

We then propose three deterministic calculator tools ($\Phi = \{A, B, C\}$) to our LLM, all taking as input an operation to compute and returning the result of this operation. We assign each tool a fixed and task-independent cost C between 0 and -1, as well as two task-dependent rounding precisions (one for additions and another one for divisions) that is used to round each tool's answer to an operation. We set the costs to be $[-0.85, -0.025, -0.2]$ (for "A", "B", and "C" respectively) and the rounding precisions to be $[6, 4, 5]$ for additions and $[4, 5, 6]$ for divisions.

Our experiment (i.e., $\mathcal{T}$, R, Φ, and C) is designed such that the theoretical optimal strategy for any β is known. In particular, we created our tools and rewards such that it

is always more rewarding to call a tool than our LLMs (i.e., the problems are too hard for our Llama 3.2 1B or Qwen 2.5 0.5B to obtain a high reward R). As each tool's expected reward R and cost C is known, we can compute the theoretical optimal strategy given β. We provide these strategies in Appendix F.1.1.

We give our experiments a budget of 2048 tasks and perform evaluations every 32 on a fixed evaluation set composed of 32 randomly sampled tasks (16 additions and 16 divisions). Each evaluation consists of 5 trials per evaluation task with greedy sampling from the bandits. We compare our LLM-based estimator of U with two baselines, both estimating the per-arm R and C separately using two per-arm averages over the history of outcomes for each arm. The first baseline is a naive multi-armed bandit that ignores the task. The second is a contextual bandit that receives the *oracle information* of a task's operation (i.e., addition or division). It thus keeps, for each arm and both R and C, two histories: one for additions and one for divisions. Access to such information (i.e., a task's category, which affects tools' performance and cost) is an extremely strong assumption that is inaccessible in more ecological settings. For this reason, this baseline should be considered as an upper-bound that one cannot use in real-world settings. We summarize the estimators we compare (see Appendix F.1.3 for more details):

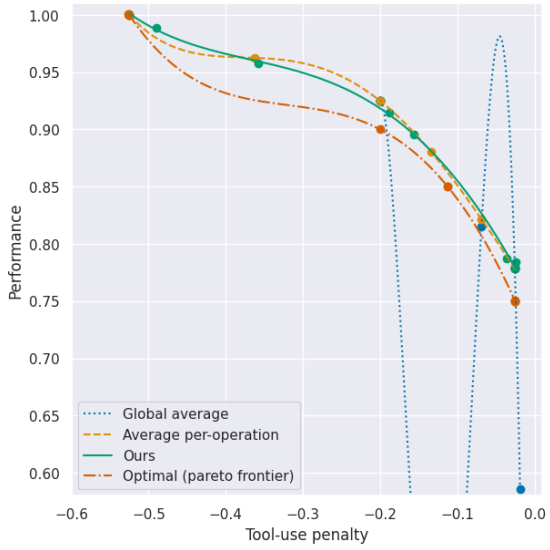
- Ours: $U(\tau, k, \beta) \approx \beta R_\theta(\tau, k) + (1-\beta) C_\theta(\tau, k)$.
- Average: $U(\tau, k, \beta) \approx \beta \frac{1}{|H_k|} \sum_{\tau' \in H_k} R(\tau', k) + (1-\beta) \frac{1}{|H_k|} \sum_{\tau' \in H_k} C(\tau', k)$, with H_k the history for arm k/
- Average per-operator: $U(\tau, k, \beta) \approx \beta \frac{1}{|H_k^o|} \sum_{\tau' \in H_k^o} R(\tau', k) + (1-\beta) \frac{1}{|H_k^o|} \sum_{\tau' \in H_k^o} C(\tau', k)$, with o the operator used in τ, and H_k^o the history for arm k and operator o.

We begin by investigating the performance of the different estimators evaluated, as well as how the resulting strategy the bandit converged to compares to the theoretical optimal strategy. At the end of the 2048 tasks, we evaluate each bandit obtained on our evaluation set. We report the per-objective evaluation utility (i.e., R and C separately) with $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ in figures 5.14a and 5.14c, and the evaluation global utility U for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ in tables 5.3 and 5.4.

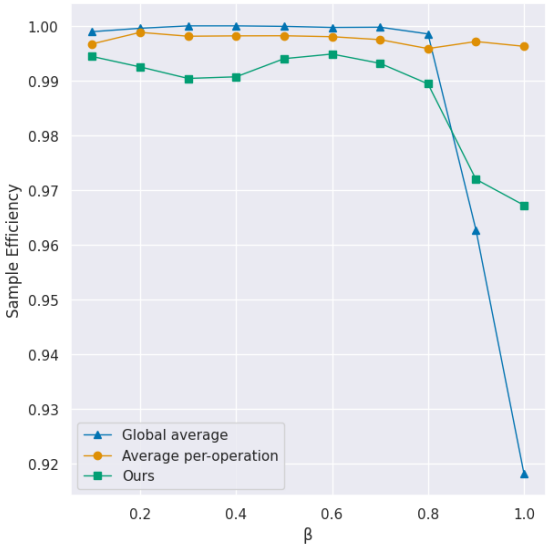
Table 5.3: Llama 3.2 1B - Evaluation utility U after 2048 tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ on maths problems. Results are averaged over 32 evaluation tasks and 4 seeds.

Method	$\boldsymbol{\beta = 0.2}$	$\boldsymbol{\beta = 0.4}$	$\boldsymbol{\beta = 0.6}$	$\boldsymbol{\beta = 0.8}$	$\boldsymbol{\beta = 1.0}$
OPTIMAL	0.13 (±0.0)	0.29 (±0.0)	0.46 (±0.0)	**0.7 (±0.0)**	**1.0 (±0.0)**
Average	**0.14 (±0.0)**	**0.3 (±0.0)**	**0.48 (±0.0)**	**0.7 (±0.0)**	0.92 (±0.0)
Average per-operation	**0.14 (±0.0)**	**0.3 (±0.0)**	0.47 (±0.0)	**0.7 (±0.0)**	**1.0 (±0.0)**
Ours	**0.14 (±0.0)**	**0.3 (±0.0)**	0.47 (±0.0)	0.69 (±0.0)	**1.0 (±0.0)**

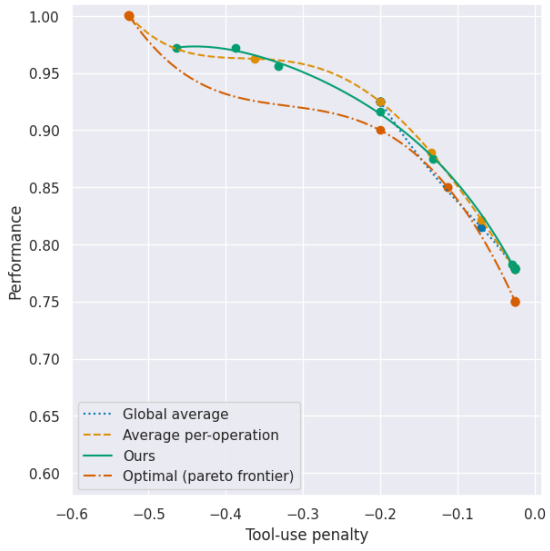
The results indicate that our LLM-based estimator matches the optimal utility for any β for Llama 3.2 1B and approaches it for Qwen 2.5 0.5B. Overall, our method obtains results close to "Average per-operator", indicating the estimator's ability to capture the information that tasks' operator lead to different behavior in accessible tools. As expected,



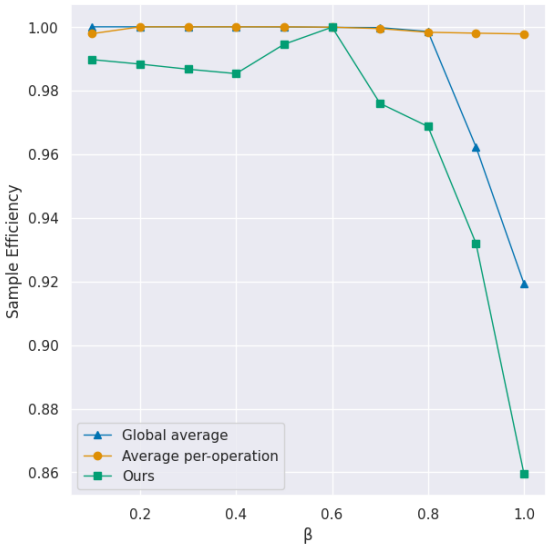
(a) Llama 3.2 1B - utilities



(b) Llama 3.2 1B - regret



(c) Qwen 2.5 0.5B - utilities



(d) Qwen 2.5 0.5B - regret

Figure 5.14: Per-objective final evaluation utility and regret sample efficiency over evaluation tasks (see Equation 5.5) for $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ on maths problems. Results are averaged over 32 evaluation tasks and 4 seeds.

"Average" does not lead to an optimal strategy at high β values, for which different tools for additions and divisions are required.

Readers might wonder why, for some β, a slightly higher performance than the theoretical optimal one is reachable. This is explained by the fact that, while the theoretical rounding precision of each tool is known, some randomly generated operations can empirically lead to results that are not rounded by a tool (e.g., 8.191039+5.524591 with "calculator-B" leads to $R = 1$ instead of the theoretical $R = 0.8$).

We then investigate how fast our estimators converge to the true value of U. For this, we compute the evolution of regret (w.r.t. the optimal U) over evaluations performed

Table 5.4: Qwen 2.5 0.5B - Evaluation utility U after 2048 tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ on maths problems. Results are averaged over 32 evaluation tasks and 4 seeds.

Method	$\boldsymbol{\beta = 0.2}$	$\boldsymbol{\beta = 0.4}$	$\boldsymbol{\beta = 0.6}$	$\boldsymbol{\beta = 0.8}$	$\boldsymbol{\beta = 1.0}$
OPTIMAL	0.13 (±0.0)	0.29 (±0.0)	0.46 (±0.0)	**0.7 (±0.0)**	**1.0 (±0.0)**
Average	**0.14 (±0.0)**	**0.3 (±0.0)**	**0.48 (±0.0)**	**0.7 (±0.0)**	0.92 (±0.0)
Average per-operation	**0.14 (±0.0)**	**0.3 (±0.0)**	0.47 (±0.0)	**0.7 (±0.0)**	**1.0 (±0.0)**
Ours	**0.14 (±0.0)**	**0.3 (±0.0)**	0.47 (±0.01)	**0.7 (±0.0)**	0.97 (±0.03)

every 32 tasks. We report in Appendix F.3.1 the regret curves of each estimator for $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. We summarize all regrets within a single figure for both LLMs in figures 5.14b and 5.14d by reporting the following sample efficiency measure:

$$SE(\beta, k) = 1 - \frac{1}{T}\sum_{t=0}^{T} \frac{1}{|\mathcal{T}_{test}|} \sum_{\tau \in \mathcal{T}_{test}} U(\tau, k^*, \beta) - U(\tau, k, \beta) \tag{5.5}$$

with k^* the optimal choice and $\mathcal{T}_{test}$ the set of evaluation tasks.

Figures 5.14b and 5.14d show that our estimator remains close to "Average per-operator" up to $\beta = 0.8$ before seeing its sample efficiency decrease. This decrease is explained by the fact that "calculator-A" is the optimal tool to call only for such high β values, and as evidenced by Appendix F.3.1, our LLM-based estimator tends to underestimate "calculator-A"'s utility. As a result, the epsilon-greedy exploration strategy adopted in our bandit requires a higher number of interactions to correct this underestimation and lead to the optimal strategy (see Figure 5.15). We argue that this is rather related to an exploration problem than a direct limitation of our LLM-based estimator. In addition to improving the action-level exploration of our bandit, a closer look at an active sampling strategy over the contexts (i.e., the maths operations and β) could have a significant impact, as optimal strategies depend on the context.

Finally, we also investigate how the different architectural choices discussed in Section 5.3.2 influence our method's sample efficiency in Appendix F.2. In particular, we demonstrate that updating the LLM with LoRA adapters yields mixed results compared to keeping the LLM unchanged, depending on the specific LLM used. Consequently, results showed here for Qwen 2.5 0.5B use LoRA adapters, while the ones for Llama 3.2 1B do not.

Overall, our experiments indicate that our method is (1) able to estimate both the performance and cost of each arm, (2) converges within a few hundred interactions, and (3) leads to optimal assistance-seeking strategies. In the next section, we demonstrate the general interest of our approach by applying it to a more ecological experimental setup by using tasks drawn from MMLU-Pro (Wang et al., 2024d).

Evaluation on MMLU-Pro

We now evaluate our approach in a more ecological setting using question-answering problems from MMLU-Pro (Wang et al., 2024d), which features 12032 problems organized in 14 categories ranging from maths to law. We randomly sample problems (and β) from

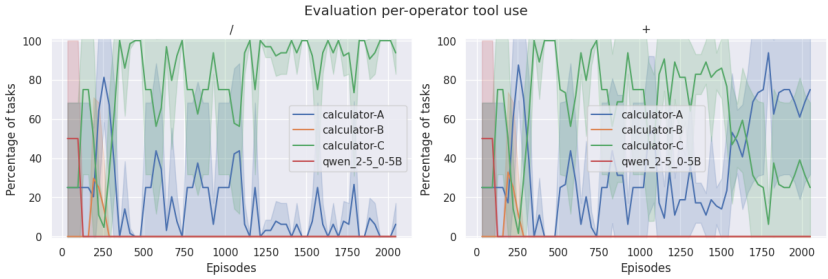


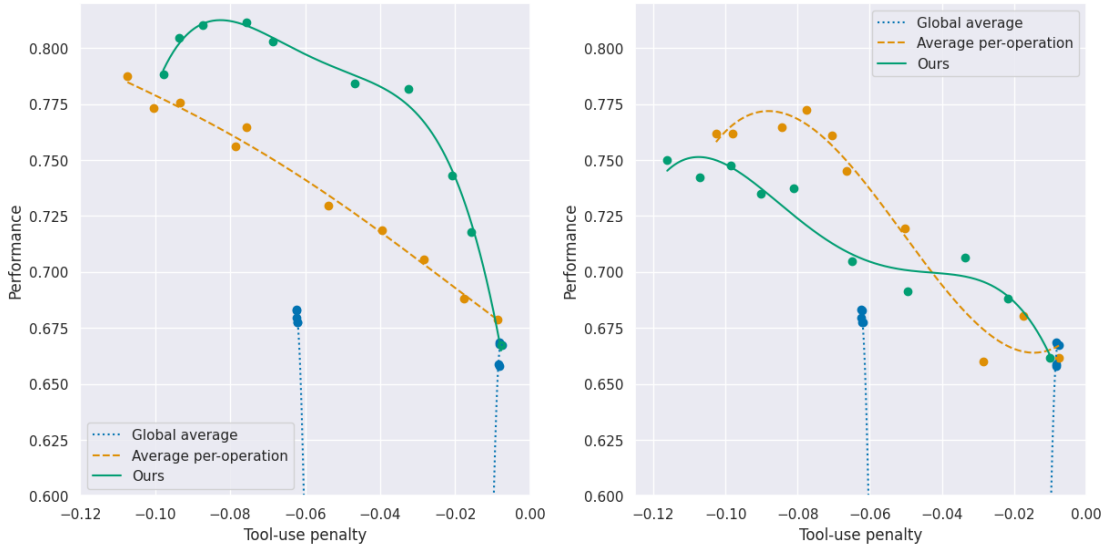
Figure 5.15: Evolution of tools called by our approach with Qwen 2.5 0.5B during evaluation for $\beta = 1$ (averaged over 32 evaluation tasks and 4 seeds). We report this evolution for divisions (left) and additions (right). While our approach quickly converged to the optimal strategy (calling calculator-A) for divisions, 1500 steps are needed for it to converge to calling calculator-A on additions. In particular, the abrupt change in strategy at 1500 steps highlights the exploration challenge of our approach.

the "test" split and filter out problems for which the number of characters exceeds 600 to reduce the computational cost, while keeping more than 80% (9834/12032) of the problems (see Appendix F.1.2). In addition to our main LLM π (i.e., either Llama 3.2 1B or Qwen 2.5 0.5B), we use three other LLMs for assistance: Llama 3.1 70B, Qwen 2.5 14B, and Llama expert, a Llama 3.2 1B model fine-tuned on problems belonging to the "maths" and "physics" categories. See Appendix F.4.1 for the performance of each if these LLMs on our outcome-based reward R. As the number of possible answers varies between problems, we randomly pick and propose four possible answers for each problem (including the right one).

Our outcome-based reward R returns 1 if the selected answer is the right one, 0 otherwise. For the cost reward C, we propose to take inspiration from how LLM providers estimate the cost of API calls by users: we compute the cost of an LLM's answer using a per-token-generated cost (see Appendix F.1.2). As in the previous section, we keep a zero cost for using our main LLM π (i.e., it has no cost for our π to keep the task to itself). We use the "validation" split from MMLU-Pro as our evaluation set $\mathcal{T}_{test}$. After filtering out problems with more than 600 characters, we obtain an evaluation set of 64 tasks (compared to 70 problems before filtering). We allocate to our experiments a budget of 10240 tasks and perform evaluation every 256 tasks (using a single trial per evaluation task). As in the previous section, we compare our LLM-based estimator to the two average-based baselines: a per-arm average ("Average") and a per-arm per problem category average ("Average per-category"). For the latter, we assume access to each task's category, which is not feasible in real-world cases where users provide problems that are not from known datasets.

In this setting, the theoretical optimal strategy for any β value cannot be known a priori. Therefore, we cannot compute the regret and only report the final observed evaluation utility over multiple β in Figure 5.16, as well as in tables 5.5 and 5.6. We also report in Appendix F.4 additional results.

The results show that our approach scales to more complex problems, such as the ones from MMLU-Pro. In particular, our estimator obtains similar (for Qwen 2.5 0.5B) or even higher (for Llama 3.2 1B) evaluation utility than "Average per-category". These results are notably explained by the outcome-based reward R of each LLM on MMLU-Pro's task.



(a) Llama 3.2 1B (b) Qwen 2.5 0.5B

Figure 5.16: Per-objective evaluation utility after 10240 tasks for $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ on MMLU-Pro. Results are averaged over 64 evaluation tasks and 4 seeds.

Table 5.5: Llama 3.2 1B - Evaluation utility U after 10240 tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ on MMLU-Pro. Results are averaged over 64 evaluation tasks and 4 seeds.

Method	$\boldsymbol{\beta = 0.2}$	$\boldsymbol{\beta = 0.4}$	$\boldsymbol{\beta = 0.6}$	$\boldsymbol{\beta = 0.8}$	$\boldsymbol{\beta = 1.0}$
Global average	**0.13 (±0.0)**	0.26 (±0.0)	0.38 (±0.02)	0.53 (±0.01)	0.68 (±0.03)
Average per-category	0.12 (±0.01)	0.26 (±0.02)	0.43 (±0.01)	0.6 (±0.01)	**0.79 (±0.02)**
Ours	**0.13 (±0.0)**	**0.29 (±0.02)**	**0.45 (±0.02)**	**0.63 (±0.03)**	**0.79 (±0.03)**

Table 5.6: Qwen 2.5 0.5B - Evaluation utility U after 10240 tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ on MMLU-Pro. Results are averaged over 64 evaluation tasks and 4 seeds.

Method	$\boldsymbol{\beta = 0.2}$	$\boldsymbol{\beta = 0.4}$	$\boldsymbol{\beta = 0.6}$	$\boldsymbol{\beta = 0.8}$	$\boldsymbol{\beta = 1.0}$
Global average	**0.13 (±0.0)**	**0.26 (±0.0)**	0.38 (±0.02)	0.53 (±0.01)	0.68 (±0.03)
Average per-category	0.12 (±0.01)	**0.26 (±0.02)**	**0.43 (±0.02)**	**0.6 (±0.03)**	**0.76 (±0.04)**
Ours	0.12 (±0.01)	0.25 (±0.03)	0.41 (±0.03)	0.58 (±0.03)	0.75 (±0.04)

As shown in Appendix F.4.1, the LLMs considered in these experiments exhibit high variance in their performance within each category (i.e., an LLM's ability at answering a question from MMLU-Pro does not only depend on the question's category). As a result, "Average per-category" can only converge to the mean performance over each category, while our approach manages to grasp finer-grained differences between tasks.

5.3.4 Conclusion

In this work, we proposed a framework for training LLMs to autonomously decide when to request external assistance. Building on the evidence presented in Section 5.2 that LLMs can estimate their own functional competence, we leveraged these metacognitive abilities to guide help-seeking behavior. Crucially, our framework accounts not only for performance but also for the cost of external assistance, framing help-seeking as a multi-objective contextual bandit problem. Using only online interactions with tasks, the LLM learns to estimate both the performance and cost of available options—its own response and that of external tools or models. These estimations enable adaptive decisions over whether to solve the task independently or delegate it using a user-defined scalarization weight to flexibly prioritize performance or cost.

We validated our approach in a controlled setting using math problems and calculator tools, showing that it converges to the optimal help-seeking strategy for any given performance-cost trade-off. We then demonstrated the method's generality by applying it to MMLU-Pro tasks with external LLMs as assistance, confirming its applicability in more realistic and diverse environments. Overall, our results show that LLMs can learn frugal and effective help-seeking strategies purely from interaction, without requiring annotated datasets or offline supervision.

However, our findings also highlight an important challenge: efficient exploration of both the assistance space and the contextual space (i.e., tasks and scalarization weights). In particular, we observed that combining our LLM-based estimator with naive epsilon-greedy exploration and random context sampling struggles to identify assistive strategies that are optimal only under specific trade-offs. A promising future direction would be to couple our framework with active sampling approaches inspired by automatic curriculum learning. For example, MAGELLAN could be used to guide the selection of tasks and scalarization weights, accelerating convergence by focusing exploration on regions of high learning potential.

Moreover, one limitation of our current experiments is the single-turn setup, where each task is delegated in its entirety to a single option (LLM or assistance). Future work could extend this approach to multi-step tasks, framing the decision as a sequential process. An even finer-grained extension could involve token-level decisions—determining at each token whether to generate or defer to an external tool—as explored in prior work on tool-augmented LLMs (e.g., (Schick et al., 2023)).

Finally, while this work focused on using metacognitive estimates to identify when a task exceeds an LLM's current capabilities, an important next step is to enable the LLM to learn from the assistance it receives. This would allow it to eventually perform tasks independently, increasing its functional competence. Integrating such learning would introduce non-stationarity into the LLM's performance, but our results from MAGELLAN suggest that LLM-based estimators can quickly adapt to evolving competence.

5.4 Discussion

In this chapter, we extended the study of functional grounding in LLMs by investigating how such grounding can be achieved in large and complex goal spaces—settings where

LLMs face a wide and potentially infinite range of functional competencies to acquire. We drew inspiration from how humans manage this challenge through autotelic learning: an intrinsically motivated process by which individuals define, select, and pursue their own goals.

In Section 5.1, we began by examining the social aspects of autotelic learning. Specifically, we explored how social partners guide children's learning by providing cues or descriptions, enabling the mapping between sequences of actions and corresponding goals. Inspired by prior work in autotelic RL with social partners, we introduced SAC-GLAM, an extension of GLAM (from Section 3.2) that incorporates off-policy RL and hindsight relabeling mechanisms. This allowed us to improve sample efficiency in goal-driven learning settings.

We then turned to the goal prioritization problem in Section 5.2, a central challenge in autotelic learning. Building upon the automatic curriculum learning literature, we proposed MAGELLAN—a metacognitive module added to the LLM. This module enables the LLM to estimate both its competence and Learning Progress across large, language-defined goal spaces. MAGELLAN addresses key limitations in prior work on LP estimation, particularly in discrete and unstructured spaces like those expressed in natural language. Our results demonstrate that MAGELLAN can guide goal selection effectively and, importantly, that learning to estimate one's own competence constitutes a form of grounding in itself—aligning the LLM's internal representations with the environment's structure. Importantly, our work introduces significant contributions to a major challenge for future LLMs: developing their metacognitive skills (Johnson, 2022). By learning to track their own LP to scaffold their learning, we showed that LLMs can learn to estimate and generalize their functional competence over large goal spaces.

In Section 5.3, we built on these metacognitive abilities to tackle a critical real-world challenge: deciding when an LLM should seek external assistance. This ability is essential for LLMs embedded in human-facing applications. We showed that, in addition to estimating their own competence, LLMs can estimate the competence and associated cost of external assistance on language-specified tasks. We framed assistance-seeking as a multi-objective contextual bandit problem, where the LLM learns to select between solving tasks autonomously or delegating them, based on a user-defined trade-off between performance and cost. Our online learning method enables the LLM to autonomously discover how to dynamically adapt its strategy across varying budgets and task distributions through interactions. Together with Section 5.2, this work underscores the importance of developing metacognitive skills in LLMs: it enables models to express when assistance is needed or to identify what to learn next. However, the potential of metacognition in LLMs extends beyond these two applications and represents a critical avenue for future research.

Despite these advances, the work presented in this chapter assumes that the complete space of possible goals is known a priori. While we have focused on goal selection and learning, a crucial next step is the study of goal generation. As discussed by Colas et al. (2022a), humans use language compositionally to generate new goals by recombining known concepts. Colas et al. (2020) previously demonstrated an autotelic agent that used language to invent goals, but their method was restricted to simple grammar-based heuristics. A major challenge remains in scaling goal generation to the complexity of

natural language, and LLMs appear as natural candidates for this task. One promising direction would involve asking the LLM to generate novel goals and filtering them using MAGELLAN. For instance, Pourcel et al. (2024b) and Pourcel et al. (2024a) showed that LLMs can be used to generate goals defined with code functions evaluating a trajectory. However, they both only leverage empirical learnability measures to evaluate a goal's interest. Leveraging MAGELLAN would enable the LLM to evaluate a goal's learnability without requiring the learner to try the goal. Parallel to this, we also argue that MAGELLAN's latent space itself offers a powerful avenue for exploration. For example, inspired by ALP-GMM (Portelas et al., 2020a), one might directly sample regions of the latent space with high expected LP and then decode these into natural language goals.

Moreover, this latent space—capturing both environmental dynamics and the agent's functional competence—could support a range of other capabilities. One limitation of our current goal-selection approach is the multi-armed bandit formulation used for sampling goals, where each goal is treated as an independent arm. This design requires estimating LP across the full goal space and relies on naive random exploration to discover progress niches. Here again, MAGELLAN's latent space could help. For instance, one might select only a subset of high-potential goals (e.g., top-n in latent LP score), or cluster goals and apply a hierarchical curriculum over goal modules, as seen in prior modular approaches (Baranes & Oudeyer, 2009; Laversanne-Finot et al., 2018; Colas et al., 2019). Similarly, exploration could be biased toward goals near known high-competence areas in the latent space. While this list is not exhaustive, we argue that MAGELLAN's latent space offers a powerful abstraction that enables LLMs to reason and act over natural language goal spaces without operating directly at the surface linguistic level. This opens new paths for scaling functional grounding in LLMs through autotelic open-ended learning.

This chapter concludes our empirical contributions to the functional grounding of LLMs through online interactions with an environment and intrinsically motivated reinforcement learning. The next and final chapter will focus on summarizing this manuscript's contribution and discussing further avenues opened up by this research.

Chapter 6

Discussion

Contents

6.1 Summary of contributions 127
6.2 Perspectives 129
6.2.1 From language models to action models 129
6.2.2 On functional grounding and reasoning 130
6.2.3 Beyond reinforcement learning 131
6.2.4 Towards self-generated exercises 131
6.2.5 Towards LLMs capturing a causal model of the world 133
6.2.6 Metacognition, alignment, and safety 133
6.2.7 The quest of autotelic agents 134
6.3 Conclusion 136

We now move to the last chapter of this thesis. We will first summarize the contributions presented throughout the manuscript before discussing future avenues for the functional grounding of LLMs.

6.1 Summary of contributions

In our introductory and background chapters, we revisited the long-standing goal of artificial intelligence: building machines capable of understanding natural language. Today, large-scale deep learning—particularly through training on massive corpora—dominates computational approaches to language understanding. While large language models (LLMs) have demonstrated impressive linguistic capabilities, a growing body of work has highlighted their limitations. These limitations are often attributed to their passive, purely statistical learning processes, raising critical questions about whether such models truly grasp meaning. In parallel, research in human cognition and developmental linguistics has emphasized the importance of embodied interaction and curiosity-driven, goal-directed behavior in language acquisition. This perspective supports the notion of grounded language learning—where meaning arises from interaction with the environment, both sensorimotor and social. This is especially relevant to functional competence, defined as the ability to use language to achieve goals. Empirical studies suggest that such competence emerges from children's intrinsic motivation to influence their surroundings and reach

self-defined goals—a principle known as autotelic learning. This manuscript explores how to bridge the gap between today's LLMs and grounded language understanding as described by developmental sciences, with a focus on functional competence. Our central contribution is the proposal of an interaction-based framework for *functionally grounding LLMs using autotelic reinforcement learning (RL)*.

In Chapter 3, we introduced the concept of functional grounding—the process by which the internal representations used in symbol processing are aligned with the external dynamics of an environment, enabling the agent to control, model, and predict the environment to pursue goals. Drawing from embodied theories of language acquisition, we proposed a framework built upon RL (to learn through interaction) and intrinsically motivated exploration (particularly goal-directed autotelic learning). The remainder of the chapter focused on the first facet of our framework: functional grounding through online RL. We introduced *GLAM*, a functional grounding method based on online RL, and conducted extensive analyses demonstrating how interactive learning significantly improves the functional competence of LLMs. Notably, we showed that GLAM specifically shapes internal representations relevant to functional competence while preserving other forms of knowledge, such as referential meaning. This representational alignment leads to better generalization in the face of environmental changes. We also identified limitations of GLAM in cases of overfitting to prompt formats. However, we demonstrated that incorporating prompt diversity and contrastive learning leads to more robust competence and generalizes to downstream tasks like environment question-answering. We further showed that passive learning methods like behavioral cloning fall short compared to online RL, highlighting the importance of active interaction as emphasized in developmental theories.

In Chapter 4, we shifted focus to the predictive aspect of functional grounding, specifically how it enhances an LLM's ability to serve as a world model. Inspired by how humans, particularly children, build and refine intuitive theories to explain their environment, we introduced *WorldLLM*, a framework in which an LLM (1) acts as a forward model conditioned on natural language hypotheses, and (2) is intrinsically motivated to explore its environment to refine these hypotheses. Using Bayesian inference for hypothesis refinement and prediction error as the intrinsic reward, we developed a first instantiation of this framework. Our results show that WorldLLM enables the iterative generation and refinement of human-interpretable theories, which enhance the LLM's ability to model and predict environmental dynamics.

Chapter 5 then explored the second facet of our framework: autotelic learning. In particular, we argued that acquiring general functional competence requires navigating an effectively infinite space of possible goals. Humans address this through curiosity-driven exploration, particularly autotelic behavior. Designing such autotelic RL agents with LLMs poses challenges—chiefly, how to efficiently learn in settings where agents may self-select goals beyond their current abilities. Drawing from prior work on autotelic RL, we proposed *SAC-GLAM*, an extension of GLAM that integrates off-policy RL and hindsight relabeling. In addition to improving the sample efficiency over GLAM, SAC-GLAM enables social learning in autotelic settings. We then addressed the task selection problem, central to autotelic agents. Building on research in automatic curriculum learning, we introduced *MAGELLAN*, a novel method that equips LLMs with metacognitive monitoring—the ability to estimate their own competence. MAGELLAN is the first

approach enabling efficient estimation of Learning Progress on language-defined goals. By combining GLAM (for acquiring competence) with MAGELLAN (for prioritizing high-LP tasks), we demonstrated that LLMs can efficiently navigate vast task spaces. This is achieved by leveraging semantic relationships between tasks, enabling generalization beyond observed examples. Our results show that not only does MAGELLAN improve functional competence acquisition, but the metacognitive estimation process itself further aligns the LLM's internal representations with environmental dynamics. Finally, we extended the role of metacognition by exploring assistance-seeking behavior. We proposed a novel, purely online framework in which LLMs learn to trigger external help based on self-assessed competence. Importantly, we introduced a multi-objective setting, where assistance carries a cost, and proposed a multi-armed bandit solution that balances task performance and assistance usage according to user-specified preferences. Our results show that LLMs can learn optimal assistance-seeking strategies under budget constraints, further demonstrating the value of metacognitive self-assessment for functional competence.

6.2 Perspectives

In this section, we outline several future research directions opened by the contributions presented in this manuscript. Our work aimed to advance the development of LLMs as functionally grounded and autotelic agents—that is, agents capable of acquiring and refining their functional competence through autonomous, goal-driven interaction with the environment.

6.2.1 From language models to action models

Throughout this manuscript, we have argued for a perspective that considers LLMs as embodied agents—systems capable of interacting with an external environment and updating their internal representations based on those interactions. This framing is increasingly shared by a growing research community that seeks to leverage LLMs' inherent strengths, such as reasoning and planning, in systems designed for real-world interaction. Such models are now often referred to as action models—general-purpose or foundation models for agents capable of perceiving, deciding, and acting in complex environments. Recent advances have demonstrated the potential of both LLMs and VLMs to control real-world robots (Ahn et al., 2022; Zeng et al., 2023; Huang et al., 2023; Zitkovich et al., 2023; Shukor et al., 2025). However, despite encouraging results, significant challenges remain.

Notably, many of today's action models rely heavily on imitation learning from expert demonstrations (Jiang et al., 2023b; Brohan et al., 2023; Zitkovich et al., 2023; Shukor et al., 2025). This raises important questions around data collection: unlike LLMs, which benefit from massive self-supervised training on readily available internet-scale text corpora, acquiring large-scale expert demonstrations for sequential decision-making tasks (particularly in robotics) remains difficult. Open-source efforts such as Shukor et al. (2025) represent important steps toward building accessible and unified datasets for training

action models, but we are still far from achieving the scale and diversity seen in language data for pre-training general-purpose action models.

More recently, Zhai et al. (2025) proposed a first attempt at adapting GLAM-style fine-tuning to VLMs used as action models. Functional grounding of VLMs is a particularly urgent challenge given evidence that VLMs often underperform LLMs in terms of functional abilities (Wang et al., 2024a; Aissi et al., 2025). While initial results demonstrate that fine-tuning VLMs using RL is feasible, performance gains remain modest—even when preceded by imitation learning—as highlighted by Aissi et al. (2025). Importantly, these studies have so far been confined to simulated environments. Scaling these approaches to real-world robotics introduces an additional set of obstacles. Most traditional RL algorithms suffer from poor sample efficiency, rendering them unsuitable for direct application in physical systems. As a result, the robotics community has largely adopted *sim-to-real* transfer strategies, where policies are first trained in simulation before being adapted to real-world platforms (Zhao et al., 2020). In conclusion, while equipping LLMs and VLMs with real-world embodiment offers a promising path toward grounding their functional competence directly in physical environments, substantial research is still needed.

6.2.2 On functional grounding and reasoning

Recent research has increasingly emphasized the role of reasoning in enhancing the performance of LLMs, particularly in settings where agents must make decisions or interact with an environment. A number of works have demonstrated that reasoning can serve as a form of internal planning, guiding the model toward better actions by leveraging its internal knowledge and commonsense priors (Yao et al., 2022; Zeng et al., 2023; Huang et al., 2023; Zhai et al., 2025). This approach has the appealing advantage of bypassing the need to explicitly learn a forward model through interaction, as traditionally done in model-based RL, instead exploiting the LLM's pre-trained internal representations to simulate outcomes. However, as we showed in Chapter 4, this form of internal simulation can be limited. Our proposed framework, WorldLLM, demonstrated that an LLM's world model can be substantially improved through actual interactions with the environment. This opens an exciting direction for future research: investigating the synergy between reasoning-based planning and interaction-based world modeling.

Another critical open question is how to improve the reasoning abilities themselves, especially in settings where no gold-standard answer or plan is available. While several RL approaches have successfully improved LLMs' reasoning in domains with well-defined supervision—such as mathematical problem solving (Uesato et al., 2022; Havrilla et al., 2024; DeepSeek-AI et al., 2025)—these settings do not generalize to open-ended sequential decision-making tasks, where the optimal trajectory is often unknown. An emerging solution to this challenge is found in token-level RL for LLM agents, where each generated token is treated as an action (Wen et al., 2024a,b; Zhou et al., 2024b). This perspective provides a unified framework that bridges action-level approaches (e.g., GLAM) with token-level methods used in RLHF and reasoning-focused fine-tuning (Ouyang et al., 2022; Ramamurthy et al., 2023; Gulcehre et al., 2023; Ahmadian et al., 2024). By treating both reasoning and environmental actions as part of the same trajectory, token-level RL offers a principled way to optimize both. However, this setting introduces substantial challenges:

in particular, reward sparsity, since a reward from the environment is only provided at the end of a long reasoning sequence. Additionally, exploring the vast space of possible reasoning chains remains an open problem, especially when intermediate reasoning steps cannot be supervised.

6.2.3 Beyond reinforcement learning

Throughout this research, we focused primarily on online RL as a mechanism for the functional grounding of LLMs. However, aligning an LLM's internal representations with the dynamics of its environment can be achieved through alternative, and often complementary, mechanisms. In particular, our proposed WorldLLM framework demonstrated that in-context learning—combined with intuitive theories—can functionally ground an LLM's internal world model without parameter updates. This ability to adapt based on examples or information embedded directly in the prompt is one of the defining and most powerful features of LLMs (Brown et al., 2020). Beyond simple task instruction, in-context learning has been increasingly leveraged to influence LLM behavior through interaction with an environment. Several recent works have shown that LLMs can adapt based on feedback received following their actions, using this feedback to improve their strategy (Huang et al., 2023; Zeng et al., 2023; Yao et al., 2022; Wang et al., 2023b,a,c).

These methods suggest a form of dynamic adaptation akin to episodic control, a learning mechanism inspired by the human ability to recall and reuse high-reward experiences stored in memory (Pritzel et al., 2017; Hansen et al., 2018; Hu et al., 2021). While classical episodic control approaches have typically focused on reproducing stored high-reward trajectories, LLMs introduce the possibility of reflecting on these past experiences to derive more abstract lessons or strategies. This has been explored in recent work through LLM-based reflection mechanisms, where models are prompted to analyze their past behaviors to improve future decisions (Shinn et al., 2023; Chen et al., 2024; Yao et al., 2024). In this context, episodic control in LLMs moves beyond imitation, offering a flexible, memory-driven approach to adaptation that can complement slower, gradient-based learning.

Moreover, while this manuscript has focused on functional competence in simulated physical environments, a broader view of functional grounding includes the ability to predict, influence, and collaborate with social partners, such as human users. As highlighted by Roy (2005b), this social dimension is core to the notion of functional competence. However, grounding through social interaction imposes new constraints: unlike simulations, human-centered interactions are costly and sparse, making sample-efficient adaptation crucial. In this context, exploring fast, non-parametric update mechanisms such as episodic control, reflection, and in-context adaptation becomes essential. Future research should investigate how these mechanisms can be systematically combined with RL to ground functional competence in settings involving partners.

6.2.4 Towards self-generated exercises

Throughout this manuscript—particularly in Chapter 5—we explored and proposed methods for building autotelic LLMs. The defining characteristic of autotelic agents is

that they self-generate the problems they learn to solve. As discussed, this self-generation process is neither purely random nor entirely individual: in humans, learning is shaped by a range of constraints, including caregiver scaffolding during infancy, cultural artifacts and values (Bruner, 1990), and intrinsic signals such as Learning Progress. In contrast, most current LLMs are trained via imitation learning on human-curated datasets or are aligned with human preferences via RLHF or supervised fine-tuning. Their learning problems are pre-defined and static. Enabling LLMs to self-generate meaningful exercises opens the door to open-ended learning, allowing models to transcend the limited scope of existing datasets and discover new problems that foster further growth in functional competence.

As a step in this direction, we introduced MAGELLAN, a method through which LLMs can develop metacognitive abilities by estimating their competence and Learning Progress across large goal spaces. These estimations can be used to scaffold the agent's learning process by prioritizing tasks with high expected progress. However, MAGELLAN, as presented, assumes a closed set of goals known a priori. Moving toward true open-endedness will require LLM agents to generate novel goals themselves. One natural extension is to use MAGELLAN's competence and Learning Progress estimations as a signal for goal generation. In the context of continuous task spaces, numerous automatic curriculum learning approaches have been proposed to dynamically generate goals based on difficulty, regret, or other learning signals (Florensa et al., 2018; Racaniere et al., 2020; Dennis et al., 2020; Jiang et al., 2021a; Parker-Holder et al., 2022). However, goal generation in natural language introduces unique challenges.

In particular, starting from a randomly initialized language goal generator is likely to be sample-inefficient. Recent work like OMNI-EPIC (Faldor et al., 2025) has shown that LLMs can be used to generate sequences of tasks, with each candidate task then accepted or rejected based on an interestingness criterion modeled by an LLM. This iterative rejection sampling framework could be adapted using MAGELLAN's Learning Progress estimates as a principled selection signal, ensuring that generated tasks meaningfully advance the LLM's competence. A similar framework could also be applied to code-defined goals. For instance, Pourcel et al. (2024b) used LLMs to generate a dataset of diverse and interesting code puzzles through a three-step process: (1) generating a new puzzle conditioned on previously generated ones, (2) attempting to solve it to assess its interestingness, and (3) assigning it a category label to enable diversity search across puzzle types. In parallel, Pourcel et al. (2024a) proposed using LLMs to generate programs that define reward functions for autotelic RL agents. By maintaining a buffer of empirically learnable reward functions, they showed that LLMs can guide the progressive acquisition of increasingly complex behaviors. In both of these approaches, MAGELLAN could serve as a powerful alternative to the interestingness heuristics currently used. Moreover, MAGELLAN induces a latent embedding space of goals, in which proximity reflects similar levels of competence or learning potential. This latent space could be directly sampled to identify new, promising goals. A key open question then becomes: how can we map these latent samples back into valid, interpretable natural language tasks?

6.2.5 Towards LLMs capturing a causal model of the world

When introducing the concept of functional grounding, we emphasized the importance of modeling environmental dynamics—a necessary ability for functional competence. More broadly, a core property expected from LLMs, especially as they are increasingly integrated into softwares supporting users' daily tasks, is the capacity to build and leverage a model of the world that aligns with human expectations and physical reality. Evaluating the extent to which LLMs model the world has become an active research focus in recent years (Hao et al., 2023a; Li et al., 2023a; Vafa et al., 2024; Ding et al., 2025; Ying et al., 2025). Despite their impressive performance, we argued in the introduction and Chapter 2 that LLMs' lack of grounding in the real world may fundamentally limit their capacity to build reliable world models. While some models appear to handle physical concepts adequately, it remains unclear whether such abilities reflect an internal causal model that supports causal inference, or if they result from surface-level statistical correlations learned during pre-training.

To address this gap, the present research proposed several contributions aimed at improving LLMs' world modeling abilities. In our large-scale study of GLAM, we showed that LLMs fine-tuned via online RL not only demonstrated stronger functional competence but also improved performance on environment-specific question answering—such as identifying which object is best suited to solve a particular task. In WorldLLM, we showed that prompting LLMs with natural language theories enhanced their forward modeling abilities. In MAGELLAN, we demonstrated that training LLMs to estimate their own competence leads to latent representations that reflect the structure of environmental goals—grouping similar dynamics together. However, whether these internal structures constitute true causal models remains an open question. For instance, while WorldLLM relies on natural language theories, it is unclear whether the LLM is capable of reasoning causally from them. To assess this, the question-answering metrics used in our GLAM evaluation could be extended to explicitly test causal inference, such as reasoning under interventions or counterfactual scenarios, providing a clearer window into whether a functionally grounded LLM develops causal understanding.

Moreover, while the discussion above focused on physical causal models, humans also build causal models of social environments. The ability to model others—referred to as *Theory of Mind*—is a foundational aspect of human cognition and closely tied to functional competence in social settings (Tomasello, 2019; Roy, 2005b). A promising avenue for future work is to explore whether functional grounding, when applied to social interactions, can lead LLMs to acquire models of other agents' beliefs, goals, and intentions. This could be investigated through interaction with both artificial peers (e.g., other LLMs) and natural users.

6.2.6 Metacognition, alignment, and safety

LLMs are increasingly embedded into real-world applications, with the recent emergence of models equipped with the ability to act through web search, tool use, or API calls significantly extending their capabilities. However, a key challenge is to evaluate and control the safety of these applications (François et al., 2025). One major concern lies in

the cultural values and biases embedded within these models (Bender et al., 2021; Gallegos et al., 2024). As LLMs learn from massive corpora of human-generated data, they may replicate or amplify societal biases. Another critical issue is the generation of factually incorrect or fabricated content, commonly referred to as "hallucinations." Studies such as Hill (2023) and Abdelghani et al. (2025) have shown that users—particularly children and students—can be easily misled by such false content. This raises urgent ethical concerns, especially in educational and decision-support settings. To mitigate these risks, a key goal is to align LLMs more closely with the real world and with human expectations. This alignment includes not only grounding models in physical reality, but also ensuring their behavior respects human values and norms. In this thesis, we explored one specific axis of alignment: the functional grounding of LLMs through interaction-based learning. By aligning an LLM's internal representations with the dynamics of the environment it operates in, functional grounding holds promise for reducing hallucinations and improving reliability. However, as discussed in the previous section, whether such grounding leads to the formation of explicit causal models of the world remains an open question

Parallel to this, another promising avenue for improving LLM safety is the development of metacognitive abilities—the capacity of a model to monitor and assess its own performance (Johnson, 2022; Johnson et al., 2025). One of the key risks with current LLMs is their inability to signal uncertainty or recognize the limits of their competence, leading users to misplace trust in incorrect outputs (Steyvers & Peters, 2025). Augmenting LLMs with metacognitive abilities is essential for (1) alerting users when they are likely to fail, (2) helping users understand LLMs' responses, and (3) assisting engineers in diagnosing and improving the system's capabilities. In Chapter 5, we proposed and empirically validated a framework for endowing LLMs with metacognitive monitoring abilities. We demonstrated that LLMs can learn to self-assess their functional competence through interaction, and use these assessments to scaffold their own learning (by prioritizing high LP tasks) and to determine when external help is needed. Nevertheless, our work focused on a specific aspect of metacognition: the ability to predict one's likelihood of success on a task. Future research could investigate broader metacognitive capacities in LLMs.

6.2.7 The quest of autotelic agents

While this manuscript primarily focused on grounding the functional competence of LLMs through autotelic RL, the contributions presented here can also be viewed as advancing the broader quest of building autotelic agents. In this perspective, LLMs are key architectural components within more general autotelic RL frameworks. Our results demonstrate several ways in which LLMs can enhance the capabilities of autotelic agents. For instance, GLAM and SAC-GLAM showed that LLMs, when fine-tuned with online RL, can serve as efficient policy learners, significantly outperforming randomly initialized networks in text-based environments. In MAGELLAN, we demonstrated that LLMs can estimate Learning Progress and enable efficient goal-selection for natural language tasks. Additionally, as discussed in Section 6.2.3, the unique in-context learning and reasoning capabilities of LLMs suggest their potential for fast policy adaptation, in particular via episodic control—retrieving and recombining past interactions to guide future behavior. We recently proposed an example of such an LLM-augmented autotelic agent with HERAKLES (Carta et al., 2025), a hierarchical autotelic RL framework in which an

agent incrementally learns a growing repertoire of compositional skills. In HERAKLES, an LLM is used to (1) plan which previously learned skills to compose (via GLAM), and (2) guide goal selection (via MAGELLAN). Moreover, a MAGELLAN estimator is also used to constrain the space of skills the LLM can compose (see Figure 6.1).

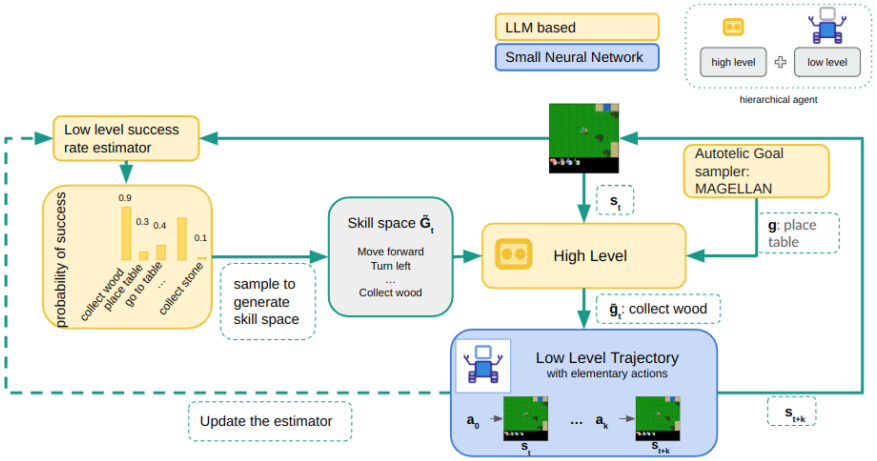


Figure 6.1: Schematic overview of HERAKLES, an autotelic hierarchical RL agent (Carta et al., 2025). The high-level policy—an LLM fine-tuned with GLAM—selects skills based on goals sampled by MAGELLAN. These skills are passed to a low-level policy, a neural network trained via online RL, which executes them in the environment. A competence estimator monitors the low-level policy's ability to execute each skill and is used to constrain the high-level policy's action space to executable skills. As MAGELLAN samples new goals, those successfully achieved by the low-level policy are compiled into new skills, progressively expanding the agent's repertoire.

More broadly, LLMs can be seen as "culture models" for autotelic agents—encoding commonsense knowledge, implicit values, and high-level priors shaped by human discourse (Kovač et al., 2024). This opens new avenues for replacing the handcrafted expert knowledge typically used in autotelic RL. For example, early work by Colas et al. (2020) relied on a manually defined goal grammar and handcrafted combination rules. With LLMs, such constraints can now be learned or generated dynamically. Importantly, this cultural grounding can also guide exploration: while naive autotelic agents treat all learnable goals as equally valuable, LLMs provide a means of prioritizing or filtering goals in line with human preferences or usefulness. For instance, OMNI (Zhang et al., 2024b) illustrated how an LLM can act as a filter to constrain the goal space to "human-interesting" goals, steering the agent's exploration towards more meaningful outcomes.

This LLM-guided autotelic learning loop opens especially exciting perspectives in the domain of automated scientific discovery. For this, Intrinsically-Motivated Goal Exploration Processes (IMGEPs) have already demonstrated that autotelic agents can generate novel hypotheses and discoveries in artificial life (Reinke et al., 2020; Etcheverry et al., 2020), biology (Etcheverry et al., 2024), chemistry (Grizou et al., 2020), and physics (Falk, 2024). In parallel, LLMs have recently been investigated as tools to automate parts of scientific workflows in machine learning (Lu et al., 2024; Yamada et al., 2025) and artificial life (Khajehabdollahi et al., 2025). Studying how LLMs can provide guidance to autotelic agents for automated scientific discoveries is a promising future direction.

6.3 Conclusion

In our introductory chapter, we posed a central question: can usage-based, grounded theories of language acquisition—emphasizing intrinsically motivated sensorimotor interactions—be reconciled with distributional semantics approaches, and in particular LLMs? To explore this, we proposed augmenting LLMs with an embodiment: enabling them to interact with external environments and update their internal representations through these interactions. Our focus was on aligning LLMs' functional competence via autotelic RL, empowering them to select, pursue, and learn their own goals. Through this lens, we demonstrated that embodied, interaction-based learning holds significant promise—not only for grounding functional competence, but also for improving world modeling and metacognitive abilities in LLMs.

However, despite these encouraging results, the broader question remains open: can embodying LLMs truly bridge the gap between grounded developmental theories and purely statistical language models? In human development, linguistic structures and cognitive representations co-emerge from early sensorimotor experiences—long before verbal language is acquired. That is, grounding is not added after but is constitutive of learning itself. In contrast, the present research explored a posteriori grounding: aligning the internal representations of LLMs that were originally shaped through passive, statistical learning alone. This contrast raises a fundamental question for future research: Is such post hoc grounding sufficient—or must embodiment be integrated into pretraining itself? One possible direction lies in hybrid learning regimes, where LLMs alternate between imitation (e.g., causal language modeling) and interactive sensorimotor experience during training. Such approaches may offer a path forward in building language agents whose representations are meaningfully grounded in the world they act within.

Appendices

Appendix A

GLAM

Contents

A.1 Environments . . . 139
A.1.1 BabyAI . . . 139
A.1.2 BabyAI-Text . . . 140
A.2 Additional results . . . 140
A.2.1 Per-task success rate . . . 141
A.2.2 Averaging success rate over the tasks . . . 141
A.2.3 Textual vs symbolic representation . . . 141
A.2.4 Impact of pre-training . . . 142
A.2.5 Impact of the size of the LLM . . . 145
A.2.6 Impact of varying action space and distractors . . . 146
A.2.7 Robustness to domain-specific vocabulary . . . 148
A.3 Evolution of actions distribution on evaluation prompts . . . 149
A.4 Generalization tests details . . . 153
A.4.1 Recapitulating results table . . . 153
A.4.2 Complementary tests for Q2 . . . 153
A.4.3 Complementary tests for Q3 . . . 154
A.4.4 LLM grounding of temporal symbols: "then" and "after" . . . 155
A.5 Distributed experimental setup . . . 155
A.6 Fine-tuning details . . . 156
A.6.1 PPO fine-tuning details . . . 156
A.6.2 Behavioral Cloning . . . 157
A.7 Confidence interval . . . 157
A.7.1 Confidence intervals for GFlan-T5, Flan-T5 and DRRN . . . 158
A.7.2 Confidence intervals for random agents . . . 159
A.8 Word substitutions for generalization tests . . . 159
A.8.1 Out of vocabulary . . . 159
A.8.2 Invented words . . . 159
A.8.3 Synonym actions . . . 160
A.8.4 Translation to French . . . 160

A.1 Environments

We extend the BabyAI platform (Chevalier-Boisvert et al., 2019) and create a text-only version named BabyAI-Text that encapsulates BabyAI and returns linguistic observations. Figure A.1 explains our environment.

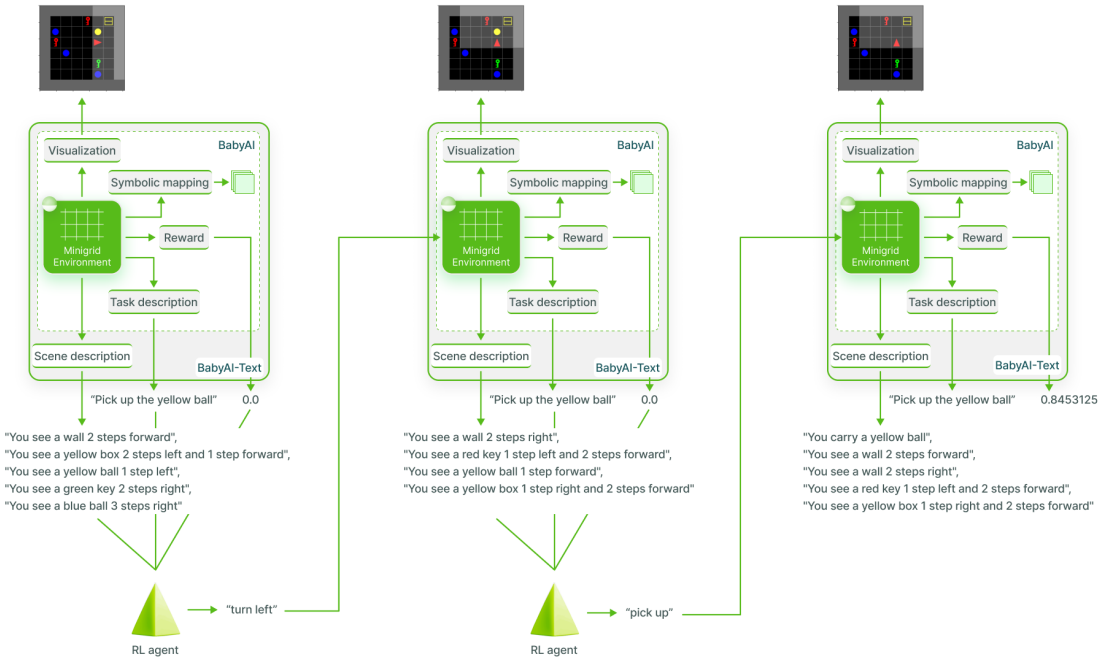


Figure A.1: An illustration of how our BabyAI-Text environment encapsulates BabyAI. We keep the inner minigrid environment as well as task descriptions and reward but map the partial view of the agent to a text description.

A.1.1 BabyAI

BabyAI Chevalier-Boisvert et al. (2019) is a language-conditioned environment where the agent has a limited number of steps to complete a language goal. This platform relies on a gridworld environment (MiniGrid) to generate a set of complex instructions-following environments. It has been specifically designed for research on grounded language learning and related sample efficiency problems. The gridworld environment is populated with the agent and objects (of 6 possible colors): boxes, balls, doors, and keys. These entities are placed in rooms of 8×8 tiles that are connected by doors that can be locked or closed. The grid is procedurally generated (i.e. objects populating an episode are randomly chosen and their position, as well as the agent's position, are also random). Some of the objects are useful for the task to achieve, while others are considered as distractors (objects can't be crossed, the agent has to either bypass them or move them). The agent can do 6 primitive actions: `turn left`, `turn right`, `go forward`, `toggle`, `pick up` to solve the language instruction (for instance `Pick up the red box`). To observe its environment, the agent has access to a partial view (i.e. it only sees the objects that belong to the

6×6 grid in front of it). BabyAI proposed to access this partial view through a symbolic mapping that returns 3 matrices of size 6×6. The first matrix contains which object is in the observed cells, the second gives the color of these objects, and the last one their state (e.g. locked, open). When the agent completes the task after N steps, it receives the reward $r_N = 1 - 0.9\frac{N}{H}$, where H is the maximum number of steps. During training, we multiply all rewards by 20 to ensure a good propagation of the rewards as per (Mirchandani et al., 2021). If the agent has not completed the task in the current step, the reward is 0. Additionally, BabyAI also provides visualization tools for experimenters to observe the grid and better grasp agents' behaviors.

A.1.2 BabyAI-Text

BabyAI-Text is a textual environment that encapsulates BabyAI and provides a description of each observation instead of a symbolic representation. A textual description consists of a list of template descriptions with the following structure:

- "You see a <*object*> <*location*>" if the object is a key, a ball, a box or a wall.
- "You see a(n) *open/closed* door <*location*>" , if the agent sees a door.
- "You carry a <*object*>", if the agent carries an object.

The <*object*>, is composed of an *adjective* (among 6 possible colours: red, blue, green, yellow, grey, purple) and a noun (among 4 possible: key, door, box, ball). The <*location*> is given as the number of steps *right*, *left*, and or *forward* from the agent to the object. We illustrate this in the leftmost observation of Figure A.1 where the "yellow box" is "2 steps left and 1 step forward" from the agent (the red triangle). Thus an object described as "1 step forward" is right in front of the agent that does not need to *go forward* if it wants to pick that object. Walls of the room are the only spatially extended objects in BabyAI-Text. We give their location at the closest distance to the agent. See the leftmost image of Figure A.1 for an example where the agent sees a wall "2 steps forward" and another wall "2 steps left". All of the choices for describing the environment constitute what we call the geometry of the environment, that the agent has to ground in order to succeed in the task. The presence of a fine grained geometry (with distances in steps to the different object in the room) is one of the main differences from other textual games such as TextWorld or ScienceWorld where all objects in a room are not spatially described.

Thanks to this extension, BabyAI-Text resembles a TextWorld (i.e. provides text descriptions of the observation and executes text commands) while keeping the inner minigrid environment along with BabyAI's tasks and visualization tools. Moreover, as our extension simply provides an alternative mapping of observations, one can both use and compare agents that either expect text-only observations (with BabyAI-Text) or symbolic observations (with BabyAI).

A.2 Additional results

A.2.1 Per-task success rate

In order to get a better understanding of our agent's capabilities, we report in Figure A.2 the success rate on each task from our "no-change" evaluation in Figure 3.5 (assessing the post-training performance of agents on 1000 test episodes of our mixed setup) of Flan-T5 and GFlan-T5. These results (with 4 seeds after 1.5 million training steps) show that our functional grounding leads GFlan-T5 to master the *GoTo* and *PickUp* tasks while improving results on *PutNextTo* and *PickupThen/AfterPickup*. However, GFlan-T5 has not found yet any robust strategy for the *OpenDoor* task (being the hardest as the agent must find the right key and discover that the action "toggle" opens the door) in the relatively short allocated time.

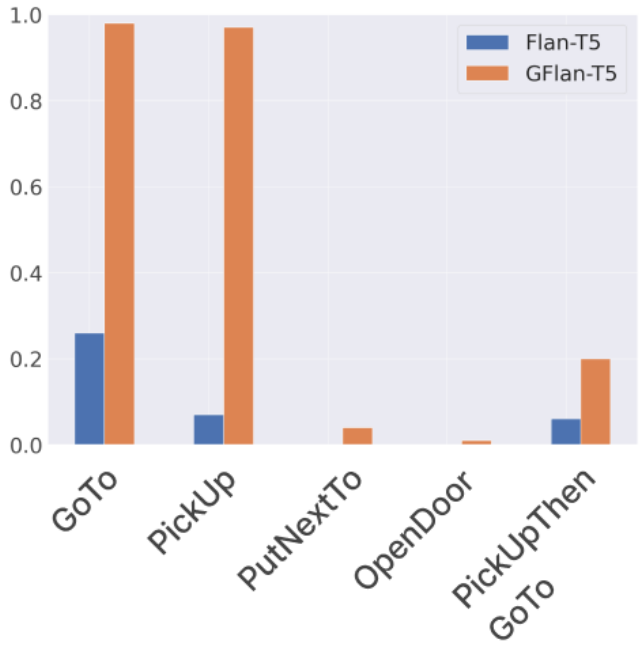


Figure A.2: Per-task success rate for the 1000 evaluation trajectories performed in Figure 3.5.

A.2.2 Averaging success rate over the tasks

In our environments, goals are divided into five types of tasks of varying difficulty. Each goal is sampled randomly from the tasks when the environment is reset. We obtain the success rate for the goals at update u by averaging over the completed trajectories during the collection phase. As several environments run in parallel, goals from easier tasks that are completed more quickly, such as *GoTo* and *PickUp*, tend to be more represented in the buffer.

A.2.3 Textual vs symbolic representation

In order to understand how the structure of the observation (i.e. either symbolic image using 3 matrices containing integers defining respectively the object seen, its color

and property if any or text) influences the success rate of an RL agent, we compare the DRRN and Symbolic-PPO respectively trained on BabyAI-Text and BabyAI on the `Go To Red Ball` task. In this task, the agent has to go in front of a red ball in 1 room without any distractor (i.e. the task never changes, only the position of agent and red ball do). The task has been voluntarily chosen as trivial so that the main difference only comes from the way the information is given to the agent. Both the DRRN and Symbolic-PPO agents have a similar number of parameters (1M), they both use recurrent layers to deal with partial observability and use the canonical action space.

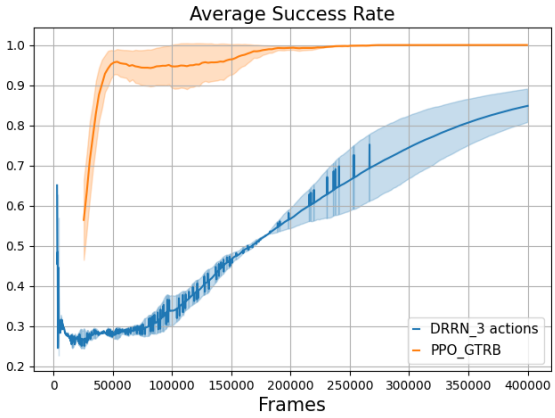


Figure A.3: Average success rate for DRRN and Symbolic-PPO on the *Go To Red Ball* task with standard deviation over two random seeds. The PPO receives symbolic information and the DRRN gets textual observations.

Contrary to what one might assume in Figure A.3 the PPO agent converges faster than the DRRN agent on this trivial task. Thus, symbolic observations make the learning easier for the agent. We conclude that even if language contains high-level information, understanding the link between spatial information and language is far more difficult than using symbolic information given in a matrix. Indeed, the matrix already contains a geometric bias favorable to the agent. We also want to point out that the DRRN is an off-policy RL method compared to PPO (which is on-policy) and that consequently, the DRRN was expected to be, by-design, more sample efficient.

A.2.4 Impact of pre-training

We test how pre-training allows efficient fine-tuning of our LLM. We vary which weights of Flan-T5 are kept pre-trained as well as how we compute actions' probability (i.e. either using our method reusing language modeling heads or using new action heads with an MLP). We evaluate the performance of 5 models:

- The full LLM is pre-trained and language modeling heads are used for actions probability: GFlan-T5 (Figure A.4)

- The full LLM is pre-trained and new action heads are used: AFlan-T5 (Figure A.5)

• Only the embedding layer's weights are kept pre-trained (the rest of the LLM is randomly initialized) and new action heads are used: NPAE-Flan-T5 (Figure A.6)

• Only the embedding layer's weights are kept pre-trained (the rest of the LLM is randomly initialized) and the (randomly initialized) language modeling heads are used for actions' probability: **NPE-FlanT5** (Figure A.7)

• All LLM's weights are randomly initialized and action heads are used: **NPA-Flan-T5** (Figure A.8)

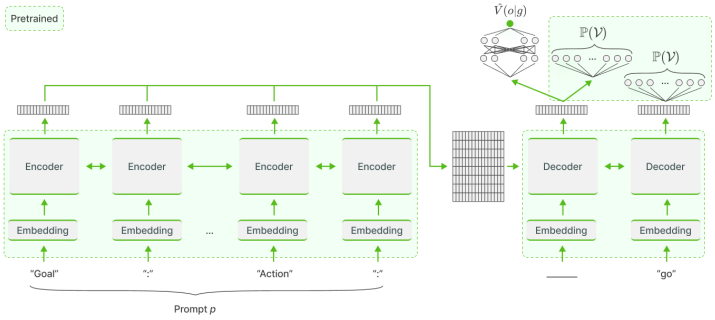


Figure A.4: GFlan-T5: We use the Flan-T5 architecture and add a value head. We initialize the agent with the pre-trained weights (framed in green in the diagram) including its language modeling heads to compute action probabilities. The weights of the value head are initialized randomly. GFlan-T5 stands for: grounded Flan-T5.

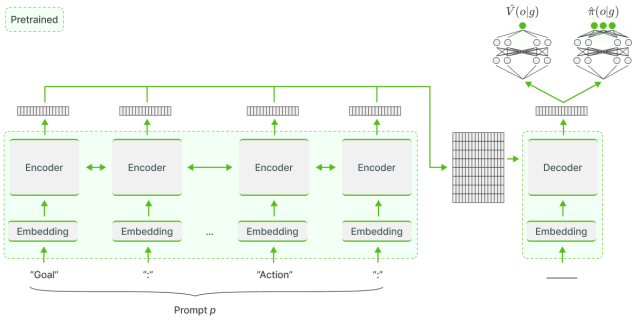


Figure A.5: AFlan-T5: We use the Flan-T5 architecture but replace the language modeling heads with action heads (that return the probability for each action) and add a value head. We initialize the embedding, the encoder and decoder parts of the agent with the pre-trained weights (framed in green in the diagram) and the other weights randomly. AFlan-T5 stands for action heads Flan-T5.

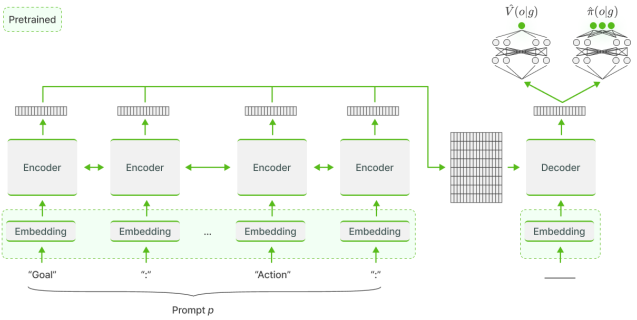


Figure A.6: NPAE-Flan-T5: We use the Flan-T5 architecture but replace the language modeling heads by action heads and add a value head. We initialize the embedding with the pre-trained weights (framed in green in the diagram) and the other weights randomly. NPAE-Flan-T5 stands for: non-pre-trained with action heads and pre-trained embedding Flan-T5.

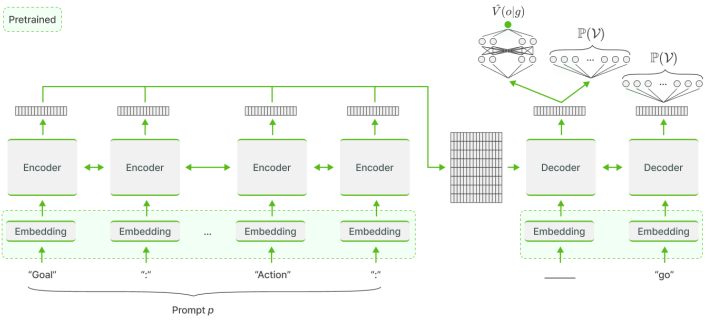


Figure A.7: NPE-Flan-T5: We use the Flan-T5 architecture and add a value head. We initialize the embedding with the pre-trained weights (framed in green in the diagram) and the other weights randomly. NPE-Flan-T5 stands for: non-pre-trained with pre-trained embedding Flan-T5.

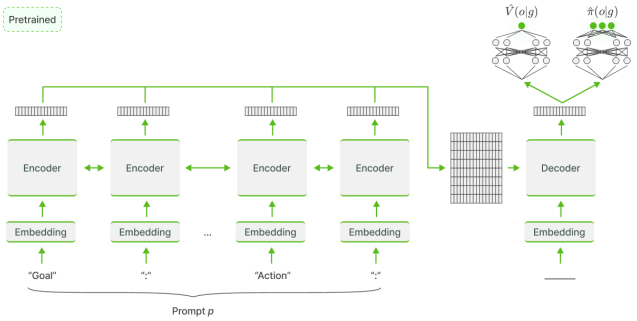


Figure A.8: NPA-Flan-T5: We use the Flan-T5 architecture but replace the language modeling heads with action heads and add a value head. We initialize all the weights randomly. NPA-Flan-T5 stands for: non-pre-trained with action heads Flan-T5.

Figure A.9 compares the training curves of the agents above on the task *Go To <object>*. GFlan-T5 has unsurprisingly the best results as it is fully pre-trained. More surprisingly, AFlan-T5 takes more steps than expected to perform better than the non-pre-trained networks (250000 frames). We hypothesize that during the pertaining, the last transformer layer encodes information in a space designed for language modeling heads ($\approx$ 32000 heads) which is not convenient for the non-pre-trained 6 actions heads. Indeed, AFlan-T5 has to make sense of this space before getting the benefits of having the rest of the network trained. This could explain why it suddenly performs better after 250000 steps. Comparing NPAE, NPA and NPE Flan-T5, we see that the presence of an action head is crucial for non-pre-trained networks. Indeed, the NPE fails to learn in the given number of steps compared to NPAE and NPA that have similar learning curves. A possible explanation is that for NPE, the information flow that is backpropagated through the gradient is really small due to the huge number of language modeling heads and the few number of tokens updated (< 100). On the opposite, GFlan-T5, that also uses language modeling heads but is fully pre-trained, only needs a light fine-tuning for the necessary tokens explaining its high success rate and sample efficiency.

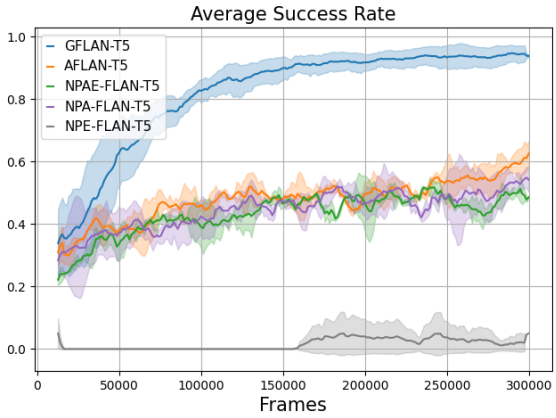


Figure A.9: Average success rate of varying pre-trained weights and scoring method with standard deviation over two random seeds. We train all LLMs on the *Go to <object>* task in 1 room, with 8 distractors, the 6 canonical actions and using Flan-T5 large (780 million parameters) as architecture.

A.2.5 Impact of the size of the LLM

The capacities of LLMs depend strongly on their size and many properties of these networks only appear when they are large enough (Chowdhery et al., 2022; Wei et al., 2022). We consequently test the influence of the size of the LLM on our results by training 3 different GFlan-T5 (as well as the DRRN and Symbolic-PPO baselines) on the *Go to <object>* task for 400.000 steps: GFlan-T5 **small** (80 million parameters), GFlan-T5 **large** (780 million parameters) and GFlan-T5 **XL** (3 billion parameters).

We show the evolution of average success rate over 2 seeds in Figure A.10 highlighting that pre-training prior knowledge only looks impactful when the network is large enough. The difference between the learning properties of small and large models relates to the

definition of an emergent behavior given by Wei et al. (2022): *"an ability is emergent if it is not present in smaller models but is present in larger models"*. Beyond the data on which a model has been trained, the size of this model seems crucial for the acquisition of new knowledge about relations between entities during the fine-tuning phase.

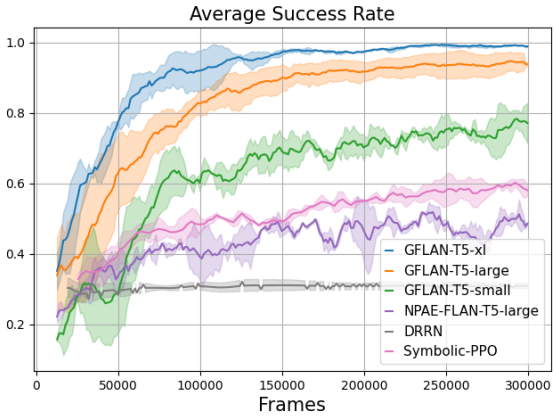


Figure A.10: Impact of the size of the LLM on online RL fine-tuning. We conduct the tests with the *Go to <object>* task in 1 room, with 8 distractors. We measure the evolution of average success rate over 2 seeds with standard deviation for GFlan-T5 **small** (80 million parameters), GFlan-T5 **large** (780 million parameters) and GFlan-T5 **XL** (3 billion parameters). DRRN, NPAE-Flan-T5-large and Symbolic-PPO are given as baselines.

A.2.6 Impact of varying action space and distractors

In this section, we detail the studies about the impact of varying the action space and the number of distractors.

Impact of the dimension of the action space

One of the expected advantages of pre-trained LLMs in RL is that they avoid the *Tabula-Rasa* paradigm and already have useful biases. In these experiments, we test the sensibility of LLMs to the size of the action space by using 3 different action spaces (**restricted**, **canonical**, **augmented**) when trained on the *Go to <object>* task.

We conduct the tests in an environment with 1 room, 8 distractors and in Figure A.11 report full learning curves used to draw Figure 3.4a. We show that GFlan-T5 efficiently handles the different action spaces compared to the other agents. Its initial biases are particularly helpful when the action space is large. Indeed, when we look at the difference of sucess rate between GFlan-T5 and the second best-performing agent after the 50000 first steps, there is almost no difference in the restricted settings and 0.35 in the augmented settings. That supports the hypothesis that the results are due to LLMs' ability to discard useless action quickly at the beginning of fine-tuning.

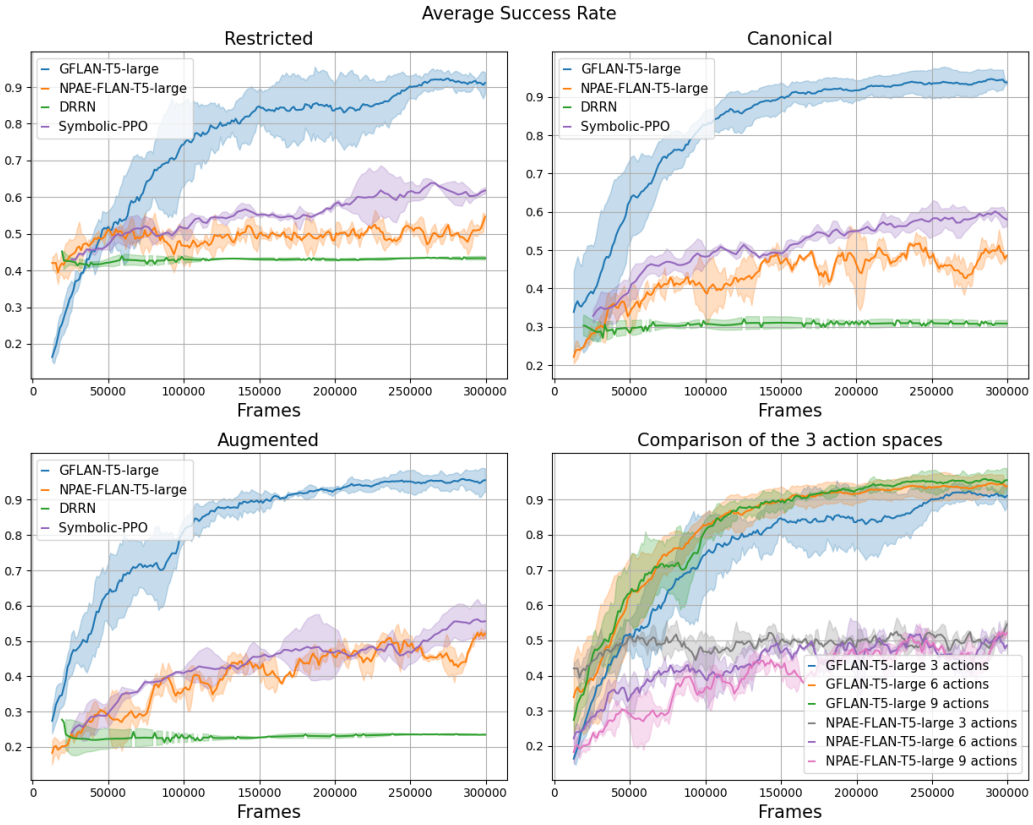


Figure A.11: Learning curves for the agents on the *Go To* task for different sizes of action space (Restricted: 3 actions, Canonical: 6, Augmented: 9, with only the 3 actions that are useful). The success rate is given over 2 seeds along with standard deviation.

Impact of the number of distractors

In Figure 3.4b we have shown that LLMs are less sensitive to variations on task complexity by plotting the evolution of sample efficiency (Equation (3.3)) for different numbers of distractors: 4, 8 and 16. In Figure A.12 we report the full learning curves. We observe that Symbolic-PPO's performances collapse when we go from 4 to 16 distractors whereas GFlan-T5's performances remain similar. NPAE-Flan-T5's performances are also non-affected by the change in the number of distractors but in this case we suppose it is because it cannot learn the task in 400000 steps.

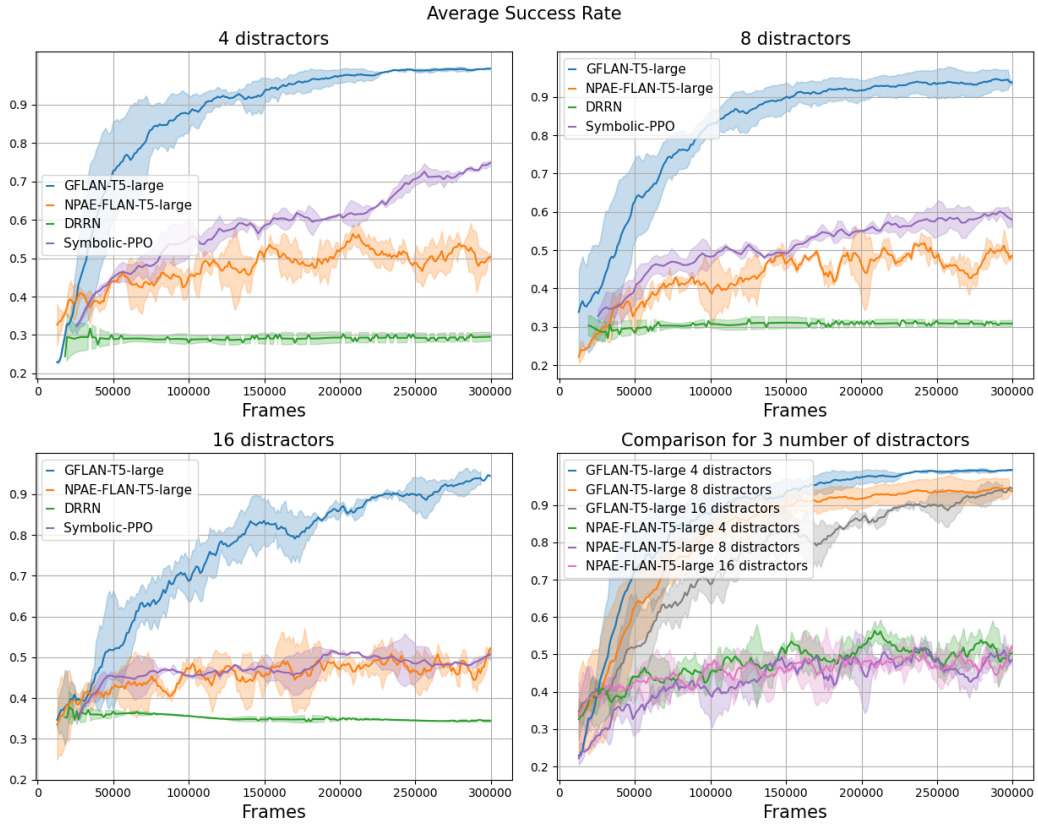


Figure A.12: Learning curves for the agents on the *Go To* task for different number of distractors (4, 8, 16). The success rate is given over 2 seeds with standard deviation.

A.2.7 Robustness to domain-specific vocabulary

In Section 3.2.4, we have shown the robustness of our method to random vocabulary for words that do not influence the grounding of actions (in our case, the objects and their colors). Nonetheless, one can imagine an environment with a specific vocabulary where common words are used to describe particular technical terms with possibly very different meanings. To verify the impact of such environment on our training process, we trained GFlan-T5 on the *GoTo* task where the actions "turn left" and "turn right" are flipped (i.e. using "turn left" makes the agent rotate to the right, and the opposite for "turn right"). Figure A.13 shows that, while the prior knowledge of the LLM leads to poorer performance at the very beginning of training (as the LLM must learn to rotate left and right), GFlan-T5 converges at a similar speed than in the non-flipped environment. This result hints robustness that our grounding method for LLMs can adapt to domain-specific vocabularies.

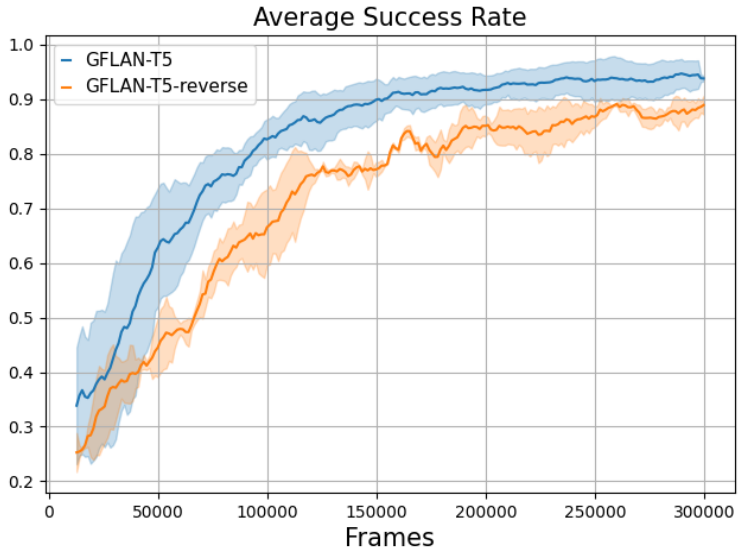


Figure A.13: Comparison of the average success rate over training for GFlan-T5 on the *Go To* when the actions "turn left" and "turn right" are flipped ("-reverse"). The success rate is given over 2 seeds with standard deviation.

A.3 Evolution of actions distribution on evaluation prompts

To better grasp the skill acquisition dynamics when performing online RL grounding on GFlan-T5 in the multi-task setting of Section 3.2.4, we test at each update the LLM on 11 prompts listed in Table A.1. We plot in Figure A.14 the evolution of action probabilities outputted by our LLM aiming to partially decipher the changes in the LLM and visualize which skill is acquired when.

Prompts 0 and 1 are simple navigation tasks. The agent has to move in the direction given in the prompt. Looking at the corresponding plots we observe two things: first, the optimal behavior is learned in less than a hundred of updates, even for prompt 1 for which the bias at the beginning is both wrong and high. Second, from the beginning, only the navigation actions (`turn left`, `turn right`, `go forward`) relevant for the *Go To <object>* task have a high probability. Therefore, the Flan-T5 780M seems to already have useful biases for navigation and is able to quickly update or correct them through interactions.

We observe similar useful biases with the *Pick Up <object>* task (using prompts 5 and 6). Indeed, at the beginning, both the `pick up` action and navigation actions already have a high probability.

We can see that the agent struggles to ground the geometry of the environment with prompts 6 and 7. Indeed, it has to understand that an object described at "1 step forward" is in front of it such that it can pick it up or drop it directly without moving further. While GFlan-T5 eventually seems to understand it, it still shows some hesitation as proven by the fact that it gives almost the same probability to `go forward` and `pick up` or `drop` for the prompt 7 at the end of training (see Figure A.14).

We also verify how GFlan-T5 understands temporal constructions such as doing an action A then an action B (prompt 8) or doing an action A after doing an action B (prompt 9). These two test prompts are exactly the same except for the goal where prompt 8 uses "then" and prompt 9 uses "after" to link the two actions. We observe that when the order of actions in the task specification is the same as the one the agent has to do (i.e. prompt 8), the LLM quickly and learns to choose the right action even if during the learning it loses its ability (with `turn left` and `turn right` that are almost at the same probability. However, when the order of actions mentioned in the goal specification is reversed (prompt 9), the LLM ends up favoring the wrong direction and exhibits much more hesitation from the beginning of the training. This qualitative observation concurs with the measure of success rate given in Appendix A.4.4.

The prompts 2, 3 and 4 show that the agent has difficulties with the task *Open <door>*. This task is fairly complex since the agent has to infer that a key of the same color as the closed door is required to open it. In the given training budget, the agent fails to associate the need of a key with the task.

Finally, we test the agent on a task that is not seen during training. It is the generalization task *Pick up <object A> then/after Pick up <object B>* from Q3 Section 3.2.6, composed from two tasks seen during training *Pick up* and *Pick up then Go To* (prompt 10). The prompt is built such that the agent has accomplished half of the instruction and has to drop the object it carries in order to pick another one. The action `drop` is the optimal one because it is the only one that allows the agent to complete the goal in a minimum number of steps. Between the updates 400 and 600 the agent begins to increase the probability of the `drop` action. This change is correlated to the change of distribution in prompt 7. It can be interpreted as the fact that the action `drop` begins to be grounded after 800 updates.

Table A.1: Test prompts. The prompts' header (*Possible action of the agent: turn left, turn right, go forward, pick up, drop, toggle*) is not shown below as it remains the same for all prompts

Ids	Tasks	Prompts	Comments
0	Go To <object>	**Goal of the agent**: go to the green ball **Observation 0**: You see a wall 2 step left, You see a purple key 1 step left and 2 steps forward, You see a yellow key 1 step left and 1 step forward, You see a green ball 3 steps forward, You see a grey ball 1 step right and 5 steps forward, You see a green key 1 step right and 2 steps forward, You see a grey ball 1 step right and 1 step forward, You see a green key 2 steps right and 4 steps forward, You see a red box 2 steps right and 2 steps forward, **Action 0**: **Expected answer: go forward**	Simple navigation task.
1	Go To <object>	**Goal of the agent**: go to the green ball **Observation 0**: You see a wall 2 step left, You see a purple key 1 step left and 2 steps forward, You see a yellow key 1 step left and 1 step forward, You see a green ball 3 steps forward, You see a grey ball 1 step right and 5 steps forward, You see a green key 1 step right and 2 steps forward, You see a grey ball 1 step right and 1 step forward, You see a green key 2 steps right and 4 steps forward, You see a red box 2 steps right and 2 steps forward, **Action 0**: go forward **Observation 1**: You see a purple key 1 step left and 1 step forward, You see a yellow key 1 step left, You see a green ball 2 steps forward,	Simple navigation task.

Continued on next page

Table A.1 – continued from previous page

Id	Task	Prompt	Comments
		You see a grey ball 1 step right and 4 steps forward, You see a green key 1 step right and 1 step forward, You see a grey ball 1 step right, You see a green key 2 steps right and 3 steps forward, You see a red box 2 steps right and 1 step forward, **Action 1**: turn right **Observation 2**: You see a wall 2 step right, You see a green key 3 steps left and 2 steps forward, You see a green ball 2 steps left, You see a red box 1 step left and 2 steps forward, You see a green key 1 step left and 1 step forward, You see a grey ball 1 step forward, **Action 2**: **Expected answer: turn left**	
2	Open <adj> door	**Goal of the agent**: open the purple door **Observation 0**: You see a wall 3 steps forward, You see a wall 3 steps left, You see a yellow key 1 step right and 1 step forward, You see a locked purple door 2 steps right and 3 steps forward, You see a purple ball 3 steps right and 1 step forward, You see a green box 3 steps right, You see a purple key 2 steps left, **Action 0**: **Expected answer: turn left**	Inference task The agent has to infer that a key of the same color is needed and moves toward it.
3	Open <adj> door	**Goal of the agent**: open the purple door **Observation 0**: You see a wall 3 steps forward, You see a wall 3 steps left, You see a yellow key 1 step right and 1 step forward, You see a locked purple door 2 steps right and 3 steps forward, You see a purple ball 3 steps right and 1 step forward, You see a green box 3 steps right, You see a purple key 2 steps left, **Action 0**: turn left **Observation 1**: You see a wall 3 steps forward, You see a wall 3 steps right, You see a purple key 2 steps forward, **Action 1**: go forward **Observation 2**: You see a wall 2 steps forward, You see a wall 3 steps right, You see a purple key 1 step forward, **Action 2**: **Expected answer: pick up**	Inference task The agent has to infer that a key of the same color is needed and pick it up.
4	Open <adj> door	**Goal of the agent**: open the purple door **Observation 0**: You carry a purple key, You see a wall 3 steps forward, You see a wall 5 steps left, You see a yellow key 1 step left and 1 step forward, You see a locked purple door 3 steps forward, You see a purple ball 1 step right and 1 step forward, You see a green box 1 step right, **Action 0**: go forward **Observation 1**: You carry a purple key, You see a wall 2 steps forward, You see a wall 5 steps left, You see a yellow key 1 step left, You see a locked purple door 2 steps forward, You see a purple ball 1 step right, **Action 1**: go forward **Observation 2**: You carry a purple key, You see a wall 1 step forward, You see a wall 5 steps left, You see a locked purple door 1 step forward, **Action 2**: **Expected answer: toggle**	Inference task The agent has to infer that you can open a closed door by toggling it while having a key of the same color.
5	Pick up <object>	**Goal of the agent**: pick up green box **Observation 0**: You see a wall 2 steps forward, You see a wall 2 steps left, You see a yellow ball 1 step left and 1 step forward, You see a green box 2 steps right, **Action 0**: **Expected answer: turn right**	The agent has to reuse knowledge from navigation task.
6	Pick up <object>	**Goal of the agent**: pick up green box **Observation 0**: You see a wall 2 steps forward, You see a wall 2 steps left, You see a yellow ball 1 step left and 1 step forward, You see a green box 2 steps right, **Action 0**: turn right **Observation 1**: You see a wall 2 steps left, You see a blue key 1 step right, You see a red ball 2 steps right and 1 step forward, You see a green box 2 steps forward, **Action 1**: go forward	The agent has to reuse knowledge from navigation task. and understand the geometry of the room.

Continued on next page

Table A.1 – continued from previous page

Id	Task	Prompt	Comments
		Observation 2: You see a wall 2 steps left, You see a red ball 2 steps right, You see a green box 1 step forward, **Action 2**: **Expected answer: pick up**	
7	Put <object A> next to <object B>	**Goal of the agent**: put blue ball next to red box **Observation 0**: You carry a blue ball, You see a wall 5 steps forward, You see a wall 2 steps left, You see a grey key 1 step right and 2 steps forward, You see a red box 3 steps forward, **Action 0**: go forward **Observation 1**: You carry a blue ball, You see a wall 4 steps forward, You see a wall 2 steps left, You see a grey key 1 step right and 1 step forward, You see a red box 2 steps forward, **Action 1**: **Expected answer: drop**	The agent has to reuse knowledge from navigation and understand the geometry of the room.
8	Pick up <object A> then go to <object B>	**Goal of the agent**: pick up the blue ball then go to the red box **Observation 0**: You see a wall 3 steps forward, You see a wall 4 steps right, You see a purple key 2 steps forward, You see a red box 2 steps right, You see a blue ball 2 steps left, **Action 0**: **Expected answer: turn left**	Prompt 8 and 9 test the ability of the agent to understand temporal concepts.
9	Go to <object B> after you pick up <object A>	**Goal of the agent**: go to the red box after you pick up the blue ball **Observation 0**: You see a wall 3 steps forward, You see a wall 4 steps right, You see a purple key 2 steps forward, You see a red box 2 steps right, You see a blue ball 2 steps left, **Action 0**: **Expected answer: turn left**	Same as prompt 8 but sentence action order different from execution order.
10	Pick up <object A> then pick up <object B>	**Goal of the agent**: pick up the green key then pick up the red box **Observation 0**: You carry a green key, You see a wall 4 steps forward, You see a wall 4 steps left, You see a red box 1 step left, You see a purple ball 2 steps left and 1 step forward, **Action 0**: **Expected answer: drop**	Task never seen in training to analyze generalization.

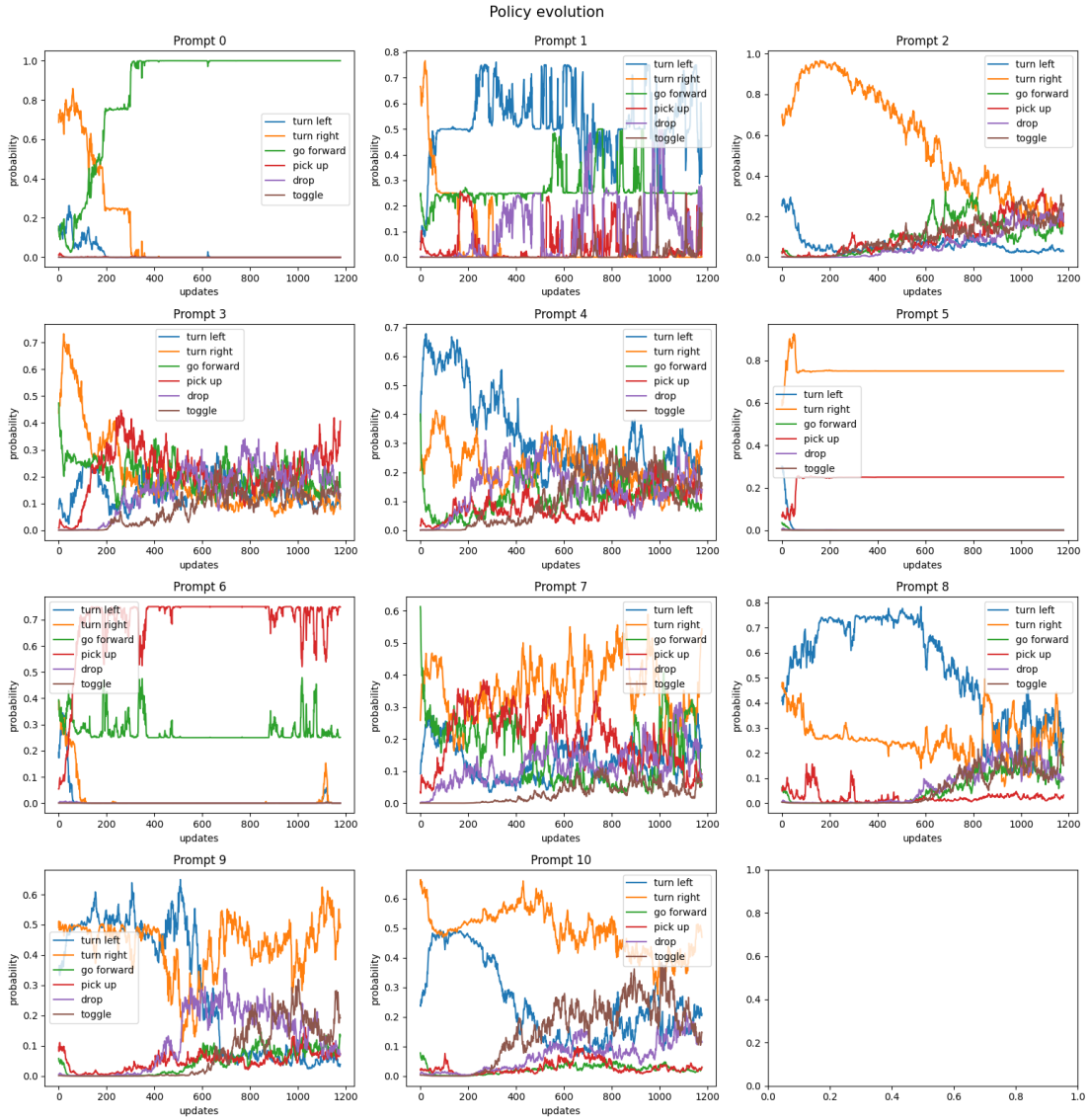


Figure A.14: Evolution of actions' probability over training for test prompts listed in Table A.1.

A.4 Generalization tests details

A.4.1 Recapitulating results table

In this section we summarize the numerical results shown in Figure 3.5 with the confidence intervals calculated as explained in Appendix A.7.

A.4.2 Complementary tests for Q2

To further analyze results from in Section 3.2.5, we conduct more systematic tests on different aspects of the generalization to new words. The results are given in Table A.3.

Table A.2: Generalization tests

	Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Q2	Mix - no change	0.89 ± 0.05	0.11 ± 0.03	0.17 ± 0.04	0.14 ± 0.02	0.15 ± 0.05
	Mix - out-of-vocabulary nouns	0.87 ± 0.05	0.09 ± 0.02	0.16 ± 0.05	0.15 ± 0.00	
	Mix - invented nouns and adjectives	0.88 ± 0.06	0.11 ± 0.03	0.16 ± 0.03	0.16 ± 0.00	
Q3	Pick up then/after pick up	0.12 ± 0.06	0.02 ± 0.00	0.06 ± 0.01	0.06 ± 0.03	0.05 ± 0.05
	Mix - synonym actions	0.12 ± 0.12	0.02 ± 0.00	0.16 ± 0.04	0.17 ± 0.04	0.15 ± 0.05
	Go To - English	0.99 ± 0.01	0.27 ± 0.03	0.31 ± 0.04	0.31 ± 0.03	0.30 ± 0.05
	Go To - French	0.02 ± 0.01	0.03 ± 0.00	0.30 ± 0.02	0.31 ± 0.02	

Table A.3: Complementary tests for Q2

Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Mix - no change	0.89 ± 0.05	0.11 ± 0.03	0.17 ± 0.02	0.14 ± 0.02	0.15 ± 0.05
Mix - unseen in-vocabulary objects	0.87 ± 0.03	0.12 ± 0.03	0.16 ± 0.08	0.17 ± 0.09	
Mix - out-of-vocabulary adjectives	0.87 ± 0.07	0.16 ± 0.03	0.16 ± 0.02	0.16 ± 0.01	

Unseen in-vocabulary objects – During training we remove tasks whose goal contain the following objects: yellow box, red key, red door, green ball and, grey door. Nonetheless, the agent can have these objects as distractors and so have seen them during training. We assess how our agents perform on the mix of tasks with goals using only these objects. The success rate of 0.87 points out that GFlab-T5 is unaffected by the use of unseen in-vocabulary objects.

Unseen out-of-vocabulary adjectives – We perform the same test as for out-of-vocabulary nouns in Section 3.2.5 but this time with adjectives that do not belong to the BabyAI-Text vocabulary. We generate the prompt by exchanging the adjectives with predefined synonyms (see Table A.8). Similarly to the test with out-of-vocabulary nouns, the test with out-of-vocabulary adjectives reveals that GFlan-T5 is unaffected by this change. Indeed, the success rate is of 0.87 compared to the one of mix of tasks without change at 0.89.

A.4.3 Complementary tests for Q3

In Section 3.2.6, we observe that GFlan-T5 fails to generalize to an environment where we change the language. We hypothesize that such a change modifies too many grounded symbols at once. To verify this hypothesis, we test a middle-ground version, where we keep the environment in English but actions are in French. In this setting, Table A.4 shows that the success rate of the agent (0.15) is better than the fully french environment (0.02). This observation supports that fine-tuned agents tend to generalize to related

Table A.4: Complementary tests for Q3

Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Go To - English	0.99 ± 0.01	0.27 ± 0.03	0.31 ± 0.04	0.31 ± 0.03	0.30 ± 0.05
Go To - French	0.02 ± 0.01	0.03 ± 0.00	0.30 ± 0.02	0.31 ± 0.02	
Go To - English with actions in French	0.15 ± 0.04	0.26 ± 0.02	0.31 ± 0.01	0.33 ± 0.00	

words in other languages. Nonetheless, this ability seems highly dependent on the number of grounded words we modify.

A.4.4 LLM grounding of temporal symbols: "then" and "after"

In this experiment we observe the dynamics of functional grounding of instructions containing the temporal symbols "then" and "after" using the tasks: *Pick up <object A> then go to <object B>* and *Go to <object B> after pick up <object A>*. As the order of the action matters to have the task considered completed, a correct grounding of these symbols is crucial. Table A.5 shows that GFlan-T5 has a better grounding of these words than the original Flan-T5 agent. Moreover, we observe a slight bias after fine-tuning: the agent has stronger performances for the tasks with "then" (success rate of 0.22) compared to the tasks with "after" (success rate of 0.17). We hypothesize it is easier to ground the word "then" because the order of the actions the agent must do is the same as the order in which the actions appear in the instructions. A qualitative example of this behavior is given in Appendix A.3 (prompts 8, 9).

Table A.5: Test on tasks with temporal components

Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Mix of tasks then/after	0.23 ± 0.06	0.12 ± 0.01	0.09 ± 0.01	0.09 ± 0.02	0.04 ± 0.05
Tasks with then only	0.22 ± 0.11	0.12 ± 0.01	0.10 ± 0.003	0.10 ± 0.02	
Tasks with after only	0.17 ± 0.05	0.13 ± 0.05	0.10 ± 0.03	0.10 ± 0.01	

A.5 Distributed experimental setup

In order to accelerate our online RL fine-tuning, we first leverage a classic distributed data collection setup where 32 BabyAI-Text environments are running in parallel (all on CPUs). Our environments are run in a synchronous way, meaning that at every step, we get 32 current states and need to send 32 actions back to the environment. In very classic RL setups, policy networks are usually small and we simply batch the 32 states, feed them to the network and obtain the 32 actions' probability before sampling from them and choosing one action per environment. However, as explained in Section 3.2.2, our method requires $|\mathcal{A}|$ forward passes on a potentially very large and computationally expensive LLM in order to compute actions' probability for a single environment. Hence,

we now need $32 \times |\mathcal{A}|$ forward passes for a single step in all environments, which can easily become a huge bottleneck in our training process.

To overcome this, we deploy for each of our experiments in Section 3.2.3 4 instances of our LLM all running in parallel. We load and use LLMs through the Hugging Face Transformers Python library[1]. Our method relies on a simple client-server architecture where the RL script acts as a client sending requests to LLMs. This client communicates with a master server which dispatches the call over multiple servers (i.e. one per LLM). Once each LLM has computed its subset of the call, the master gathers results and sends the response to the RL client. We use Pytorch Distributed[2] with the GLOO backend for communication (hence possible both on CPU-only and GPU setups). We wrap all these in a Python library called *Lamorel* which can dispatch calls over the deployed LLMs from a single line of code in the RL loop asking for actions' probability for all environments. Using this method, we observe a quasi-linear scaling with the number of deployed LLMs.

Once transitions have been collected, we update our LLM using the PPO loss. For this, *Lamorel* helps parallelize the gradients' computation with a Distributed Data Parallelism[3] setup where forward and backward passes over transitions are also dispatched on the different instances of our LLMs. Then, *Lamorel* helps gather gradients and update each LLM (as well as their value head) the same way. In addition, *Lamorel* also helps define a custom computational graph linked to the LLM. We use this to add MLPs on top of our Flan-T5 model for the value head (see experiments with action heads in Section A.2.4).

When using **Flan-T5 780M**, each LLM instance is distributed (Vertical Model Parallelism[4]) 2 Nvidia A100 80GB GPUs requiring thus a total of 8 Nvidia A100 80GB GPUs to run an experiment (2 GPUs $\times$4 LLM instances). For **Flan-T5 80M** and **Flan-T5 3B**, we respectively use 1 Nvidia V100 32GB and 4 Nvidia A100 80GB per LLM instance.

In total, to conduct experiments and ablations we use 160 GPU.hours on the Nvidia V100 32G and 18880 GPU.hours on Nvidia A100 80GB.

A.6 Fine-tuning details

A.6.1 PPO fine-tuning details

We reused PPO's hyperparameters from Ramamurthy et al. (2023) and did not perform any further tuning (see Table A.6). We used an Adam (Kingma & Ba, 2015) optimizer with the hyperparameters listed in Table A.7). For additional heads, we used MLPs with 3 hidden layers of 1024 units with Sigmoid activation.

[1] https://huggingface.co/docs/transformers/index
[2] https://pytorch.org/docs/stable/distributed.html
[3] https://pytorch.org/tutorials/intermediate/ddp_tutorial.html
[4] Layers are spread across GPUs (https://huggingface.co/docs/transformers/v4.15.0/parallelism)

Table A.6: PPO hyperparameters

Variable	Value
Number of transitions collected between two updates	1280 (32 environments ×40 steps in each environment)
Number of epochs per update	4
Batch size	64
Entropy loss coefficient	0.01
Value function loss coefficient	0.5
Discount factor	0.99
lr	1×10^{-6}
λ factor of the Generalized Advantage Estimator	0.99
Clipping parameter ϵ	0.2
Maximum gradient norm	0.5

Table A.7: Adam hyperparameters

Variable	Value
Learning rate	1×10^{-6}
ϵ	1×10^{-5}
β_1	0.9
β_2	0.999

A.6.2 Behavioral Cloning

In Section 3.2.7, we show how grounding using RL differs from BC. For this, we fine-tune **Flan-T5 780M** on 400.000 transitions collected on the *Go To <object>* task. As indicated in Table A.4, GFlan-T5 obtains a 0.81 success rate on the 1000 test episodes of the *Go To <object>* task. Hence by fine-tuning Flan-T5 to imitate GFlan-T5, one could expect an on-par performance (or worse, but not better). We therefore use GFlan-T5 to collect 400.000 transitions and fine-tune Flan-T5 using them. However, the stochasticity in the GFlan-T5 policy leads to deceptive transitions in the dataset (potentially harmful for BC). We thus also assess whether using optimal transitions to fine-tune Flan-T5 leads to better results than GFlan-T5. To collect optimal trajectories, we use the bot provided by BabyAI and also gather 400.000 transitions on the *Go To <object>* task.

For fine-tuning, we use causal language modeling with the same prompt as the one given to our LLM agents in Section 3.2.3 as input and the performed action as label. We use the same learning rate as the one used by Rae et al. (2022) to generate Flan-T5 (i.e. 5×10^{-4}) and perform a single epoch on the 400.000 examples.

A.7 Confidence interval

In sections 3.2.5 and 3.2.6, we perform several generalization tests. For each test we report the success rate over 2 seeds tested on 1000 episodes each. In the following, we explain how we get the 99% confidence interval.

A.7.1 Confidence intervals for GFlan-T5, Flan-T5 and DRRN

We model the success of an agent, trained with the seed i, on a task (i.e. episode with its associated task) using a Bernoulli variable $X^i \sim \mathcal{B}(p_i)$, with p_i the probability of success of the agent . The number of successes after doing n episodes is the random variable $Y_n^i = \sum_{k=0}^{n} X_k^i$ which follows a binomial law $\mathcal{B}(n, p_i)$. If n is large enough, the binomial distribution can be approximated by a normal distribution[5]. Thus we have

$$\begin{cases} p_i & \sim \mathcal{N}(p, \tau^2) \\ Y_n^i | p_i & \sim \mathcal{N}(n\,p_i, n\,p_i(1-p_i)) \end{cases} \tag{A.1}$$

where p is the mean success rate and τ the variance.

Moreover, one property of normal random variables is that if

$$\begin{cases} V & \sim \mathcal{N}(V_0, \Sigma_V) \\ U|V & \sim \mathcal{N}(U_0 + XV, \Sigma_{U|V}) \end{cases} \tag{A.2}$$

for any X,then

$$\begin{pmatrix} U \\ V \end{pmatrix} \sim \mathcal{N}(\begin{pmatrix} U_0 + XV_0 \\ V_0 \end{pmatrix}, \begin{pmatrix} X\Sigma_V X^T + \Sigma_{U|V} & X\Sigma_V \\ \Sigma_V X^T & \Sigma_V \end{pmatrix}) \tag{A.3}$$

Hence we obtain $U \sim \mathcal{N}(U_0 + XV_0, X\Sigma_V X^T + \Sigma_{U|V})$.

By identification with Equation A.1, we have

$$Y_n^i \sim \mathcal{N}(np, (n\tau)^2 + n\,p_i(1-p_i)) \tag{A.4}$$

We can rewrite it using the random variable SR_i the success rate of the the agent (trained with seed i) during the test time (over n trajectories).

$$SR_i = \frac{Y_n^i}{n} \sim \mathcal{N}(p, \tau^2 + \frac{p_i(1-p_i)}{n}) \tag{A.5}$$

Because $\frac{p_i(1-p_i)}{n} \leq \frac{0.25}{n} \xrightarrow[n\to\infty]{} 0$ and n is large, we can neglect this term with respect to τ in equation above (we verify at the end that we rightfully neglected it) and obtain:

$$SR_i \sim \mathcal{N}(p, \tau^2) \tag{A.6}$$

Using the maximum likelihood estimation for normal random variables, we get with a 99% confidence interval:

$$\hat{p} \pm\ 2.58\frac{\hat{\tau}}{\sqrt{s}} \tag{A.7}$$

$$\begin{cases} \hat{p} & = \frac{1}{s}\sum_{i=1}^{s} SR_i \\ \hat{\tau}^2 & = \frac{1}{(s-1)}\sum_{i=1}^{s}(\hat{p} - SR_i)^2 \end{cases} \tag{A.8}$$

with s the number of seeds used, $\hat{p}$ the estimator for p and $\hat{\tau}^2$ the unbiased sample variance.

[5] using the Berry-Essen theorem, the approximation is good enough if $n > 9\frac{p(1-p)}{p}$ and $n > 9\frac{p}{p(1-p)}$

A.7.2 Confidence intervals for random agents

As previously mentioned, we model the success of an agent using a Bernoulli variable $X \sim \mathcal{B}(p)$, with p the probability of success. The measured success rate after doing n episodes is the random variable $SR_n = \frac{1}{n}\sum_{k=0}^{n} X_k$ which also follows Bernoulli's law $\mathcal{B}(p)$. Following Hoffending's inequality, we have:

$$\mathbb{P}(|SR_n - p| > \varepsilon) < 2\exp\left(-2n\varepsilon^2\right) = \delta \tag{A.9}$$

with δ the error.

Thus if we use $n = 1000$ episodes to measure the success rate and we want a confidence of 99% ($\delta = 0.01$) with $\varepsilon = \sqrt{|\frac{1}{2n}\ln\frac{\delta}{2}|}$, we get $\varepsilon = 0.05$.

A.8 Word substitutions for generalization tests

For the generalization tests given in the sections 3.2.5, 3.2.6, and A.4, we use the dictionaries given below to substitute some words by others.

A.8.1 Out of vocabulary

To generate descriptions with out-of-vocabulary nouns and adjectives, we modify the prompt by substituting words as per Table A.8.

Table A.8: Out-of-vocabulary substitutions for Nouns and adjectives

Original Word	New Word
key	chair
ball	table
box	car
red	vermillion
green	jade
blue	cyan
purple	violet
yellow	golden
grey	silver

A.8.2 Invented words

Similarly to Section A.8.1, we apply the substitutions indicated in Table A.9.

Table A.9: Invented substitutions for Nouns and adjectives

Original Word	New Word
key	dax
ball	xolo
box	azfe
red	faze
green	jatu
blue	croh
purple	vurst
yellow	gakul
grey	sil

A.8.3 Synonym actions

In A.10, we choose the synonym actions to avoid as much as possible to reuse already used words in the fine-tuning (only "left" and "right" cannot be changed). To verify that Flan-T5-Large considers these words as synonyms we ask it: *"Answer the following yes/no question by reasoning step-by-step. Are <original action> and <synonym action> synonymous?"*. We retain the synonym only if it considers that this is the case.

Table A.10: Synonym actions

Original Words	Synonyms
turn left	rotate left
turn right	rotate right
go forward	move ahead
pick up	take
drop	release
toggle	switch

A.8.4 Translation to French

We give in Table A.11 the chosen translation for the french environment (the adjectives are given in the feminine form as all the objects are feminine).

Table A.11: French translation

English	**French**
turn left	tourner à gauche
turn right	tourner à droite
go forward	aller tout droit
pick up	prendre
drop	lâcher
toggle	basculer
go to a/the adj n	aller à une/la n adj
steps	pas
You see a <*object*> <*location*>	Tu vois une <objet><location>
You see a(n) *open/closed* door <*location*>	Tu vois une porte *ouverte/fermée* <location>
You carry a <*object*>	Tu portes un <objet>
key	clef
ball	balle
box	boîte
red	rouge
green	verte
blue	bleue
purple	violette
yellow	jaune
grey	grise

Appendix B

Functionally grounded representations and knowledge in LLMs

Contents

B.1 Experimental setup 162
B.1.1 Environments 162
B.1.2 Prompt variations 163
B.1.3 Question-answering on environmental knowledge 163
B.1.4 Contrastive learning regularization 165
B.2 Training costs 166
B.3 Additional results 166
B.3.1 Training curves 167
B.3.2 Statistical tests between fine-tuning approaches 167
B.3.3 Fine-tuning on each prompt separately 168
B.3.4 Larger models 169
B.3.5 Examples of trajectories 169
B.3.6 Complementary results for latent representations analysis . . 170
B.3.7 Prompt information use 172

B.1 Experimental setup

B.1.1 Environments

Our experiments use both BabyAI-Text and TWC (Murugesan et al., 2021). The latter is more complex regarding the number of objects and actions. We also argue that commonsense knowledge is critical in TWC. Indeed, agents must achieve a series of household tasks, such as "picking up an apple and placing it in an appropriate location". The agent receives a description of the room it is situated in and a list of possible actions.

TWC games are categorized into easy, medium, and hard difficulties. As difficulty increases, the number of target objects and rooms to clean up also increases, as detailed in Table B.1.

	Objects	Targets	Rooms
Easy	1	1	1
Medium	2-3	1-3	1
Hard	6-7	5-7	1-2

Table B.1: Number of objects, target objects, and rooms in TWC games per difficulty level.

To choose a difficulty level, we first conducted a σ_{zs} evaluation of Flan-T5 on TWC-Easy using different prompt formulations. The results, summarized in Table B.2, indicate that the LLM does not encounter significant difficulties in solving tasks in σ_{zs}. This is why we performed our experiments on TWC-Medium.

	P_0	P_1	P_2	P_3
78M	0.73	0.81	0.75	0.83
780M	0.83	0.9	0.86	0.88

Table B.2: Flan-T5 evaluation in zero-shot on TWC-Easy.

B.1.2 Prompt variations

In this section, we detail the prompt formulations by providing examples: Figure B.1 shows an example of the initial state s^1 in TWC-Medium, formatted with different prompt formulations, and Figure B.2 presents the same for BabyAI-Text.

B.1.3 Question-answering on environmental knowledge

In this section, we provide details on the datasets used for evaluating the environmental knowledge acquired through functional grounding, following the methodology from Xiang et al. (2023). The evaluation consists of two tasks:

(1) Task-related (TC) questions, where the model is asked to identify the relevant object needed to complete a household activity. Example: *"Question: To wash clothes, a possibly related item could be. Possible answer: ["highlighter", "crackers", "laundry detergent", "cupcake"] Answer:*

(2) Object counting (OC) questions, where the model must determine the number of objects in a specific location. Example: *"you opened a cooking pot and grabbed an apple. Next, you pulled out a dish bowl and scrubbed another apple. He found a bookshelf and put the first apple on it. Then, you opened a clothes pile and washed it before putting it on the same bookshelf. He grabbed another dish bowl and plate and put the cutlets on the bookshelf. He moved the clothes pile, grabbed the first apple, and moved it back to its original spot on the bookshelf. Finally, you put the second dish bowl on the bookshelf. How many items are there on the bookshelf?"*

P_0:
Possible actions of the agent: close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table
Goal: clean the Kitchen
Observation: You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.
Inventory: You are carrying nothing.
Next action of the agent:

P_1:
Goal: clean the Kitchen
Inventory: You are carrying nothing.
Observation: You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.
Possible actions of the agent:'close fridge', 'close kitchen cupboard', 'close oven', 'take bottle of cold water from kitchen cupboard', 'take clean mug from dining table'
Next action of the agent:

P_2:
<Begin Possible actions> close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table **<End Possible actions>**
<Begin Goal>clean the Kitchen **<End Goal>**
<Begin Observation> You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.**<End Observation>**
<Begin Inventory> You are carrying nothing. **<End Inventory>**
Next action :

P_3 :
Welcome to TextWorld! You find yourself in a messy house. Many things are not in their usual location. Let's clean up this place. After you'll have done, this little house is going to be spick and span! Look for anything that is out of place and put it away in its proper location. What you can do is to close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table. Your goal is to clean the Kitchen. You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.. Now, You are carrying nothing., and your next action is to

P_4:
{Context: 'Welcome to TextWorld! You find yourself in a messy house. Many things are not in their usual location. Let's clean up this place. After you'll have done, this little house is going to be spick and span! Look for anything that is out of place and put it away in its proper location.'
What you need to do: clean the Kitchen
What you see: 'You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.'
What are you carrying:
What you can do: "close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table"
Next action :}

Figure B.1: Example of an initial state described using different prompt formulations in TWC-Medium.

P_0:
Possible actions of the agent: turn left, turn right, go forward, pick up, drop,toggle
Goal of the agent: go to the purple box
Observation: You see a wall 2 steps forward, You see a purple box 2 steps left, You see a purple ball 1 step right and 1 step forward, You
see a grey key 2 steps right
Next action :

P_1:
Goal of the agent: go to the purple box
Possible actions of the agent: turn left, turn right, go forward, pick up, drop, toggle
Observation: You see a wall 2 steps forward, You see a purple box 2 steps left, You see a purple ball 1 step right and 1 step forward, You
see a grey key 2 steps right
Next action :

P_2:
<Begin.Possible actions>turn left, turn right, go forward, pick up, drop, toggle **<End Possible actions>**
<Begin Goal> go to a grey box**<End Goal>**
<Begin Current Observation>
Observation: You see a wall 3 steps forward, You see a wall 2 steps
left, You see a grey ball 1 step right and 1 step forward, You see a
grey box 2 steps right and 1 step forward, You see a grey box 3 steps
right and 1 step forward**<End Current Observation>**
Next action :

P_3 :
you are on a maze and you have to solve a task, what you can do is: turn left, turn right, go forward, pick up, drop, toggle your task is to go to a grey box, what you see now: You see a wall 3 steps forward, You see a wall 2 steps
left, You see a grey ball 1 step right and 1 step forward, You see a grey box 2 steps right and 1 step forward, You see a grey box 3 steps right and 1 step forward
and you next action is to

Figure B.2: Example of an initial state described using different prompt formulations in BabyAi-Text.

B.1.4 Contrastive learning regularization

In this section, we clarify the rationale behind our choice of which token the contrastive regularization should be applied to, as well as its adaptation to both encoder-decoder and decoder-only architectures. We studied two pooling methods to obtain a single representation of the prompt: mean pooling of token embeddings or using the first token embedding. We conducted an evaluation (see Table B.3) of both methods on Flan-T5 78M in TWC-Medium, and observed that applying contrastive regularization to the mean embeddings does not significantly affect performance, whereas using the first token embedding yields better regularization.

For encoder-decoder architectures, we apply regularization to the encoding of the first token in the encoders, following the work of Ni et al. (2022). In contrast, for decoder-only architectures, the causal nature of these networks implies that naively applying

	P_0	P_1	P_2	P_3
Contrastive Mean Token	0.85	0.71	0.82	0.66
Contrastive First Token	**0.97**	**0.91**	**0.95**	**0.93**

Table B.3: Comparison of the effect of contrastive regularization applied to the first token versus mean token embeddings on Flan-T5 78M in TWC-Medium.

contrastive regularization to the first token can diminish the contrastive loss, as the first token does not have access to future tokens. To address this issue, we introduce a new token, <*contrastive*>, positioned at the beginning of the prompt, which has access to bidirectional self-attention layers (i.e., no causal mask is applied. This approach enables the <contrastive> token to access the entire prompt. Figure B.3 summarizes the method.

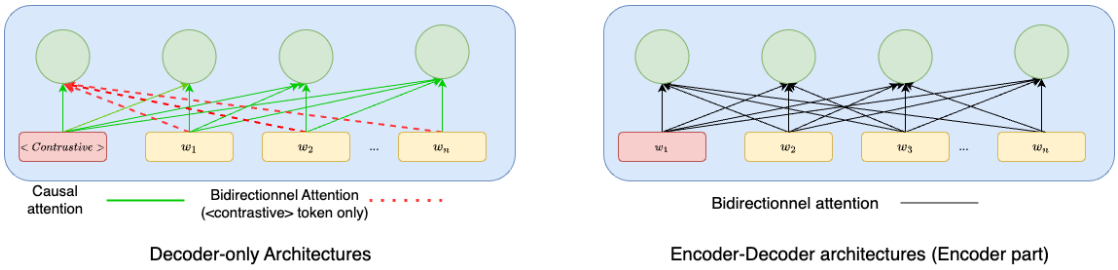


Figure B.3: Selection of a regularization token for encoder-decoder and decoder-only architectures. The regularization token is highlighted in red. In decoder-only architectures, bidirectional attention is permitted for the regularization token, enabling it to access the entire prompt and encode the semantics.

Additionally, we examined which layers are most effective for applying contrastive regularization. Empirical tests conducted on the first five layers of the LLM, presented in Figure B.4, indicate that applying the regularization on lower layers yields superior performance. Additionally, we found in our experiments that setting $\alpha = 0.5$ in Equation 3.3.3 provides a better balance between the PPO loss and the contrastive loss.

B.2 Training costs

We trained Flan-T5 78M, 80M, and 2.7B, GPT-Neo 1.3B, and Llama 7B, with *lamorel* to deploy eight instances of the LLM, each using a single NVIDIA A100 80GB. For Flan-T5 78M, we employed four instances that used NVIDIA V100 32GB GPUs. Training was conducted with five different seeds in each scenario and environment. In total, our experiments required 10800 GPU hours on A100 80GB and 6400 GPU hours on V100 32GB.

B.3 Additional results

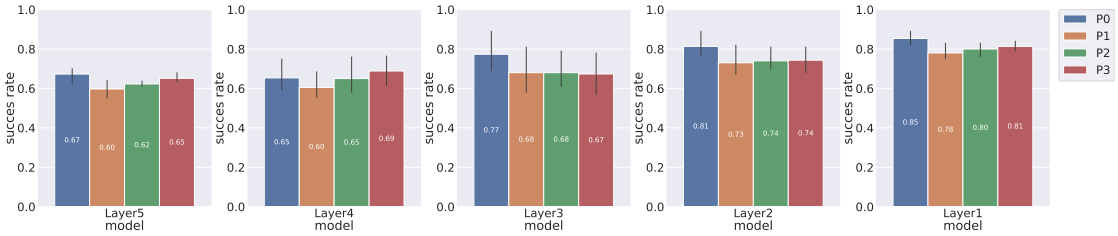


Figure B.4: Comparison of different layer choices for applying contrastive regularization on GPT-Neo 1.3B in TWC-Medium. Results reveal that lower layers provide better performance. In contrast, performance declines as the regularization moves towards higher layers.

B.3.1 Training curves

In this section, we present the SR curves over training for models trained on σ_0, $\sigma_{0:3}$, and $\sigma_0^{0:3}$ in TWC-Medium with Flan-T5 78M (see Figure B.5). All training scenarios converge, exceeding a 90% success rate. This indicates that the policy has effectively learned to solve the required tasks. The models are trained for an identical number of steps to ensure a fair performance comparison.

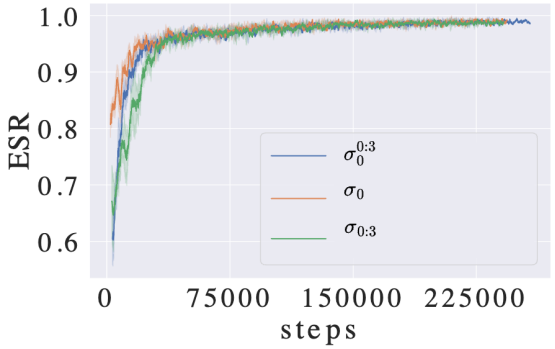


Figure B.5: **Evolution of the Success Rate (SR)** during the training of Flan-T5 78M in TWC-Medium.

B.3.2 Statistical tests between fine-tuning approaches

Table B.4 provides complementary results on the differences between our fine-tuning scenarios by displaying the mean SR and chi-squared results for the different Flan-T5 models, environments, and training scenarios.

ENV	Model	Metrics	σ_{zs}	σ_0	$\sigma_{0:3}$	$\sigma_0^{0:3}$
TWC	78 M	SR	0.28 ±0.13	0.80 ± 0.12	0.88 ± 0.04	0.94 ±0.03
		χ^2	$7x10^{-}4$ $\pm 6{,}6x10^{-4}$	$1{,}49x10^{-}4$ $\pm 2x10^{-4}$	0.99 ±0.01	0.99 ±0.01
	780 M	SR	0.38 ± 0.03	0.83 ± 0.11	0.87 ± 0.03	0.89 ± 0.05
		χ^2	0.99 ±0.01	$4{,}5x10^{-2}$ $\pm 6\times10^{-3}$	0.99 ±0.01	0.98 ±0.01
BabyAI	78 M	SR	0.25 ± 0.13	0.38 ± 0.23	0.49 ± 0.17	0.52 ± 0.21
		χ^2	$1x10^{-4}$ $\pm 10^{-4}$	$3{,}09x10^{-4}$ $\pm 2x10^{-4}$	0.531 ±0.21	$3{,}1x10^{-3}$ $\pm 4x10^{-3}$
	780 M	SR	0.41 ± 0.01	0.63 ± 0.24	0.85 ± 0.12	0.82 ± 0.12
		χ^2	0.99 ±0.01	$1{,}5x10^{-3}$ $\pm 2{,}1x10^{-4}$	0.99 ±0.01	0.97 ±0.02

Table B.4: **Mean success rate and chi-squared (χ^2) p-value for Flan-T5 models.**

B.3.3 Fine-tuning on each prompt separately

In Figure B.6, we provide complementary results to the ones presented in Section 3.3.5 by showing the performance of Flan-T5 78M and 780M when trained on each single prompt separately.

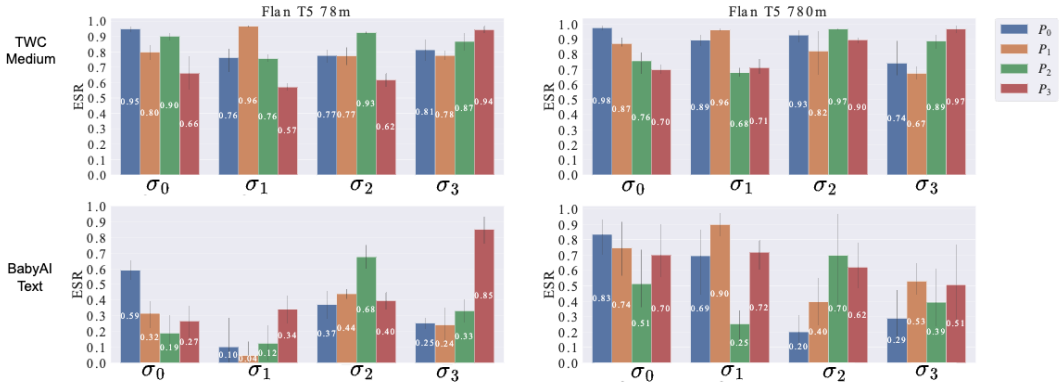


Figure B.6: **Complementary results of success rates** for both the 78M and 780M models in two environments across prompt formulations P_0 to P_3.

B.3.4 Larger models

We perform the same success rates analysis as the one presented in Section 3.3.5 on Flan-T5 2.7B and LLama 7B. Table B.5 summarizes the results, demonstrating that prompt overfitting is also present in larger models. Due to the computational cost of fine-tuning such large LLMs, this evaluation was performed exclusively on the TWC environment with σ_0 scenario.

Models	P_0	P_1	P_2	P_3
Llama 7B fine-tuned with σ_0	0.9	0.72	0.75	0.79
Flan-T5 XL 2.7B fine-tuned on σ_0	0.93	0.89	0.84	0.74

Table B.5: Success Rate (SR) for LLaMA 7B and T5-XL fine-tuned on a single prompt in the TWC environment shows prompt overfitting.

B.3.5 Examples of trajectories

Figure B.7 shows an example of four trajectories of the Flan-T5 780M model trained with σ_0 and evaluated across P_0, P_1, P_2 andP_3. The minimal number of actions is observed when formatted with the formulation used during training, i.e. P_0. By contrast, changing the prompt formulation results in irrelevant actions. For instance, when formatted with P_2, the LLM loops four times between two actions before moving to the correct actions to achieve the goal.

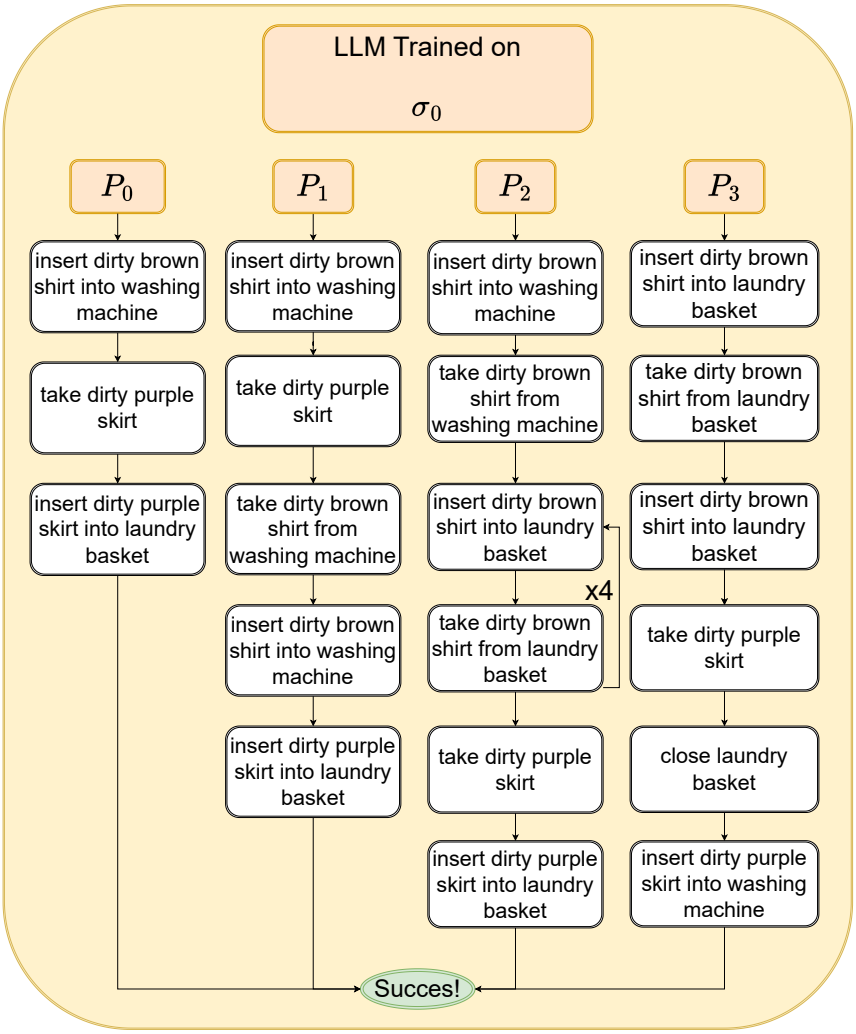


Figure B.7: **Trajectories of Flan-T5 780M** trained with σ_0 and queried using prompts P_0 to P_3. Each vertical line represents a trajectory up to achieving the goal, with the prompt formatted using a specific formulation P.

B.3.6 Complementary results for latent representations analysis

While Table 3.3 displays the mean intra and inter-prompt similarities for LLMs in σ_{zs}, σ_0, $\sigma_{0:3}$, and $\sigma_0^{0:3}$, Table B.6 provides an individual breakdown of these similarities.

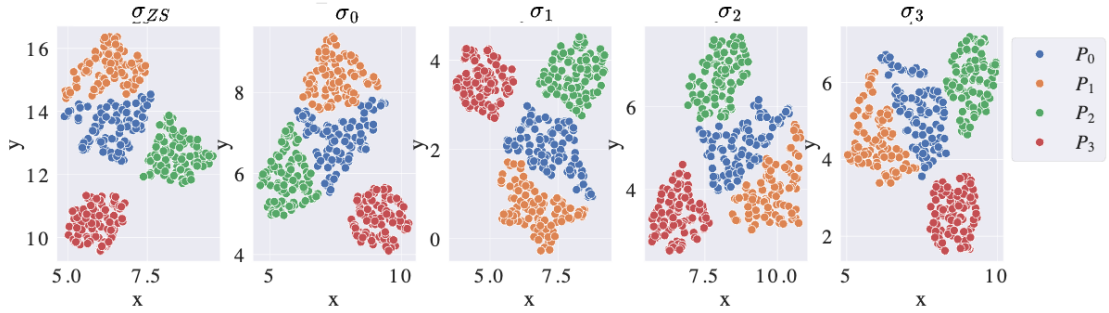


Figure B.8: **UMAP visualization of the hidden representations of Flan-T5 780M** with σ_{zs} and fine-tuning with σ_0, σ_1, σ_2 and σ_3 on TWC-Medium.

Models	similarity	σ_{zs}	σ_0	$\sigma_{0:3}$	$\sigma_0^{0:3}$
78M	$Intra(P_0)$	0.988	0.987	0.987	0.938
	$Intra(P_1)$	0.989	0.988	0.988	0.834
	$Intra(P_2)$	0.999	0.999	0.999	0.999
	$Intra(P_3)$	0.993	0.992	0.993	0.858
	$Inter(P_0, P_1)$	0.384	0.388	0.390	0.853
	$Inter(P_0, P_2)$	0.319	0.310	0.295	0.726
	$Inter(P_0, P_3)$	0.427	0.448	0.429	0.823
780M	$Intra(P_0)$	0.998	0.987	0.998	0.938
	$Intra(P_1)$	0.998	0.988	0.997	0.834
	$Intra(P_2)$	0.999	0.999	0.999	0.999
	$Intra(P_3)$	0.999	0.992	0.999	0.858
	$Inter(P_0.P_1)$	0.623	0.388	0.624	0.853
	$Inter(P_0.P_2)$	0.165	0.310	0.164	0.726
	$Inter(P_0.P_3)$	0.619	0.448	0.622	0.823

Table B.6: **Detailed inter and intra-similarity for Flan-T5 78M and 780M models** on TWC-Medium.

Besides, we further examine the UMAP visualization of models trained with σ_0, σ_1, σ_2, and σ_3 on TWC-Medium to observe potential cluster separation based on prompt formulation rather than semantic content. Results in Figure B.8 reveal distinct clusters corresponding to different prompt formulations in Flan-T5 780M, aligning with the findings presented in Table B.6. Similarly, UMAP visualization with Flan-T5 78M in BabyAI-Text (see Figure B.9) underscores the persistence of prompt overfitting across different model sizes and environments. Notably, the clustering tendency appears more pronounced in BabyAI-Text, which may elucidate the challenges encountered in implementing the contrastive solution in the 78M model variant on BabyAI-Text.

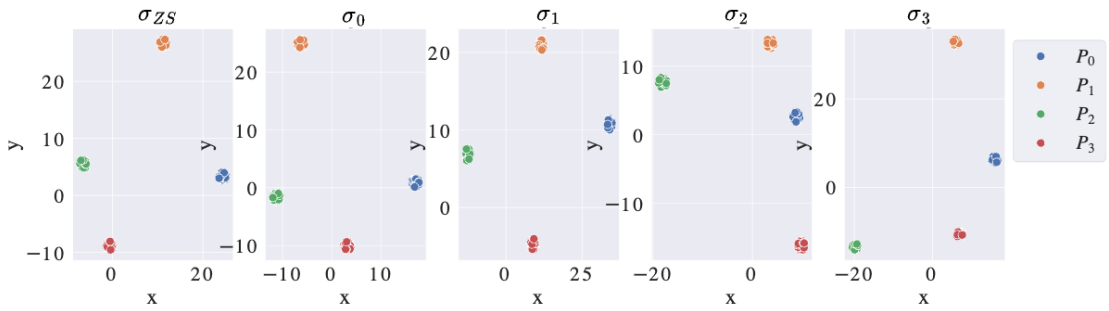


Figure B.9: **UMAP visualization of the hidden representations of Flan-T5 78M** with σ_{zs} and fine-tuning with P_0, P_1, P_2 and P_3 on BabyAI-Text.

B.3.7 Prompt information use

In Figure B.10, we provide complementary results to the ones presented in Section 3.3.6 by showing the saliency maps of Flan-T5 78M on TWC-Medium when trained on each single prompt separately. Results show a variability in the importance of prompt parts across different prompt formulations, even after fine-tuning the LLM.

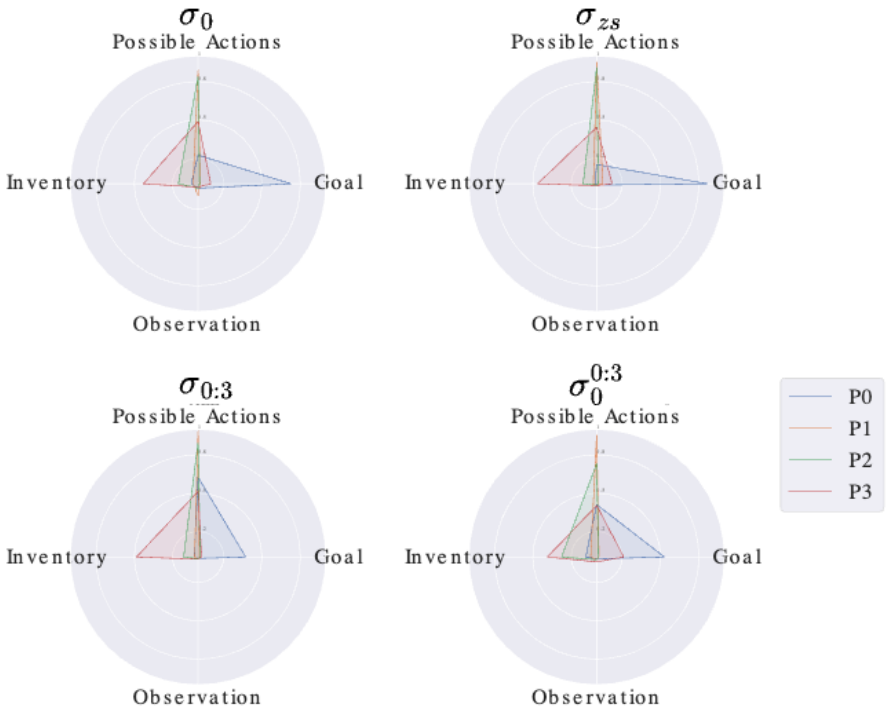


Figure B.10: **Complementary results of Saliency maps** of different parts of prompts for fine-tuned Flan-T5 78M models on TWC-Medium on σ_0 to σ_3.

Figure B.11 summarizes outcomes based on two types of prompt formulations: 1) when the formulation used during fine-tuning is equal to the one used in evaluation ($P_i = P_j$) and 2) when σ_0 differs from P_j.

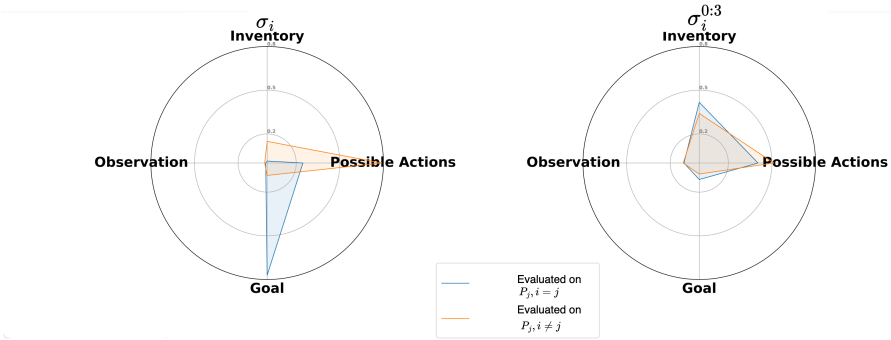


Figure B.11: **Saliency maps** for fine-tuned Flan-T5 78M models on TWC-Medium when $P_i = P_j$ and when $P_i \neq P_j$.

Notably, the contrastive regularization approach leads to more homogeneous representations, indicating consistent behavior of the LLM across various prompt formulations.

Appendix C

WorldLLM

Contents

C.1 The Playground-Text Environment . . . 174
C.2 WorldLLM details . . . 175
C.2.1 Statistician . . . 176
C.2.2 Scientist . . . 176
C.2.3 Experimenter . . . 176
C.3 Generalization Tests . . . 178
C.4 Additional Results . . . 179
C.4.1 Evolution of likelihoods . . . 179
C.4.2 Analysis of the Metropolis algorithm in the Scientist . . . 180
C.4.3 Analysis of generated hypotheses . . . 181
C.4.4 A deeper study RL-ALP . . . 182

C.1 The Playground-Text Environment

We use the Playground-Text environment from Appendix D.1, itself being a textual adaptation of the Playground environment introduced by (Colas et al., 2020). This textual environment returns, at each time step, a description of its state and possible actions. In our experiments, we used a modified version of Playground-Text where the current state still describes the whole environment's state, but the next state only describes how the performed action changed the environment (see Figure C.1).

Playground-Text features 4 different classes of objects with different varieties: *Water*, *Plants* (carrot, potato, beet, berry, and pea), *Small Herbivores* (pig, cow, and sheep), and *Big Herbivores* (elephant, giraffe, rhinoceros). As depicted in Figure 4.4, all objects except *Water* possess two states: young (seed for plants and baby for animals) and grown. These objects transition from their young state to their grown one by consuming other objects.

Three different types of actions are possible: *Go to <object>*, to move to the selected object, *Grasp*, to put the object the agent is standing on in its inventory (which has two slots) and *Release <object>/all* to release one or all objects from the inventory on the

object the agent is currently standing on. The resulting transitions can be categorized into 6 different types: *Standing*, *Holding 1*, *Holding 2*, *Grow Plant*, *Grow Small Herbivore* or *Grow Big Herbivore*. Growing transitions are triggered when the appropriate objects are released on a young one (see the technology tree in Figure 4.4). All actions involving non-existent objects in the current scene, as well as release actions that do not result in a transformation (e.g. releasing water onto water) are considered illegal and masked.

State: You see the baby cow, the water, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. You are next to the potato seed. In your inventory, there is nothing.
Action: You go to the water.
Change: You are standing on the water.

State: You see the baby cow, the water, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. You are next to the water. In your inventory, there is nothing.
Action: You pick up the object.
Change: In your inventory, there is the water.

State: You see the baby cow, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. In your inventory, there is the water.
Action: You go to the potato seed.
Change: You are standing on the potato seed.

State: You see the baby cow, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. You are next to the potato seed. In your inventory, there is the water.
Action: You give the water.
Change: The objects transform into the potato.

Figure C.1: An example of a trajectory in Playground-Text.

C.2 WorldLLM details

In this section, we provide details about our WorldLLM implementation. All our LLMs were loaded and prompted using the *transformers* library from Hugging Face. We also provide our code for reproduction on the following repository: `https://github.com/flowersteam/WorldLLM`.

C.2.1 Statistician

The LLM used for the Statistician is *Phi-3-mini-4k-instruct* quantized to 4 bits. The prompt used is shown in Figure C.2.

System prompt:
You like doing a lot of puzzles. Please answer with a brief answer and be as precise as you can.
User prompt:
You are in an environment that contains multiple objects. It can contain water, plant seeds(carrot, porato, beet, berry and pea seeds), small herbivores(pig, cow and ship) and large herbivores(elephant, giraffe, rhinoceros). You can move objects, like water, plants or herbivores and release them on another object to make them interact and transform into a new object. You know that:
<hypothesis>
Your objective is to predict the next change in the environment given the state of the environment and the action taken.
The last state was: <state>
The last action was: <action>
The change is: **Assistant prompt:**
<change>

Figure C.2: Prompt used for the Statistician. The <hypothesis> corresponds to the hypotheses to test. The <state>, <action> and <change> represent the current state, the action taken, and the resulting change, respectively, for the given transition.

C.2.2 Scientist

The LLM used for the Scientist is the same as the one used for the Statistician. We show its prompt in Figure C.3.

C.2.3 Experimenter

For the Experimenter, we provide further details for both our oracles and RL-based policies.

Oracles

We show in Figure C.4 the distribution of transitions collected by each of our four oracles.

System prompt:
You like doing a lot of puzzles. Please answer with a brief answer and be as precise as you can.
User prompt:
You are in an environment that contains multiple objects. It can contain water, plant seeds(carrot, potato, beet, berry and pea seeds), small herbivores(pig, cow and ship) and large herbivores(elephant, giraffe, rhinoceros). You can move objects, like water, plants or herbivores and release them on another object to make them interact and transform into a new object. Your previous experiences were:
<trajectories>
Can you find a set of easily understandable and concise hypotheses to predict how the environment will change based on these trajectories? You can take inspiration from the previous rules:
<previous hypothesis>
You also know that the previous set of rules failed the most on those trajectories:
<worst trajectories>
Answer with just the hypothesis.

Figure C.3: Prompt used for the Scientist. <trajectories> correspond to the trajectories collected by the Experimenter, <previous hypothesis> corresponds to the set of hypotheses that have been accepted at the last iteration and <worst trajectories> to the trajectories where the previous hypotheses obtained the worst log-likelihood.

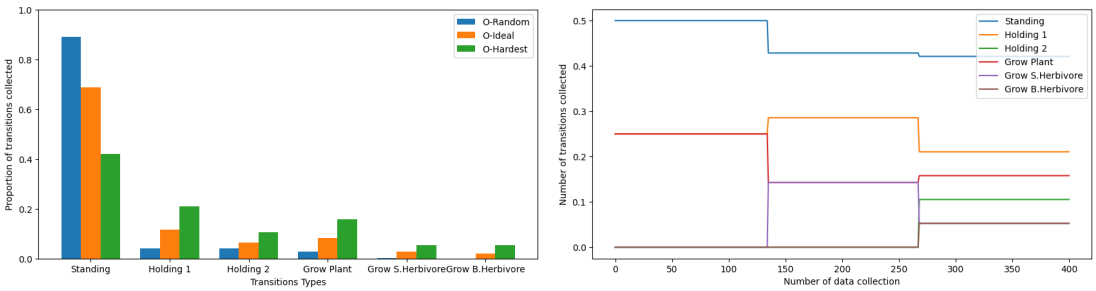


Figure C.4: Proportion of collected transition types for each oracle. We show on the left the distribution of our stationary oracles and on the right the evolving distribution of O-Curriculum.

RL agents

For our RL agents we used Masked PPO from stable-baselines3 contributions[1]. To accelerate training, we used 15 parallel environments and collected 8 trajectories per environment. We show the hyperparameters used in Table C.1.

Concerning the reward signals, we provide in Algorithm 2 the details on how RL-ALPEXP compute its ALP reward. RL-ALPEXP introduces an additional hyperparameter α which regulates the reward's smoothing of the reward. For each transition type, α corresponds to the weight of the previous rewards obtained for this transition type. Its role

[1] https://sb3-contrib.readthedocs.io/en/master/modules/ppo_mask.html

is highlighted in our study from Appendix C.4.4 but α can be interpreted as an adjustment factor determining the extent to which rewards from previous steps are retained. A value of 0 for α implies that only the reward from the most recent step is considered, which corresponds to RL-ALP, while a value of 1 indicates that the rewards are accumulated from the beginning. After a manual hyperparameter search (see Appendix C.4.4), we chose $\alpha = 0.9$ across all transition types.

hyperparameter	**value**
gamma	0.99
lr	$5e-4$
vf coef	0.5
gae λ	0.95
entropy	0.01
epochs	20
architecture	64×64

Table C.1: PPO hyperparameters used for our RL agents

Algorithm 2 RL-ALPEXP

Input: α moving average coefficient, $\mathcal{I}$ set of transition types, T total number of iterations, n_{steps} number of Metropolis steps at each iteration, f function returning the type of a transition given a transition, π the Experimenter
Initialize $(m_i)_{i\in\mathcal{I}} \leftarrow 0$ ▷ Initialize moving average for each type
for $i = 0$ to n **do**
  Collect trajectories $\mathcal{D}_t \leftarrow$env(π)
  Perform n_{steps} Metropolis steps
  for $\tau \in \mathcal{D}_t$ **do**
    Compute reward $(r_{alp})_\tau$
    $(r_{alpexp})_\tau = \alpha * m_{f(\tau)} + (1-\alpha) * (r_{alp})_\tau$ ▷ Compute ALPEXP reward
  for $i \in \mathcal{I}$ **do**
    $\tilde{m}_\beta = \frac{\sum_{\tau\in\mathcal{D}_t} \mathbf{1}_{f(\tau)=\beta}(r_{alp})_\tau}{\sum_{\tau\in\mathcal{D}_t} \mathbf{1}_{f(\tau)=\beta}}$ ▷ Compute mean over transitions
    $m_\beta = \alpha * m_\beta + (1-\alpha) * \tilde{m}_\beta$ ▷ Update moving average
  Update Experimenter π

C.3 Generalization Tests

We show in Figure C.5 how the syntax of observations was changed in our generalization tests. In particular, we modified the words associated to the transition (e.g. standing, transform) and inverted the sentence. These changes were also applied to the prompts used by the Scientist and the Statistician.

Standing: You are standing on the <*object*>. ⟶ The <*object*> is beneath you.
Holding 1: In your inventory, there is the <*object*>. ⟶ The <*object*> is in your grasp.
Holding 2: In your inventory, there are the <*object*> and the <*object*>. ⟶ The <*object*> and the <*object*> are in your grasp.
Grow Plant: The objects transform into the <*object*>. ⟶ The <*object*> results from combining the objects.
Grow Small Herbivore: The objects transform into the <*object*>. ⟶ The <*object*> results from combining the objects.
Grow Big Herbivore: The objects transform into the <*object*>. ⟶ The <*object*> results from combining the objects.

Figure C.5: Changes in the observations for the generalization environment.

C.4 Additional Results

In this section, we first provide the complete evolution of log-likehoods on our test set from Section 4.2.2. Then, we provide additional studies and experiments to improve one's comprehension of WorldLLM's behavior. These investigations notably explore the reasons behind the failure of RL-ALP and its differences with RL-ALPEXP.

C.4.1 Evolution of likelihoods

We report in Figure C.6 the evolution of log-likelihoods on our test set for all the evaluated methods.

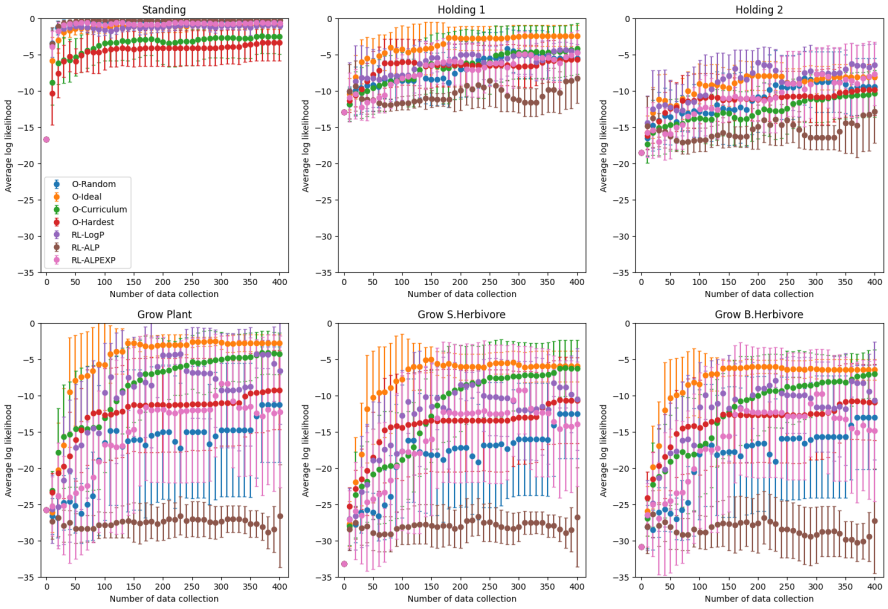


Figure C.6: Evolution of log-likelihoods on the test set for all methods.

C.4.2 Analysis of the Metropolis algorithm in the Scientist

Hypotheses Retention Rate

First, we analyze how the Metropolis algorithm performs when using an LLM as both the proposal and target distribution. In particular, we study the evolution of accepted hypotheses throughout WorldLLM's iterations when using different Experimenters (Figure C.7).

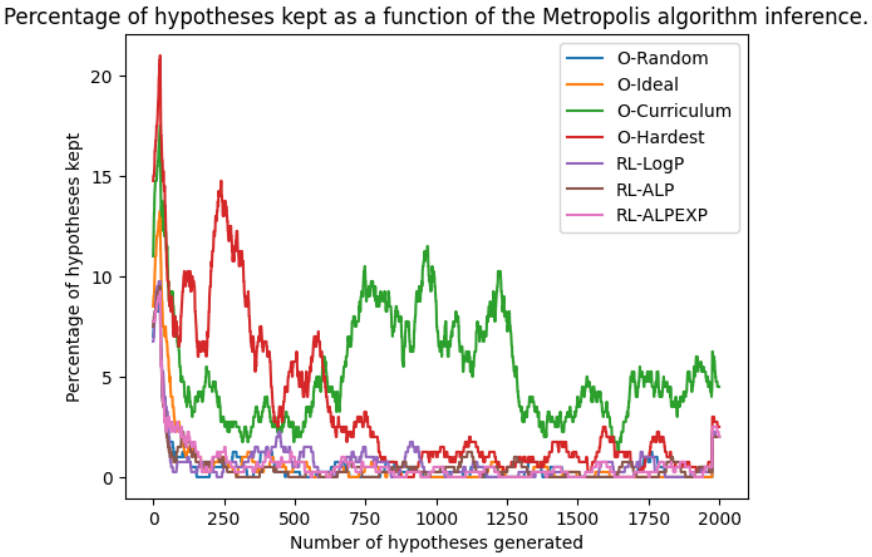


Figure C.7: Evolution of the hypotheses retention rate for the different Experimenters.

Our results show a high percentage of accepted hypotheses (around 20%) at the beginning of the experiment. However, as iterations progress, it becomes increasingly challenging for the Scientist to generate more effective hypotheses, leading to a decline in the retention rate (it reaches 2%). As highlighted in our manuscript, a developmental strategy is key in WorldLLM. This is here illustrated by O-Curriculum's ability to maintain a higher retention rate than all other oracles. Additionally, preliminary experiments performed with Phi4 suggested that employing a larger LLM for the Scientist improves hypotheses generation and leads to higher retention rates for the same Statistician model.

Embeddings of the hypotheses

In order to compare the hypotheses generated by our Scientist, we used a sentence transformer, **all-mpnet-base-v2**[2], to project hypotheses in a latent space. We then used T-SNE(Maaten & Hinton, 2008) to obtain a 2D representation of the embeddings. As the number of generated hypotheses is high, we plotted only 1/10 of them for each Experimenter and seed. We show these embeddings in Figure C.8.

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

The results show that O-Curriculum and O-Hardest mostly produce similar hypotheses. This indicates a Scientist's tendency to refine previous hypotheses rather than generating brand new ones. This corroborates our results on retention rate from previous section. We also compute an inertia score across all seeds for each Experimenter:

$$\forall \beta \in \mathcal{M}, \quad I = \sum_{i=1}^{n} (x_i - C_\beta)^2 \qquad \text{with} \quad C_\beta = \sum_{i=1}^{n} \frac{x_i}{n} \tag{C.1}$$

with $\mathcal{M}$ the set of Experimenters, x_i the 2D projection from T-SNE for each set of hypotheses and n the total number of hypotheses.

Results indicate that the clusters produced by O-Curriculum and O-Hardest are each produced by a different seed. This highlights the current limitation of our Bayesian inference approach where a single particle is used and never reset. As a result, our Scientist may easily get stuck in local optima.

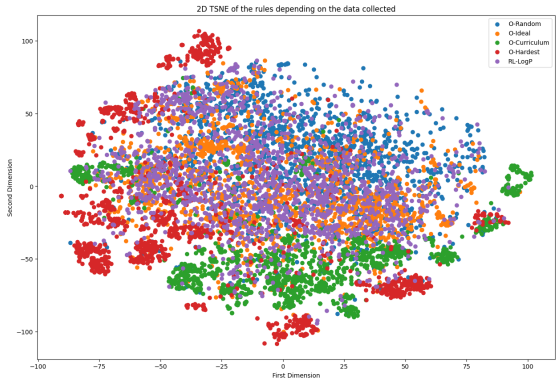


Figure C.8: T-SNE embeddings obtained using the *all-mpnet-base-v2* sentence transformer on the hypotheses generated by different Experimenters.

Methods	Inertia Score
O-Random	4.12e6
O-Ideal	4.38e6
O-Curriculum	2.77e6
O-Hardest	2.88e6
RL-LogP	4.52e6
RL-ALP	3.62e6
RL-ALPEXP	4.92e6

Table C.2: Mean of the inertia score on the T-SNE embeddings across the different seeds for each Experimenter.

C.4.3 Analysis of generated hypotheses

In this section, we propose a qualitative study of hypotheses produced by WorldLLM when using O-Curriculum, O-Ideal and RL-LogP as Experimenters (see Figure C.9).

One can observe that each set of hypotheses covers multiple transition types. This can be explicit, as seen in the hypotheses from O-Ideal, or more abstract, as the ones from O-Curriculum, which replace specific animals and objects with abstract representations. A closer examination of O-Curriculum's hypotheses shows that new entities that do not exist in Playground-Text such as "bear", "mouse" or "wolf" are introduced. This aligns with our argument that natural language hypotheses could help the world model generalize. However, our poor generalization results can be explained by the fact that most hypotheses contain examples of transitions from the environment. Such examples tremendously help our Statistician predict outcomes but also significantly impair its generalization capabilities.

O-Curriculum:
1. If the action is 'You go to' and the object is a seed or gentle creature (cow, elephant, sheep, giraffe, bear, mouse, wolf), predict "You are standing on the [object]."
2. If you are standing on an animal's seed and the action is 'You give the water', predict "The objects transform into the [animal]."
3. If the object is a seed or gentle creature (cow, elephant, sheep, giraffe, bear, mouse, or wolf) and the action is 'You give the water', predict "The objects transform into the [animal]."
4. If you are standing on the water and the action is 'You pick up', predict "In your inventory, there is the water."
5. If you are standing on the water and the action is 'You pick up the object', predict "In your inventory'
O-Ideal:
1. The action was: You go to the water, and the change was: You are standing on the water.
2. The action was: You pick up the object, and the change was: In your inventory, there is the water.
3. The action was: You go to the beet seed, and the change was: You are standing on the beet seed.
4. The action was: You give the water, and the change was: The objects transform into the beet.
5. The action was: You pick up the object, and the change was: In your inventory, there is the beet.
6. The action was: You go to the baby cow, and the change was: You are standing on the baby cow.
7. The action was: You give the beet, and the change was: The objects transform into the cow.
8. The action was: You go to the pe'
RL-LogP:
1. Go to the water.
Change: You are standing on the water.
2. Pick up the object.
Change: In your inventory, there is the water.
3. Go to the water.
Change: You are standing on the water.
4. Pick up the object.
Change: In your inventory, there are the water and the water.
5. Go to the berry seed.
Change: You are standing on the berry seed.
6. Give the water.
Change: The objects transform into the berry.
7. Go to the water.
Change: You are standing on the water.
8. Pick up the object.
Change: In your inventory, there are the water and the water.
9. Go to the pea seed.
Change: You are standing on the pea seed.
10. Give the water.
Change: The objects transform into

Figure C.9: Examples of accepted hypotheses.

C.4.4 A deeper study RL-ALP

In this section, we provide a deeper analysis of RL-ALP's failure. While expected to be more robust than RL-LogP (e.g. to the "noisy TV" problem), our results showed RL-ALP was unable to collect other transitions than *Standing*, leading to very poor hypotheses. We begin this section by studying how directly using the per-transition ALP as reward for an RL policy is challenging. Then, show how RL-ALPEXP differs and helps stabilize training.

Failure of RL-ALP as the reward

To get a better grasp of the RL policy's behavior when using RL-ALP, we plot the evolution of rewards obtained by the policy in Figure C.10. We observe that most rewards equal 0. Higher rewards are only obtained in the first data collections when *Standing* transitions are not well predicted by the Statistician. After gathering such transitions, the policy eventually converges to a random one. This absence of rewards can be explained by our reward definition from Eq. 4.2.

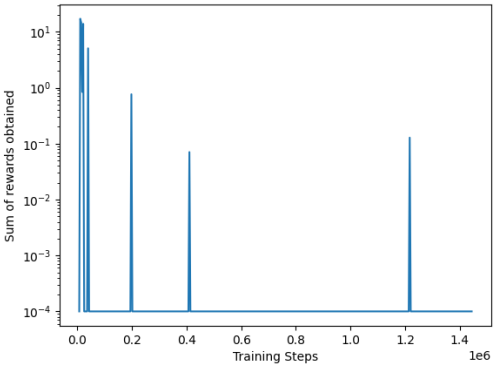


Figure C.10: The evolution of rewards obtained by the RL agent when using RL-ALP for a single seed. We plot the sum of rewards obtained for each PPO update.

A positive reward is obtained only if a set of hypotheses are accepted between the previous and the current iteration. However, as discussed in Appendix C.4.2, the number of accepted hypotheses tends to quickly drop and eventually reach a very low percentage. As a result, very few improvements (or degradation) are made to the Statistician's log-likelihoods, leading to zero rewards.

Ablation study on RL-ALPEXP

In Appendix C.2.3, we detailed RL-ALPEXP, which includes a new hyperparameter α. This hyperparameter governs the weighting of smoothing applied to the reward computed on various transition types over time. In this section, we explore the impact of this hyperparameter.

To assess how α influences RL-ALPEXP, we conducted qualitative experiments with α values set to $\{0.6, 0.75, 0.9, 0.95\}$. To gain insights into the underlying behavior, we analyze the transitions collected by the policy. The results can be found in Figure C.11.

From these results, we observe two behaviors depending on the values of α. On one hand, when α is smaller than 0.9, the RL policy demonstrates a periodic behavior. This phenomenon becomes more pronounced as α decreases, leading to instability that hinders WorldLLM's ability to improve consistently.

It seems that the reward diminishes to zero before the Scientist and Statistician can refine the existing hypothesis. This decline ultimately leads to a collapse in the RL agent's

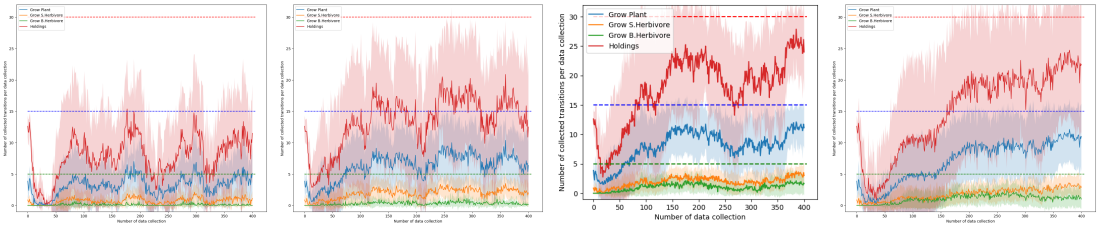


Figure C.11: Transitions collected by the Experimenter when using RL-ALPEXP for $\alpha = \{0.6, 0.75, 0.9, 0.95\}$ respectively.

policy. As a result, the distribution of the collected data begins to resemble that of a random agent, as in the first iterations of WorldLLM. In response, the Scientist generates hypotheses that cause the framework to revert to an earlier stage.

On the other hand, as α gets closer to 1, the reward takes increasingly longer to fade. This can be problematic as the reward function used is an estimation of the learning progress. For example, the high rewards collected at the beginning of training are no longer relevant after multiple iterations.

This hyperparameter search explicitly shows that RL-ALPEXP has not fully resolved the sparse reward problem inherent to RL-ALP. If α is too small, the agent's policy collapses to random. If it is too close to 1, the old rewards obtained require too much time to fade. One potential solution could be to add a small fraction of RL-LogP's rewards. This addition would prevent the agent's policy from collapsing but would require a new hyperparameter to balance the reward from RL-ALP and RL-LogP.

Appendix D

SAC-GLAM

Contents

D.1 Playground-Text 185
D.2 Implementation details 186
D.3 Batch sampling with HER 187
D.4 Hyperparameters 188

D.1 Playground-Text

The Playground-Text environment is a text-based environment where goals, observations, and actions are all represented in natural language. The environment consists of a scene with N=6 objects. The agent can move toward these objects using the action "*Go to {object}*", and can interact with them using "*Grasp*" to pick up objects and "*Release*" to put them down. There are four types of objects: furniture, supplies, plants, and animals. The "supplies" category contains two items: food and water. When the agent brings water to a plant, the latter grows; animals can grow with either water or food. Objects' colors are irrelevant for solving goals and serve only as distractions for the agent. Observations from the environment fully describe its state, as shown in Figure D.1.

Goal: Grow green mouse then grow blue rose
You see: green water, green mouse, blue rose, red rose, blue lion, red food
You are on: nothing
You are holding: nothing
You have grown: nothing
Action:

Figure D.1: **An observation in the Playground-Text environment.** All necessary information is provided in the observation, making the environment fully observable.

There are two types of goals in the environment: "*Grasp {object}*" and "*Grow {object}*." Additionally, we introduce sequential goals such as "*Grow {object A} then grasp {object B}*" or "*Grow {object A} then grow {object B}*." Rewards are set to 0 at each step, with a reward of 1 when achieving the goal. The agent is trained on goals sampled uniformly from all available goals. However, the distribution of goal types is not uniform, with the majority being sequential goals (Figure D.2).

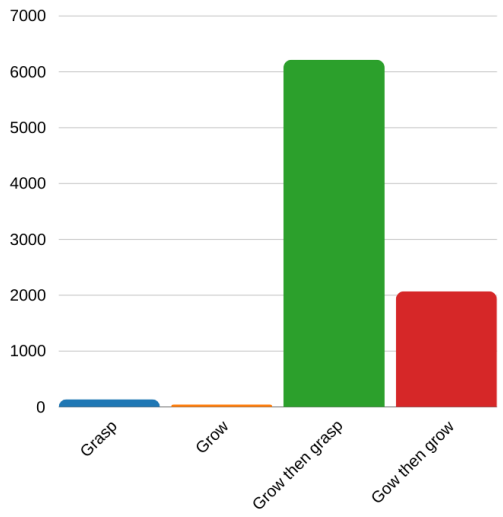


Figure D.2: **Proportion of each goal type.** The distribution of goals is highly imbalanced, with sequential goals making up 98% of the total.

D.2 Implementation details

In our approach, the actor is modeled using a pre-trained encoder-decoder LLM, with the critic positioned as an MLP attached to the final decoder block, similar to the GLAM architecture. To optimize GPU VRAM usage during training, we employed LoRA (Hu et al., 2022) and 4-bit quantization.

We implemented SAC with automatic tuning of the entropy coefficient (using a learning rate of 2e−3). For SAC-GLAM alone, we initialized it at 0.005 and set the target entropy to be 0. For SAC-GLAM with HER, we initialized it at 0.05 with a target entropy of 0.01. Both the actor and critic were trained using the Adam optimizer with a learning rate of 1e−4. The critic MLP has two hidden layers, each with 1024 units and ReLU activation functions. Detailed hyperparameters are provided in Appendix D.4.

We used a vectorized version of Playground-Text with 32 instances of the environment running (synchronously) in parallel. We leveraged Lamorel[1], as in GLAM, and deployed four instances of the LLM in parallel (distributing both the forward passes to compute the actions' probability and the training in a Data Parallelism setting). When using

[1]https://github.com/flowersteam/lamorel

Flan-T5 250M, each LLM instance is deployed on one Nvidia V100 32GB GPUs, requiring a total of 4 Nvidia V100 32GB GPUs to run an experiment (1 GPU $\times$ 4 LLM instances). In total, running all experiments and ablations required 80 GPU hours per seed on the Nvidia V100 32GB.

D.3 Batch sampling with HER

We studied the impact of how relabeled trajectories are integrated in the replay buffer (Figure D.3). We implemented HER with the *future* strategy proposed in Andrychowicz et al. (2017) (i.e., a trajectory is relabeled with every achieved goal and all these relabeled trajectories are added to the replay buffer) and compared three settings:

- The default one, where transitions are randomly sampled from the replay buffer to train our policy and critic (which can have the disadvantage of leading to batches predominantly filled with relabeled trajectories, as shown in Figure D.4)
- The "prior" scenario, where, instead of uniformly sampling from the replay buffer, batches were sampled with transitions that reflected the environment's goal distribution (see Appendix D.2)
- The "ratio" scenario (used in all our experiments as described in in Section 5.1.1), where we balanced the batches by sampling 50% of relabeled transitions and 50% of transitions with their original goal (a similar approach was used in Colas et al. (2020), where they also employed HER and sampled an equal proportion of positive and negative transitions)

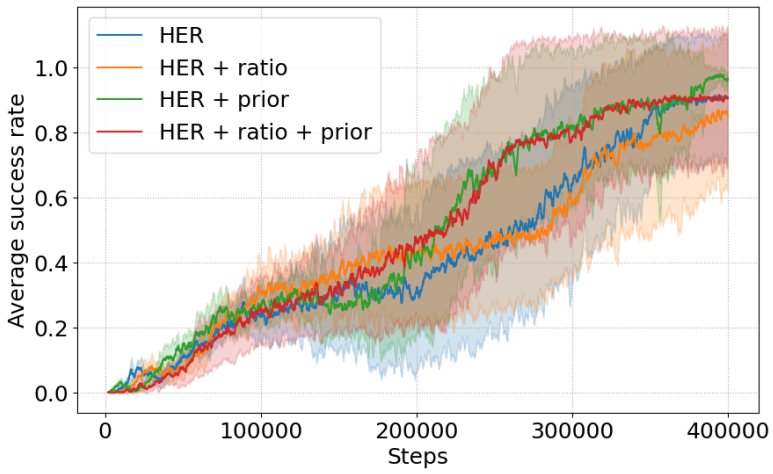


Figure D.3: **Comparison of different sampling strategies.** "prior" refers to the sampling strategy used to address issue (1), while "ratio" refers to the strategy used to address issue (2).

We only observed a minor impact on sample efficiency and opted for the "HER + ratio" strategy in our experiments, as the "prior" method makes the strong assumption of knowing the environmental goal distribution.

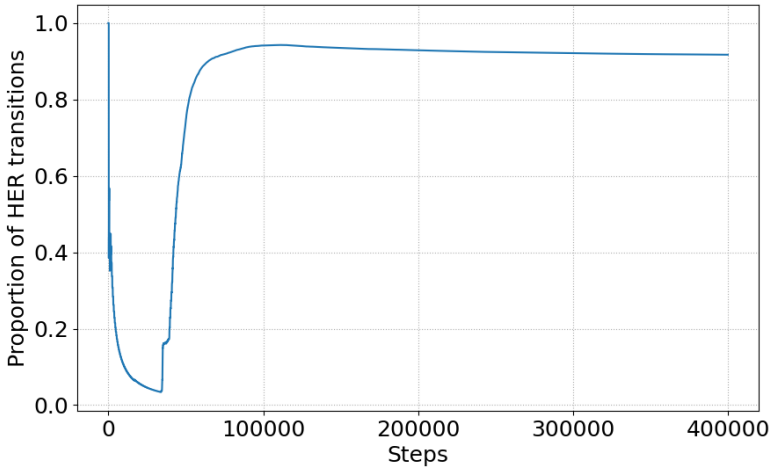


Figure D.4: **Proportion of HER transitions in the replay buffer when uniformly sampling.** The plot shows the proportion of transitions relabeled by HER compared to those directly collected from the environment.

D.4 Hyperparameters

Tables D.1, D.2 and D.3 summarize all the hyperparameters used in the experiments whose results are presented in Figure 5.2.

Table D.1: PPO-GLAM hyperparameters

Variable	**Value**
Number of transitions collected between two updates	2048
Number of epochs per update	16
Batch size	256
Entropy loss coefficient	0.01
Value function loss coefficient	0.5
Discount factor	0.99
Optimizer	Adam
Learning rate	1×10^{-4}
λ factor of the Generalized Advantage Estimator	0.99
Clipping parameter ϵ	0.2
Maximum gradient norm	0.5

Table D.2: SAC-GLAM hyperparameters

Variable	**Value**
Update frequency	32
Number of updates	2
Batch size	256
Discount factor	0.99
Optimizer	Adam
Critic learning rate	1×10^{-4}
Actor learning rate	1×10^{-4}
Entropy coefficient	auto
Entropy coefficient initialization	0.005
Target entropy	0
Entropy coefficient learning rate	2×10^{-3}
n-step	3
Replay buffer capacity	100000

Table D.3: SAC-GLAM + HER hyperparameters

Variable	**Value**
Update frequency	32
Number of updates	2
Batch size	256
Discount factor	0.99
Optimizer	Adam
Critic learning rate	1×10^{-4}
Actor learning rate	1×10^{-4}
Entropy coefficient	auto
Entropy coefficient initialization	0.05
Target entropy	0.01
Entropy coefficient learning rate	2×10^{-3}
n-step	3
Replay buffer capacity	100000
Hindsight proportion per batch	0.5

Appendix E

MAGELLAN

Contents

E.1 Little-Zoo environment . . . 190
E.1.1 Environment mechanics . . . 190
E.1.2 Goal space generation . . . 192
E.1.3 Example of impossible goals . . . 192
E.1.4 Goal repartition . . . 193
E.2 Comparison of LP methods . . . 194
E.3 Implementation details . . . 195
E.3.1 LLM-based RL agent . . . 195
E.3.2 MAGELLAN . . . 196
E.3.3 Baselines . . . 197
E.3.4 Compute budget . . . 198
E.4 Additional results . . . 199
E.4.1 Ablations on the MAGELLAN architecture . . . 199
E.4.2 Q1. Competence estimation properties . . . 202
E.4.3 Q2. Training an LLM-based RL agent with MAGELLAN . . . 206
E.4.4 Q3. MAGELLAN's generalization abilities . . . 208
E.4.5 Q4. Leveraging the generalization abilities when facing new goals . . . 211

E.1 Little-Zoo environment

E.1.1 Environment mechanics

There are 5 categories of elements the agent can interact with: the objects (i.e. "bookshelf"), the water, the plants (i.e. tomato) which can bee `seed` and `grown`, the herbivores (i.e cow) which can be `baby` or `grown`, and the carnivores (i.e. lion) which can be `baby` or `grown`.

To interact with any element of the environment, multiple actions are accessible. First, the agent can use `"Go to {element}"` to move to any element. You can only move

to an element (not a random point in the plane) and after a `"Go to"` action you are placed on the element. When performing the `"Grasp"` action, the element the agent is standing on is added to its inventory (up to 2 items). Some elements can interact with each other. To make two elements interact, the agent must release one element in its inventory by using the `"Release {element}"` action while standing on the other element. Interactions between three elements are also possible: the agent has to hold two elements in its inventory and perform `"Release all"` while standing on the third element. The "Release" action is only accessible if an interaction can be made between the element to release and the one the agent is standing on. Therefore, the agent must carefully manage its inventory to avoid filling it with useless elements without being able to release them. At each turn you get a full description of the environment with `"You see {element_1}, {element_2}, {element_3}, {element_4}"`.

To grow a plant, an herbivore or a carnivore, the agent needs to follow a certain sequence of actions:

- To `"Grow a plant"`, the agent has to `"Release water"` while `"Standing"` on the plant seed.
- To `"Grow an herbivore"`, the agent has to `"Release a grown plant"` while standing on a baby herbivore.
- To `"Grow a carnivore"`, the agent has to `"Release a grown herbivore"` while standing on a baby carnivore.

Table E.1 summarizes the optimal action strategy for each category. In order to maintain a coherent level of difficulty between the different categories, we fix the maximal number of actions allowed to the agent to solve a goal as 150% of the minimal number of actions required to solve the goal. The required exploration to discover strategies has, therefore, the same level of difficulty for all goals.

Table E.1: The optimal trajectories per category.

Categories	Optimal action strategy	Minimal number of actions	Maximal number of actions
Grasp X	Go to X, Grasp	2	3
Grow plant	Go to water, Grasp, Go to plant, Release water	4	6
Grow herbivore	Go to water, Grasp, Go to plant, Release water, Grasp, Go to herbivore release plant	7	11
Grow carnivore	Go to water, Grasp, Go to plant, Release water, Grasp, Go to herbivore, release plant, Grasp, Go to carnivore, release	10	15

E.1.2 Goal space generation

To define our goal space, we start from the sets of objects $\mathcal{O}$, plants $\mathcal{P}$, herbivores $\mathcal{H}$, carnivores $\mathcal{C}$ and the element water (each set being of size 6). One goal g in the goal space is defined using 5 elements $(E_0, E_1, E_2, E_3, E_4) \in \{\mathcal{O}, \mathcal{P}, \mathcal{H}, \mathcal{C}, \text{water}\}$ and by using the objective function $\mathcal{F}_g$ from the set $\{\texttt{Grasp\{X\}}, \texttt{Grow\{X\}}\}$ on E_0. Thus, we get the goal $g = (\mathcal{F}_g(E_0), E_1, E_2, E_3, E_4)$. As each set contains 6 elements, we get a total of $19,531,250$ goals. The vast majority of these goals are impossible for various reasons (see Appendix E.1.3).

In our experiments, we subsample two goal spaces: a train and held-out test space. To generate them, we sample the total number of impossible goals n_{impo} we want to have, then sample among the possible goals: $\frac{n_{impo}}{5}$ goals where $E_0 \in \mathcal{O}$, $\frac{n_{impo}}{5^2}$ goals where $E_0 \in \mathcal{P}$, $\frac{n_{impo}}{5^3}$ goals where $E_0 \in \mathcal{H}$, and $\frac{n_{impo}}{5^4}$ goals where $E_0 \in \mathcal{C}$. Appendix E.1.4 gives more details on the goal repartition.

E.1.3 Example of impossible goals

In our environment, a goal is defined as a combination of elements present in the environment and an instruction, such as "grasp the rabbit". Some combinations are incompatible such as:
`Grasp rabbit`
`You see: bookshelf, baby lion, desk, baby cow`,
where there is no rabbit. The generation of our environment leads to having 80% of impossible goals in the goal space, which is similar to any complex environment Zhang et al. (2024b); Matthews et al. (2025). However, conversely to Zhang et al. (2024b), identifying impossible goals and avoiding to select them is not trivial, as there are multiple reasons why a goal is impossible.

The easiest reason for impossibility is the absence of the element the agent has to act on:

- Absence of the element to grasp
  `Goal: Grasp bed`
  `You see: bookshelf, baby lion, desk, baby cow.`
- Absence of the element to grow
  `Goal: Grow deer`
  `You see: baby giraffe, bookshelf, water, tomato seed.`

Another reason is that the element cannot follow the dynamic required in the objective:

- Growing an element that is neither an animal nor a plant
  `Goal: Grow desk`
  `You see: bed, baby bobcat, baby elephant, door.`

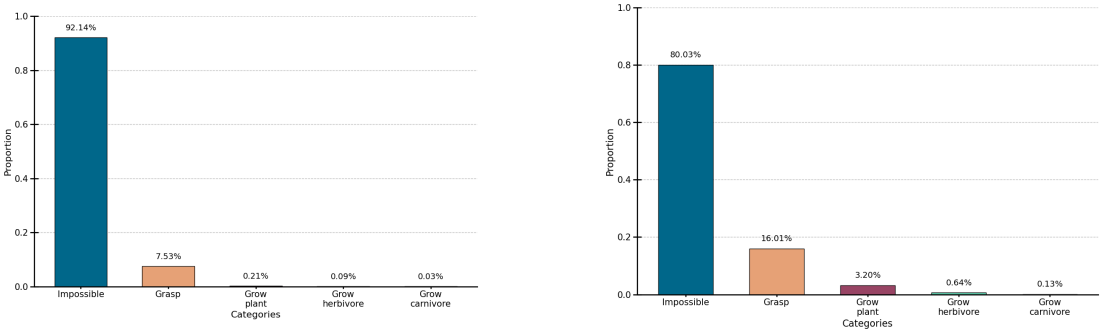
A more difficult reason that requires a better understanding of the underlying mechanisms of the environment is the lack of at least one element necessary to reach the objective grow:

- The lack of water:
  `Goal: Grow cucumber`
  `You see: cucumber seed, carrot seed, baby coyote, baby wolf,`
  `Goal: Grow coyote`
  `You see: cucumber seed, berry seed, baby coyote, baby cow.`

- The lack of a plant preventing the agent to grow an herbivore resulting in the impossibility to grow a carnivore:
  `Goal: Grow coyote`
  `You see: water, baby elephant, baby coyote, baby cow.`

Consequently, to efficiently build a curriculum, being able to understand why a goals is impossible and quickly generalize to infer the other impossible goals is crucial.

E.1.4 Goal repartition



(a) Goal distribution per category on the full goal space.

(b) Goal distribution per category in our experiments.

Figure E.1: We first generate the full goal space whom the distribution is given Figure E.1a then for computational reasons we sample a smaller goal space with the following distribution Figure E.1b.

At the end of the goal space generation, we have 19531250 goals distributed as presented in Figure E.1a. However, using all the goals was computationally impossible considering all the comparisons we made. Moreover, the repartition with 92% of impossible goals and 7.53% of "Grasp" goals made the epsilon-greedy exploration of our goal selector too slow to discover LP niches in an acceptable compute budget. Subsequently, we sample a training goal space and a testing goal space following the distribution given in Figure E.1b. Both goal spaces are composed at 80% of impossible goals, the remaining goal categories becoming less and less represented as the difficulty increases. This repartition synthetically simulates how natural language goals mostly lead to sequences of words without any meaning. This structure makes the strategy of uniformly sampling goals inefficient and underlines the necessity of a curriculum to be able to reach the hardest goals in the allocated training budget (500k episodes in our experiments).

E.2 Comparison of LP methods

We compare prior work computing LP for automatic curriculum learning under the dimensions from Section E.4.2. We show the comparison in Table E.2 evaluating methods:

- **Efficiency**: Computational cost of additional evaluation episodes not used to train the policy.
- **Competence transfer tracking**: How well does the method track all the possible competence transfer.
- **No expert knowledge required**: if they require any external expert knowledge such as pre-defined goal groupings.

We consider a method's efficiency as "high" if it does not require any additional evaluation (i.e. it only uses the performance observed on goals sampled), and as "low" otherwise. We evaluate the competence transfer tracking using the following criteria:

- absence of +: the estimated competence is updated only on sampled goals.
- +: the estimated competence is updated on a predefined goal subset the sampled goal belongs to.
- ++: the estimated competence is updated on a dynamically learned goal subset the sampled goal belongs to.
- +++: the estimated competence is updated on all goals.

Methods	Efficiency	Competence transfer tracking	No expert knowledge required
Oudeyer & Kaplan (2007)	high	++	✓
Baranes & Oudeyer (2009)	high	++	✓
Baranes & Oudeyer (2013)	high	++	✓
Moulin-Frier & Oudeyer (2013)	high	++	✓
Moulin-Frier et al. (2014)	high	++	✓
Portelas et al. (2020a)	high	++	✓
Forestier & Oudeyer (2016)	high	++	✓
Kovač et al. (2023)	high	++	✓
Stout & Barto (2010)	high	+	×
Lopes & Oudeyer (2012)	high	+	×
Matiisen et al. (2017)	high	+	×
Fournier et al. (2018)	high	+	×
Colas et al. (2019)	high	+	×
Blaes et al. (2019)	high	+	×
Akakzia et al. (2021)	high	+	×
Kumar et al. (2024)	high	+	×
Kanitscheider et al. (2021)	low	+++	✓
Zhang et al. (2024b)	low	+++	✓

Table E.2: Comparison of prior work w.r.t. their LP estimation approach. We use the following dimensions: computational efficiency, dynamical competence transfer assumptions and no required expert knowledge.

E.3 Implementation details

To facilitate reproduction and future work, we provide our code at https://github.com/flowersteam/MAGELLAN.

E.3.1 LLM-based RL agent

We use SAC-GLAM with the hyperparameters listed in Table E.3. Following Section 5.1, the Q-value head is a two-layer MLP with 1024 ReLU-activated units, applied to the last hidden state of the decoder. The policy and the critic share the same LoRA adapters. Additionally, we apply a warm-up phase of 10 updates, during which only the Q-function is trained.

Table E.3: SAC hyperparameters

Variable	Value
Update frequency	64
Number of updates	1
Batch size	256
Discount factor	0.99
Optimizer	Adam
Critic learning rate	1×10^{-4}
Actor learning rate	1×10^{-4}
Entropy coefficient	auto
Entropy coefficient initialization	0.05
Target entropy	-0.125
Entropy coefficient learning rate	1×10^{-3}
n-step	3
Replay buffer capacity	500000

The prompt used as the LLM agent's observation is shown in Prompt E.3.1.

Prompt C.1: RL Agent observation

Goal: Grow lion
You see: water, carrot seed, baby lion, baby cow
You are standing on: nothing
Inventory (0/2): empty
Action:

E.3.2 MAGELLAN

Our competence estimator uses a single-layer MLP with 128 units and Tanh activations, applied to the last hidden state of the LLM. It is updated every 32 policy updates, with batch sampling that prioritizes recent data. The sampling distribution is determined by $\frac{i}{\sum_{j=1}^{M} j}$, where i represents the position of a data point in the buffer $\mathcal{D}$ (with 1 being the oldest and M the buffer size). This strategy ensures that the competence predictor remains responsive while preserving batch diversity, incorporating data from different versions of the learning agent to help mitigate noise introduced by the fluctuations in the RL agent's learning process.

To compute the LP, we store the weights of our competence estimator in the buffer $\mathcal{B}$. This is done by saving the LoRA adapters and the MLP weights whenever the competence estimator is updated. The window for LP computation is defined by $|\mathcal{B}| \times$ *update frequency*. The LP is calculated by comparing the competence estimation obtained with the oldest and most recent weights in the buffer.

The MAGELLAN hyperparameters are provided in Table E.4.

Table E.4: MAGELLAN hyperparameters

Variable	Value
ϵ start	1
ϵ end	0.2
ϵ decay period	320
$\mathcal{B}$ size	100
$\mathcal{D}$ size	5000
Batch size	256
Optimizer	Adam
Learning rate	1×10^{-4}
Update frequency	32

The prompt given to the competence estimator is the same as the prompt given to the RL agent Prompt E.3.2.

Prompt C.2: MAGELLAN prompt

Goal: Grow lion
You see: water, carrot seed, baby lion, baby cow
You are standing on: nothing
Inventory (0/2): empty
Action:

E.3.3 Baselines

We compared our method against four baselines: Online-ALP, Eval-ALP, EK-Online-ALP, and EK-Eval-ALP. Below, we outline the implementation details for the different families of methods:

- **Non-EK methods** For methods that do not rely on expert knowledge, competence and LP estimations are computed individually for each goal.
- **EK methods** For methods incorporating expert knowledge, goals are grouped into five buckets, with competence and LP computed at the bucket level.
- **Online methods** In online methods, a buffer of size M stores goal-outcome pairs. The agent's current competence is estimated from the most recent half of the buffer, while past competence is derived from the oldest half. Non-EK methods maintain a separate buffer for each goal, whereas EK methods use a buffer for each bucket.
- **Eval methods** In evaluation-based methods, the agent's competence is assessed every N episodes. For non-EK methods, the agent is evaluated k times on each individual goal, while for EK methods, evaluations are performed k times per bucket.

The hyperparameters for Online methods are presented in Table E.5, while those for Eval methods are shown in Table E.6.

Table E.5: Online methods hyperparameters

Variable	Value
ϵ start	1
ϵ end	0.2
ϵ decay period	320
Buffer size	100

Table E.6: Eval methods hyperparameters

Variable	Value
ϵ start	1
ϵ end	0.2
ϵ decay period	320
Eval frequency	1000
Number of eval	2048

E.3.4 Compute budget

To optimize GPU VRAM usage during training, we employ 4-bit quantization. We use a vectorized version of Little-Zoo with 32 instances of the environment running (synchronously) in parallel. In order to accelerate training of our LLM agent (both its policy with online RL and MAGELLAN), we leverage Lamorel[1] to deploy 2 instances of the LLM in parallel. Thus, we distribute both the forward passes to compute actions' probability or LP and training in a Data Parallelism setting. When using Flan-T5 250M, each LLM instance is distributed (Vertical Model Parallelism) over one Nvidia H100 80GB GPUs requiring thus a total of 2 Nvidia H100 80GB GPUs to run an experiment (1 GPU $\times$ 2 LLM instances). For one seed of one ALP method trained in Section 5.2.3, performing 500k training episodes requires 80 GPU hours on the Nvidia H100 80GB.

[1] https://github.com/flowersteam/lamorel

E.4 Additional results

E.4.1 Ablations on the MAGELLAN architecture

In order to implement MAGELLAN, we try various architectures presented in Figure E.2. In all of them, to reduce the computational cost, we froze the LLM and used LoRA adapters Hu et al. (2022) for training the policy (along with its Q-value) and the SR estimator. The architectures tested are:

- **A: Different adapters for the policy and MAGELLAN** (Figure E.2a). This is the architecture adopted in this work. The environment representation for the policy and the goal space representation for the competence estimator are learned separately on different adapters.
- **B: Shared adapters for the policy and MAGELLAN**(Figure E.2b). We test whether a shared useful representation between the policy and competence estimator can emerge through training.
- **C: Shared adapters but only trained with the policy loss**(Figure E.2c). We use a single set of LoRA adapters to train both the policy and the competence estimation module, but apply the latter's gradient only to the MLP outputting the competence. This ablation tests if the policy loss on its own creates semantical relationships between goals. We argue that training of the policy should push towards a clustering of the goal space.
- **D: Adapter only for the policy, frozen LLM representation for MAGELLAN**(Figure E.2d). In this ablation, we test if MAGELLAN can learn to predict competence using the LLM's original latent space. Learning to estimate competence over a fixed representation is closer to prior works such as ALP-GMM Portelas et al. (2020a).

We compare the learning dynamics of the four architectures in Figure E.3. We observe no difference in their ability at learning "grasp" goals. However, the agents with Architecture A continue to progress on other goal types whereas the ones using Architectures B, C, and D stagnate between 50k and 100k episodes before progressing again. At the end of training (500k episodes), agents using architecture A have learned all goal types. The ones based on Architectures B and C learned up to "grow herbivore" goal types. The agents that utilize Architecture D succeed plateaued on "grow plant".

To gain a finer comprehension of the learning dynamics implied by the different architectures, we look in Figure E.4 at the embeddings at the end of training for each architecture.

Architecture A (Figure E.4a) and architecture B (Figure E.4b) shared very similar representations with in both cases goals clustered between "grasp", "grow" and impossible goals. Inside the cluster "grow" the different element types (i.e. plants, herbivores, carnivores) are also clustered. Nonetheless, as seen in Figure E.3, Architecture B does not master "grow carnivore" goals and classified them as impossible. This indicates that

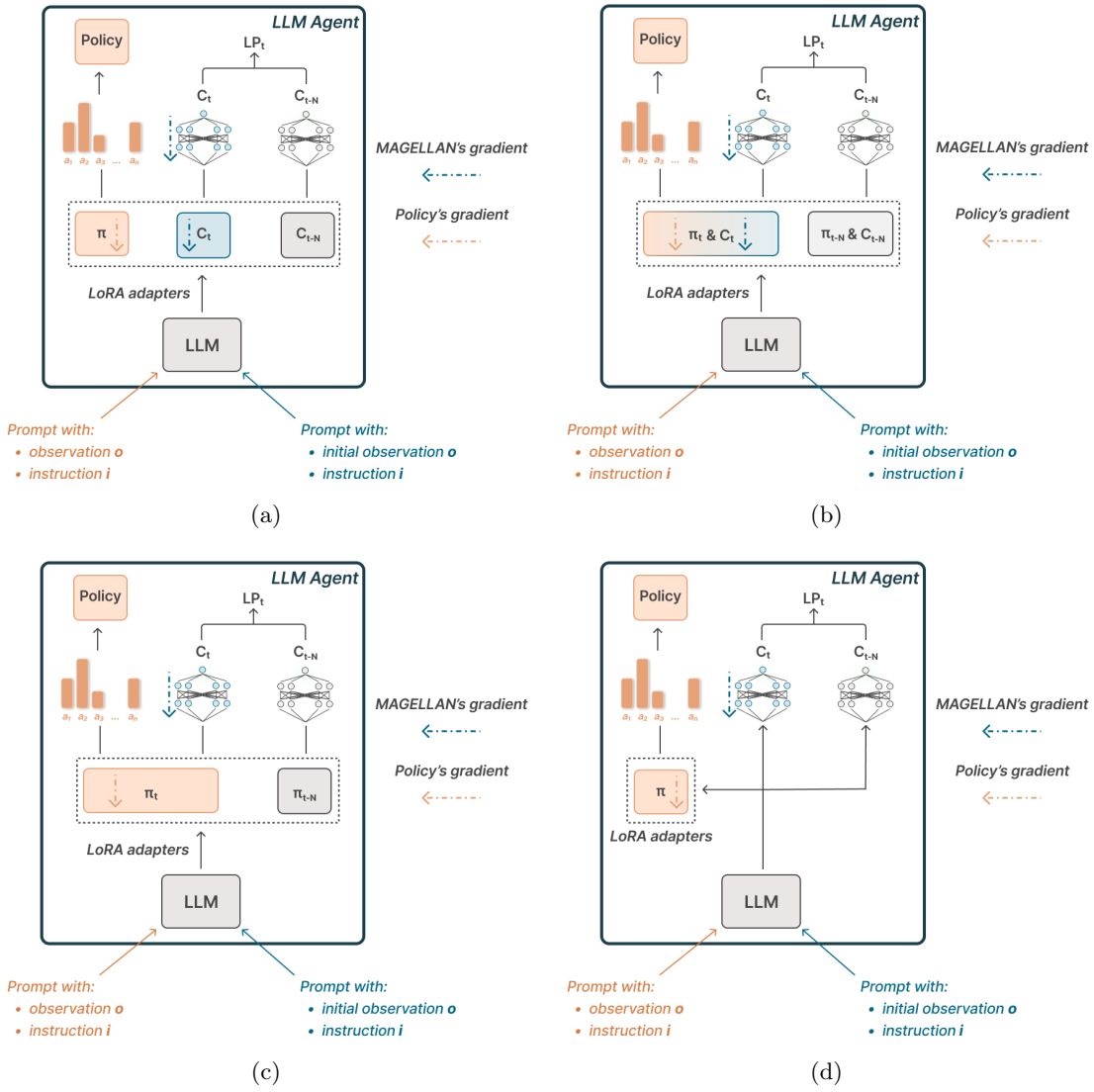


Figure E.2: Different architectural choices in MAGELLAN: (a) we learn separate LoRA adapters between the policy and MAGELLAN (used in this work); (b) we share adapters and update them using both the policy and MAGELLAN gradient; (c) we share adapters but they are only updated by the policy gradient; (d) MAGELLAN directly uses the latent representation produced by the pre-trained LLM.

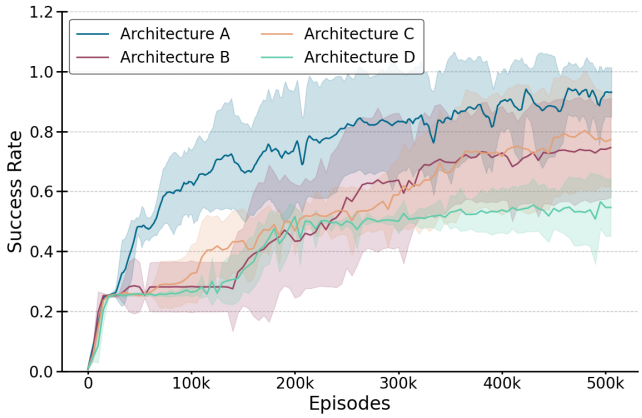
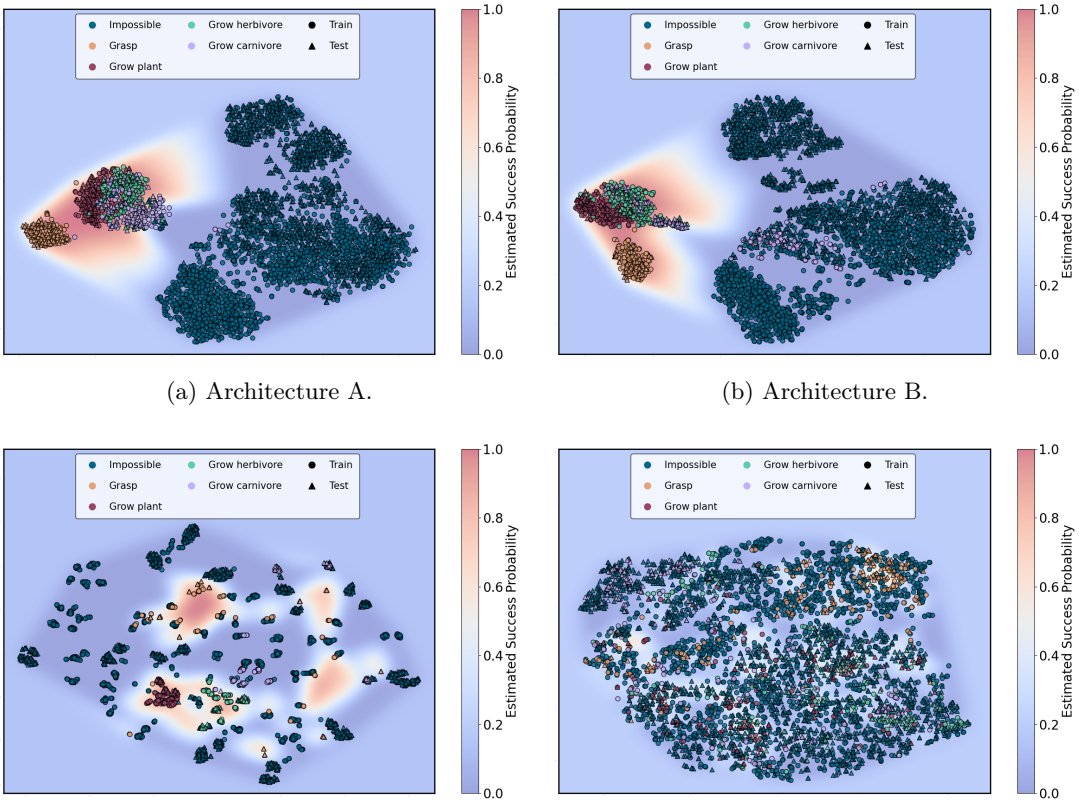


Figure E.3: Training curves of the four different possible architecture for MAGELLAN. We use 8 seeds to plot the mean and the standard deviation (shadow area around the solid line).

obtaining a shared latent space for the policy and MAGELLAN is possible but slows down skill acquisition.

Architecture C (Figure E.4c) where MAGELLAN uses the representation solely changed by the policy still manages to accurately estimate competence. However, predicting competence is much more difficult using this architecture as the goal space is not modified ease competence estimation. Yet, using only the policy gradient still improves the initial latent space of the LLM as shown by Architecture D (Figure E.4d).



(a) Architecture A.

(b) Architecture B.

(c) Architecture C.

(d) Architecture D.

Figure E.4: The LLM embedding space of MAGELLAN displayed using t-SNE with goals used in Q2 (Train) and Q3 (Test), along with MAGELLAN's estimated success probability and linear interpolation between goals. We show the embedding space for a single seed for the four architectures described in Appendix E.4.1 at the end of the 500k training episodes.

E.4.2 Q1. Competence estimation properties

Per-goal competence estimation

In this section, we report the evolution of the per-goal competence estimation of experiments from Section 5.2.3. We show it separately for the three goal space size: 25k (Figure E.5), 50k (Figure E.6) and 100k (Figure E.7).

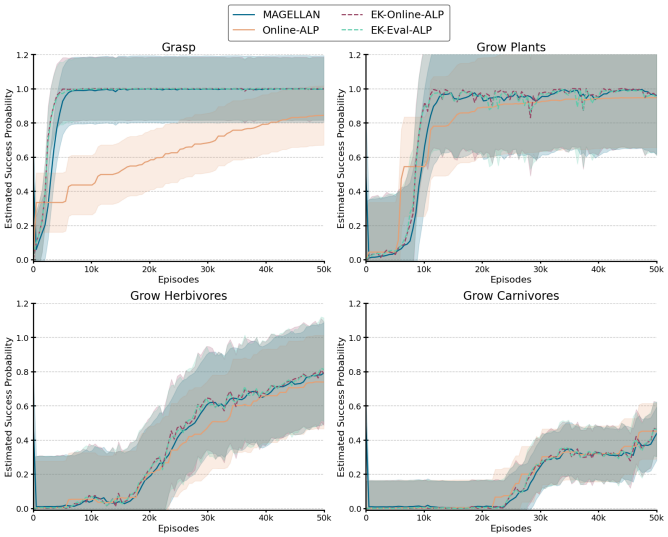


Figure E.5: Evolution of competence estimation for each ALP method on each goal category for 25k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals.

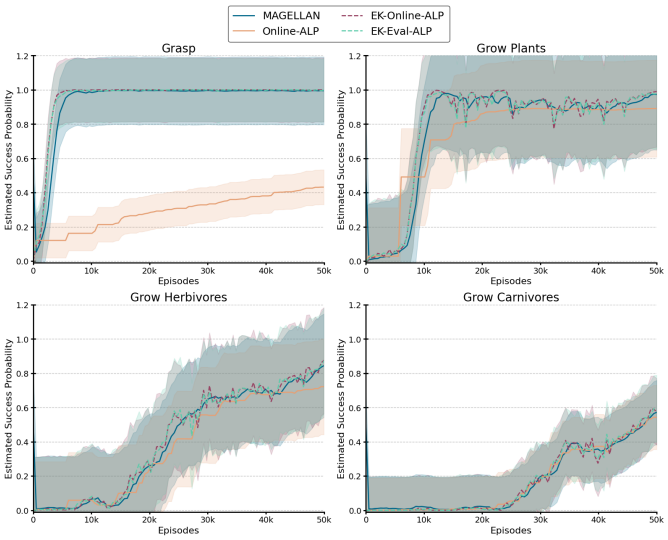


Figure E.6: Evolution of competence estimation for each ALP method on each goal category for 50k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals.

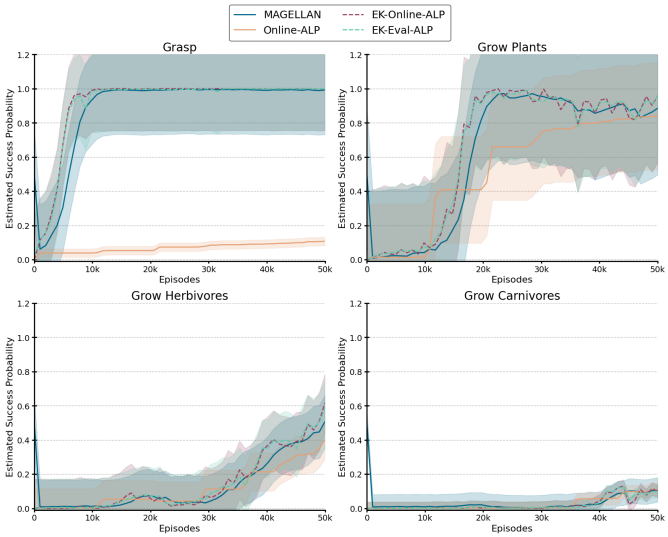


Figure E.7: Evolution of competence estimation for each ALP method on each goal category for 100k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals.

Competence estimation on the BabyAI-text goal-space

We replicated the experiment from Section 5.2.3, simulating a learning agent and estimating its competence online using MAGELLAN and Online-ALP. In this setting, the goal space is drawn from the BabyAI-Text environment, consisting of five goal categories of increasing difficulty: Go to *"object"*, Pick up *"object"*, Open *"door"*, Put *"object A"* next to *"object B"*, and Pick up *"object A"* then go to *"object B"*. As shown in Figure E.8, MAGELLAN outperforms Online-ALP, successfully tracking the agent's competence progression across the different task categories.

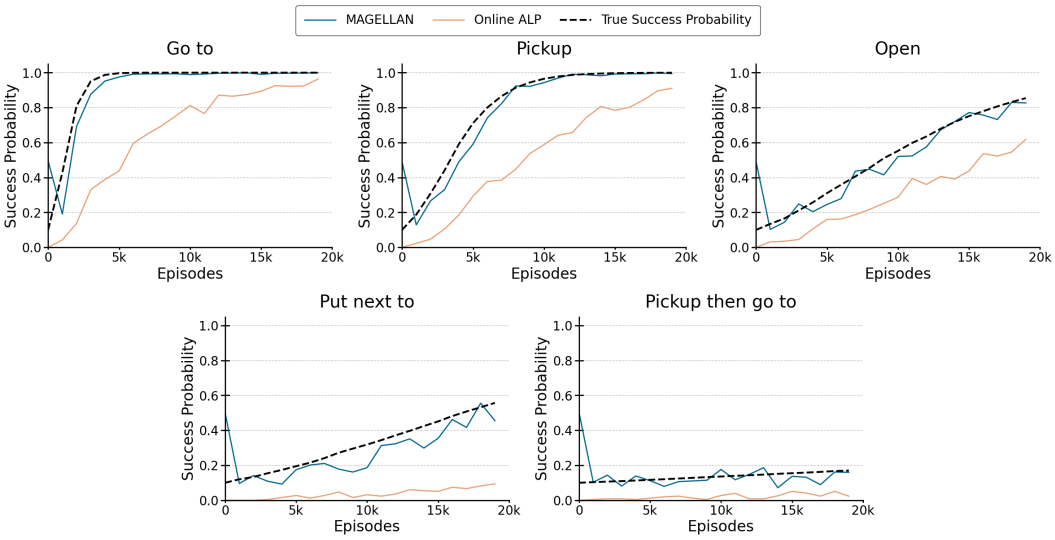


Figure E.8: Competence estimation on BabyAI-Text. MAGELLAN accurately tracks competence across five goal types of increasing difficulty and consistently outperforms Online-ALP.

Impact of the LLM used in MAGELLAN

In this section, we compare different LLMs (Flan-T5-small, Flan-T5-base, and Qwen2.5-0.5B (Qwen et al., 2025)) to evaluate their impact on the competence estimation performance of MAGELLAN. The experiments simulate agent learning using two goal spaces: OpenR1-Math-220k and BabyAI-Text (figures/chapter-5/magellan E.9 and E.10). Results show that all tested LLMs achieve similar performance, suggesting that MAGELLAN's competence estimation remains robust across a range of lightweight language models.

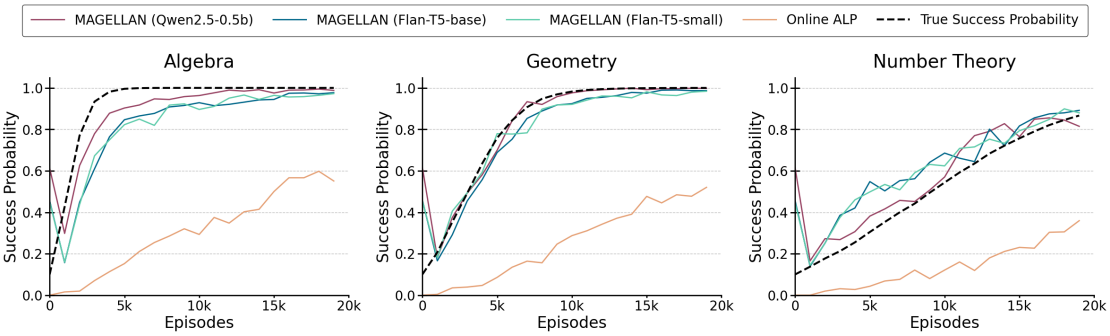


Figure E.9: Effect of LLM choice on MAGELLAN's competence estimation for OpenR1-Math-220k. All models yield similar performance.

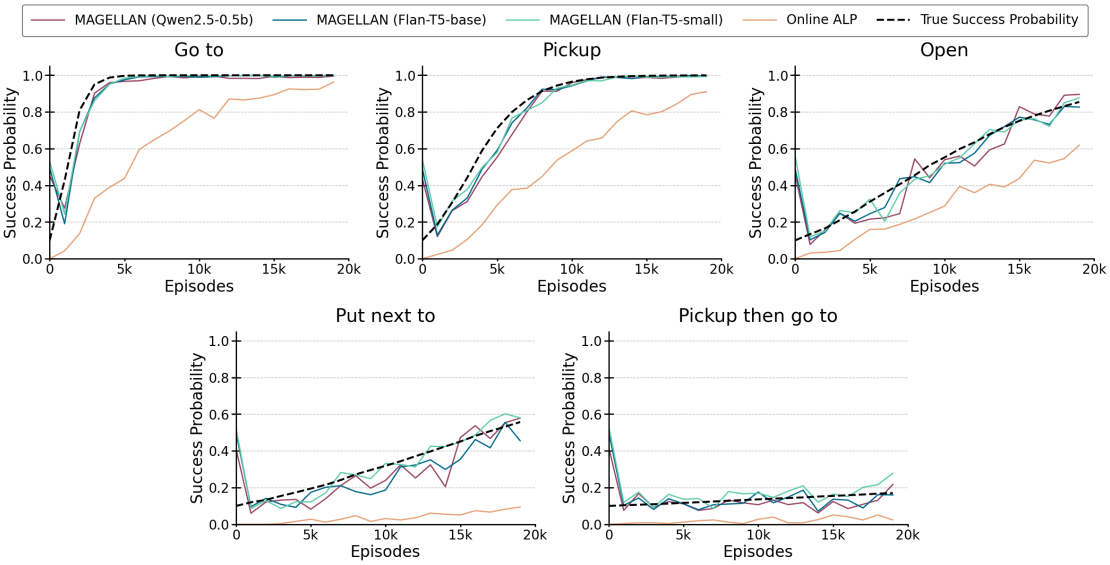


Figure E.10: Effect of LLM choice on MAGELLAN's competence estimation for BabyAI-Text. Performance remains consistent across models.

E.4.3 Q2. Training an LLM-based RL agent with MAGELLAN

Per-goal Success Rate

We detail in Figure E.11 the evolution of the success rate per method and per goal type. All methods perform similarly on the type "Grasp". However, MAGELLAN outperforms all the baselines that do not rely on expert knowledge (Uniform and Online-ALP) on all other types. Learning dynamics fostered by MAGELLAN are close to the ones generated with EK-Online-ALP which relies on expert knowledge. Such results hint towards a clustering of the goal space by MAGELLAN to efficiently and reliably estimate the SR of the LLM agent. This hypothesis is strengthened by the plot of the embedding space in Appendix E.4.4.

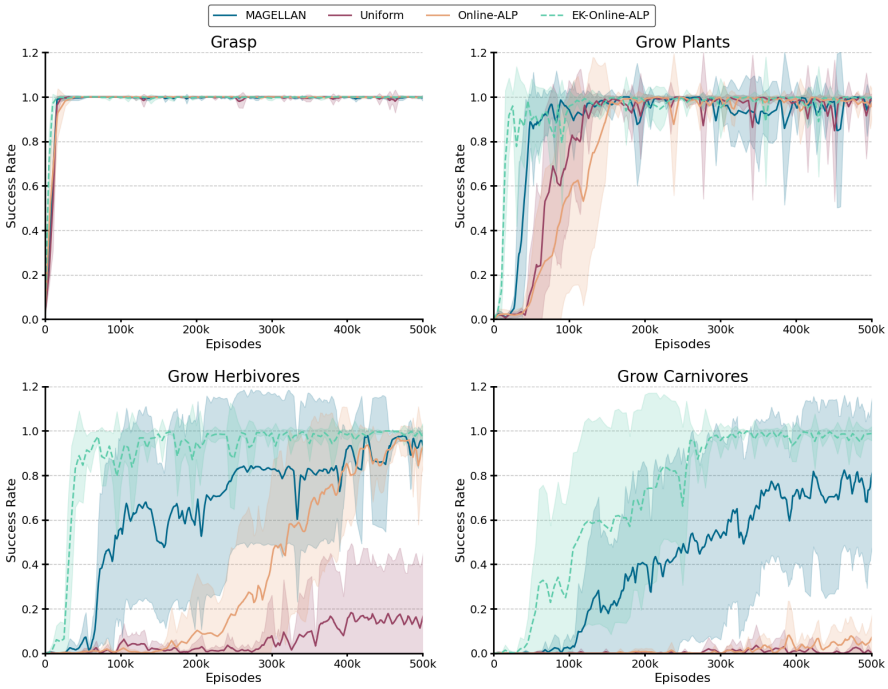
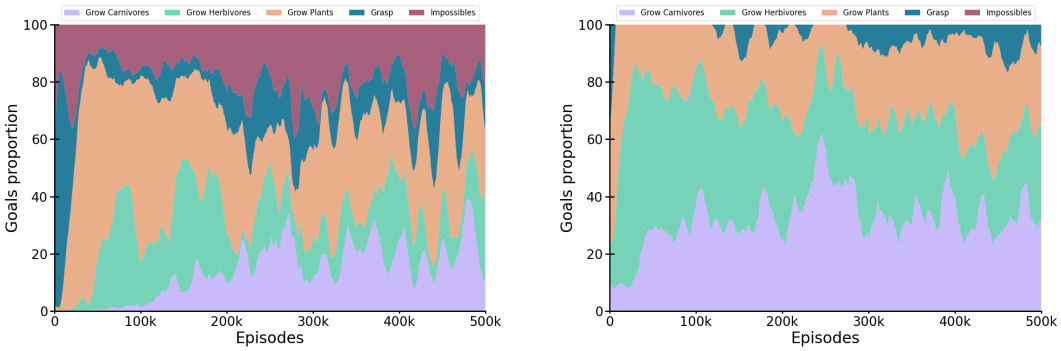


Figure E.11: Evolution of average SR for each ALP method for each goal category. To get the success rate within a category, we evaluate the policy on 64 goals of this category. We use 8 seeds to plot the mean and the standard deviation.

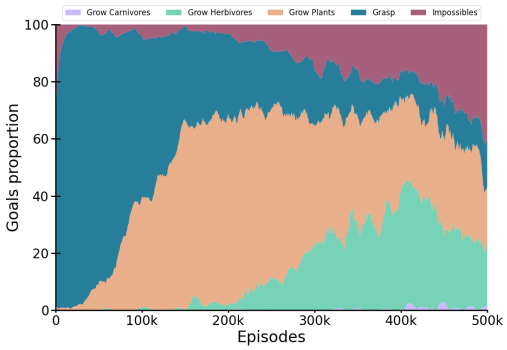
Goal sampling strategies

We analyze the sampling strategies of the different methods. In Figure E.12, it appears that MAGELLAN has a sampling strategy close to the one of EK-Online-ALP. The main difference is that EK-Online-ALP does not select impossible goals thanks to the use of expert knowledge. MAGELLAN's sampling strategy quickly begins to sample goals of type "Grow herbivore" after sampling goals of type "Grow plant", which contrasts with the strategy of Online-ALP. Indeed, Online-ALP mostly selects goals already encountered, getting stuck on one goal type before moving to another. The similarity between MAGELLAN and EK-Online-ALP indicates that MAGELLAN is able to cluster the goal space dynamically, using this information to generalize its competence estimation to sample interesting goals not yet discovered. More results on the dynamic clustering of the goal space by MAGELLAN are given in Appendix E.4.4.



(a) MAGELLAN's sampling strategy.

(b) EK-Online-ALP's sampling strategy.



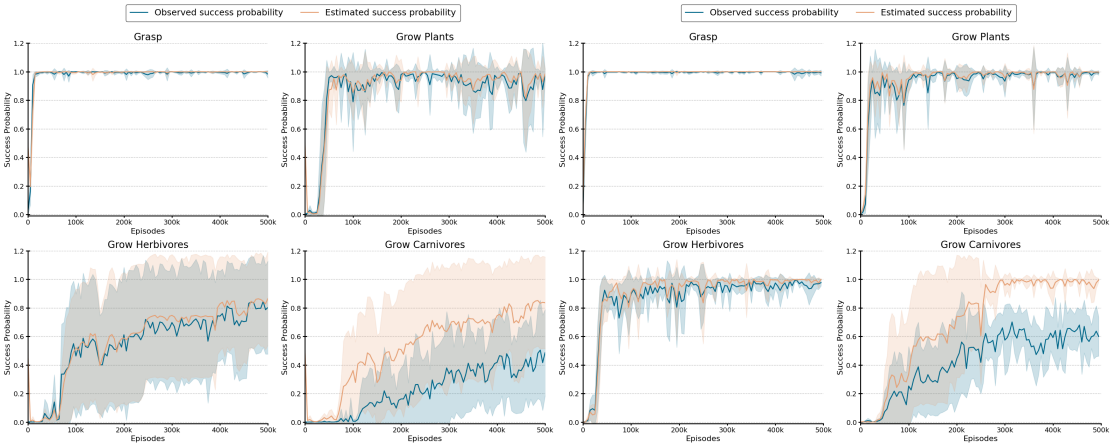
(c) Online-ALP's sampling strategy.

Figure E.12: Goal sampling strategies of MAGELLAN, EK-Online-ALP, Online-ALP. We do not take into account the 20% uniformly sampled goals from the ε-greedy exploration.

E.4.4 Q3. MAGELLAN's generalization abilities

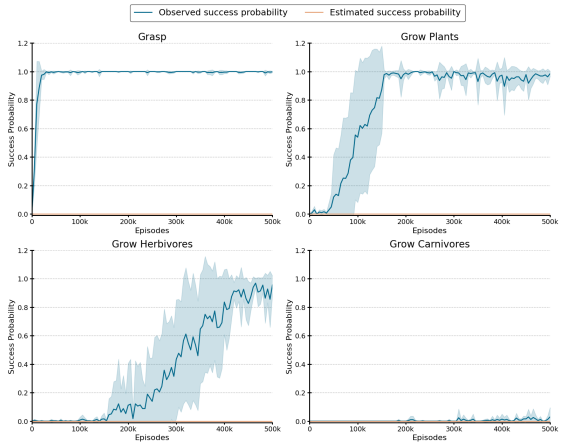
Per-goal Success Probability estimation on test set

In this section, we provide a detailed analysis of the results presented in Section 5.2.3, specifically examining the ability to generalize success probability estimations to test goals across different goal types (see Figure E.13). As expected, Online-ALP exhibits the largest errors, as it can only assign a success probability of 0 to unseen goals. In contrast, both MAGELLAN and EK-Online-ALP achieve highly accurate estimations for "Grasp", "Grow plant", and "Grow herbivore". However, both methods tend to overestimate the generalization abilities of the policy on "Grow carnivore" goals.



(a) MAGELLAN.

(b) EK-Online-ALP.



(c) Online-ALP.

Figure E.13: Per-goal estimation of the success probability for each method. The average result (over 8 seed) is the solid line and the shaded zone represents the standard deviation.

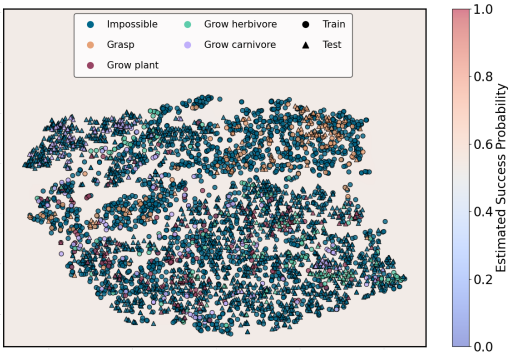
Evolution of the embeddings

In Section 5.2.3, we present the embedding at the beginning and the end of the training for the seed 0.[2] In this section, we explore how the goal space is dynamically modified throughout training with Figure E.14. It is a chronogram of the embedding space used by MAGELLAN projected using t-SNE Maaten & Hinton (2008) with the estimated success probability over the whole space calculated using linear interpolation with a Gaussian filter. Each embedding (except the first one, which is the initial state) is plotted after the agent masters a goal category.

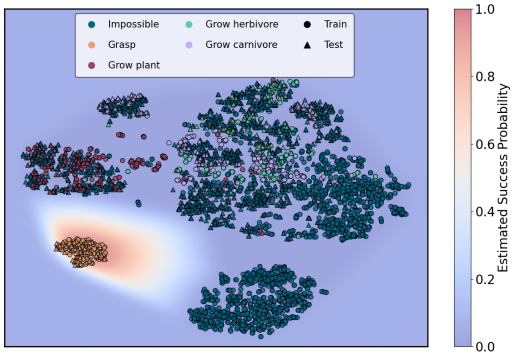
- At the beginning, Figure E.14a, no structure is discernible in the goal space and the estimated success probability is uniform around 0.5.
- After mastering the goals of type "Grasp", Figure E.14b shows several clusters appeared. All "Grasp" goals are in the same cluster with a high estimated success probability zone. The goals of type "Grow plant" are also clustered together and close to the zone of high estimated success probability. That is a hint that MAGELLAN has already picked them as the next candidate type for the curriculum. It also correctly places the "Grow plant" goals from the test set in the same cluster as the ones from the train set. However, it still mixes them with impossible goals, underlying that it does not fully master this type of goal. The "Grow herbivore" and "Grow carnivore" are mixed with other impossible goals.
- After achieving mastery of "Grow plant", in Figure E.14c, we see both "Grasp" and "Grow plant" are correctly clustered in the zone of high success probability, with the goals from the test set correctly placed into the two clusters. The goals from the type "Grow herbivore" and "Grow carnivore" are clustered together and almost separated from the impossible goals. Their cluster is close to the zone of high estimated probability.
- Once "Grow herbivore" has been solved, we see in Figure E.14d that the three achieved goal types are clustered in three groups, all in the zone of high estimated success probability. The "Grow carnivore" are far from this zone but still clustered together, apart from the impossible ones. It also appears that the model places "Grow carnivore" goals from the test close to the "Grow herbivore".
- Finally, when the type "Grow carnivore" is mastered, in Figure E.14e, all the goals from the possible goal types (from both train and test) are placed in the zone of high success probability. The goals of type "Grow X" form three tightly packed clusters. The "Grasp" type is separated from the other types.

What is clear from the chronogram is that MAGELLAN's learning modifies the embedding space in a way that facilitates prediction of the probability of success.

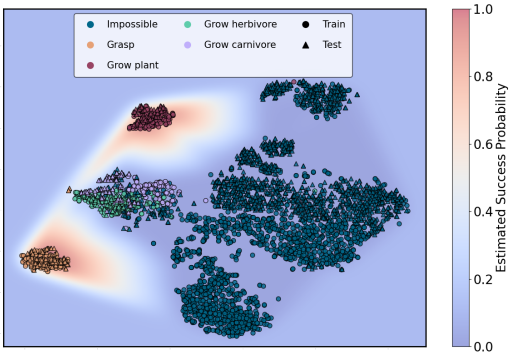
[2]The dynamic is the same for the other seeds.



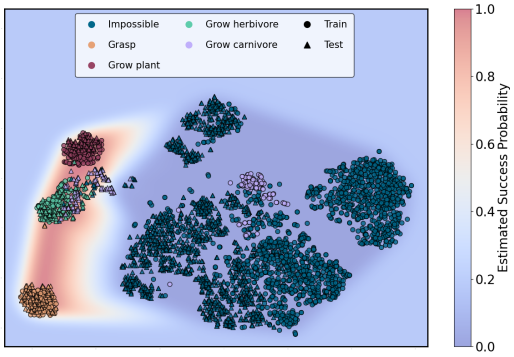
(a) Beginning of training, no goal mastered.



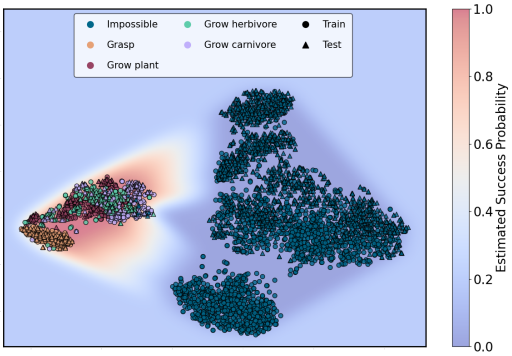
(b) "Grasp" type of goals mastered.



(c) "Grow plant" type of goals mastered.



(d) "Grow herbivore" type of goals mastered.



(e) "Grow carnivore" type of goals mastered.

Figure E.14: Chronogram of the embedding space of the seed 0, at the beginning and after mastering each type of goal.

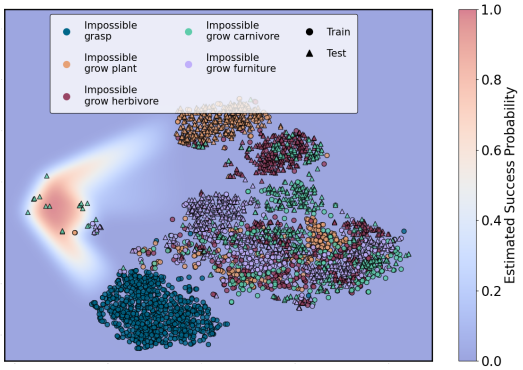
Embedding of impossible goals

As detailed in Section E.1.3, a goal is considered impossible either due to the absence of a required element or when attempting to grow furniture. Accurately identifying impossible goals and generalizing this knowledge is crucial for strong performance in the Little-Zoo environment. In this section, we analyze how MAGELLAN handles impossible goals.

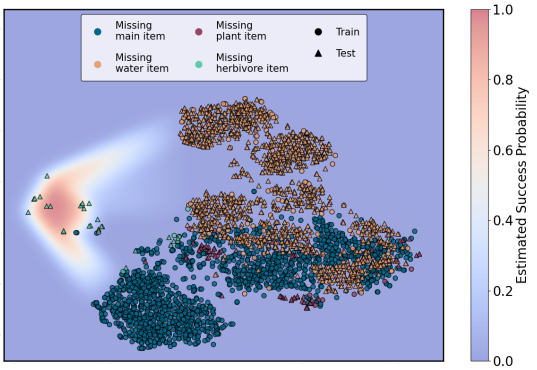
Figure E.15a illustrates the clustering of different categories of impossible goals. The impossible "Grasp" goals form a compact cluster, while the "Grow" goals—categorized

as "Grow plant", "Grow herbivore", "Grow carnivore", and "Grow furniture"—exhibit four less-defined clusters. Additionally, a large, mixed cluster contains various impossible "Grow" goals. However, when examining the same embeddings through the lens of missing elements, as shown in Figure E.15, a distinct structure emerges: MAGELLAN also organizes goals based on the absent element in the scene.

For instance, the previously observed clusters of impossible "Grow" goals largely align with the "missing water" category, as water is a prerequisite for all "Grow" goals. Moreover, the central mixed cluster from Figure E.15a is further clustered into sub-clusters based on the missing element. In particular, the red cluster at the bottom of Figure E.15 represents the absence of a plant, encompassing both "Grow herbivore" and "Grow carnivore" goals. This suggests that MAGELLAN effectively captures underlying environmental dependencies to determine goal feasibility. This ability to infer structural properties of the environment likely contributes to the strong generalization performance observed in Section 5.2.3 and Section 5.2.3.



(a)



(b)

Figure E.15: Plot of the t-SNE-projected embeddings of impossible goals, where each goal is color-coded based (a) on the type of impossible goal (b) on the specific missing element required to make it possible.

E.4.5 Q4. Leveraging the generalization abilities when facing new goals

10 adaptation cases throughout training

In Section 5.2.3, we present adaptation tests where the goal space evolves by replacing goals with new unseen goals from similar categories. We conduct the test by training an agent using MAGELLAN for 500k steps. Then, every $50000 * n$ episodes (with $n \in \{1; 2; 3; 4; 5; 6; 7; 8; 9; 10\}$), we stop training and replace MAGELLAN by one of the four ALP methods of goal sampling (MAGELLAN, EK-Online-ALP, Online-ALP, and Uniform). Finally, we resume the training for 50k steps and measure the SR. For EK-Online-ALP, we make it track the agent's competence based on goals sampled by MAGELLAN during the first phase, allowing it to start with an estimation on the new unseen goals. Our test measures the ability of a method to estimate and quickly update competence on unseen goals. Figure E.16 details all the results obtained at the ten points we use in the experiments.

We can divide the different experiments in 2 typical scenarios:

- **Scenario zero LP (Figure E.16a):** the agent has mastered the "Grasp" and "Grow plant" goals and has 0 ALP across all goals. In this scenario, all ALP estimations are equivalent. EK-Online-LP manages to discover new ALP niches faster as all impossible goals are in the same group.
- **Scenario with LP**:
  - **High LP (figures E.16b, E.16c, E.16d):** the agent is getting a high ALP as it is learning some "Grow carnivore" goals. Here, MAGELLAN outperforms baselines by generalizing its ALP estimation and continuing training on these goals.
  - **Medium LP at the end of training (figures E.16e to E.16j):** the agent has almost learned all the goals, and it only remains few goals in the "Grow carnivore" category on which its performance is not stable. When changing the goal space, Online-ALP and Uniform take a lot of steps to find the remaining goals with LP in the new goal space, destabilizing the agent. As a result, the agent's SR decreases just after the transition and does not recover after 50k training epispdes. MAGELLAN and EK-Online-ALP maintain their high SR.

  We notice that the learning curve of the agents using MAGELLAN and EK-Online-ALP are almost identical, but our method does not rely on expert knowledge and naturally clustered the goal space as seen in Figure 5.11b.

Global Sample Efficiency assessment

To obtain a quantitative analysis of the results from Section 5.2.3, we plot in Figure E.17 the average sample efficiency (after κ episodes) of each method, averaged over the 10 tests, with $\kappa \in \{10000; 20000; 30000; 40000; 50000\}$. The sample efficiency after κ episodes is calculated as $\frac{1}{10} \sum_{n=1}^{10} \int_0^{\kappa} SR(50000 * n + t) - SR(50000 * n)\, dt$.

We observe that the Uniform method is the only approach exhibiting a negative average sample efficiency, which is expected since it predominantly samples impossible goals,

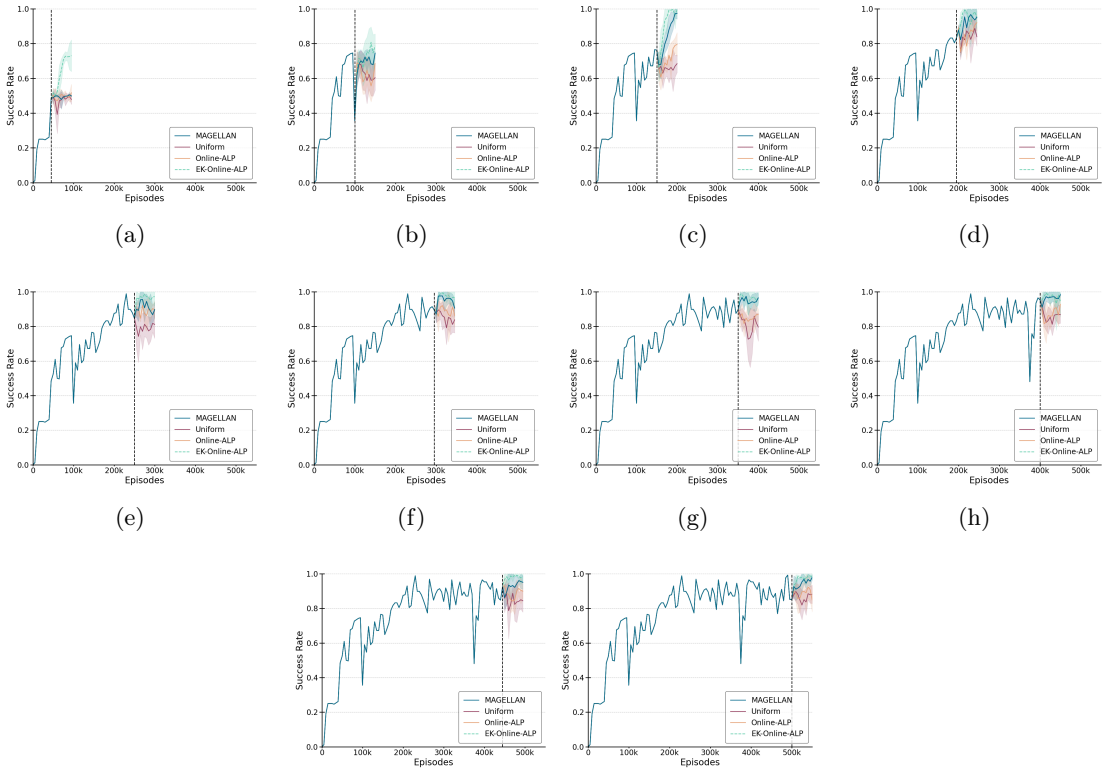


Figure E.16: **Adaptation tests**: Using a single's seed training of 500k episodes, we stop and replace all goals by new unseen goals every 50k episodes. We then resume training by sampling goals using each of our four methods' ALP estimation (MAGELLAN, EK-Online-ALP, Online-ALP, Uniform) and perform 50k training episodes. We report the evolution of observed competence (SR) when evaluating the policies on 64 goals per category from the new training set every 5000 episodes. We report the average competence over evaluated goals along with standard deviation.

thereby destabilizing the agent. In contrast, the other methods demonstrate increasing sample efficiency as κ progresses, reflecting improved success probability estimation and, consequently, more effective goal sampling. However, Online-ALP exhibits only a marginal improvement, as it must continually re-estimate success probabilities across the entire goal space.

Both MAGELLAN and EK-Online-ALP, which leverage generalization to estimate success probabilities for novel goals, achieve higher sample efficiency. Notably, EK-Online-ALP benefits significantly from expert knowledge in clustering the goal space and identifying impossible goals. However, its strong performance is contingent on the assumption that new goals belong to the same categories as those in the training set, which may limit its generalization capacity.

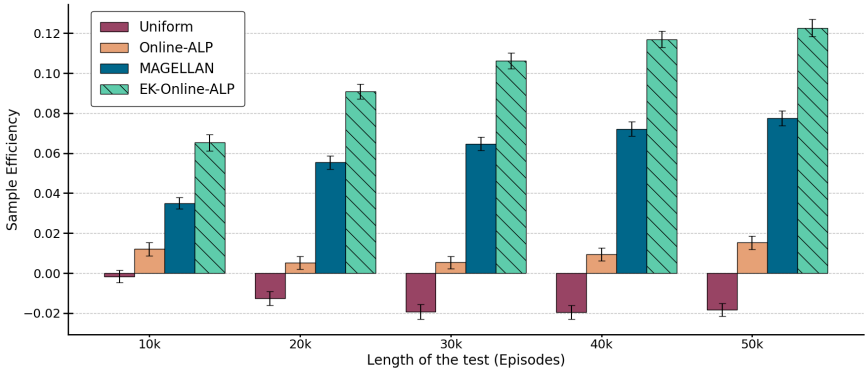


Figure E.17: Average sample efficiency (after κ, the length of the test) of each method average over the 10 tests.

Appendix F

Autonomous discovery of frugal assistance-seeking in LLMs

Contents

F.1 Experimental setup 215
F.1.1 Maths problems 215
F.1.2 MMLU-Pro 217
F.1.3 Methods 218
F.1.4 Prompts 219
F.2 Architectures comparison 219
F.3 Additional results on maths problems 222
F.3.1 Convergence analysis 222
F.3.2 Strategies 224
F.4 Additional results on MMLU-Pro 225
F.4.1 LLMs performances 225
F.4.2 Strategies 225

F.1 Experimental setup

In this section, we provide more details on our experiments.

F.1.1 Maths problems

Rewards

We begin by explaining our reward function R. Given an answer, we first verify that it matches the expected syntax: (1) the answer is provided between "<answer>X</answer>" tags and (2) the answer is a number. If this is not the case, the answer gets a reward of -2. We then compare the answer with the expected answer by computing their absolute

difference d and computing the following reward function:

$$R(d) = \begin{cases} -1 & \text{if } d \geq 10^{-3} \\ 0.05 & \text{if } d \geq 10^{-4} \\ 0.2 & \text{if } d \leq 10^{-5} \\ 0.5 & \text{if } d \leq 10^{-6} \\ 0.8 & \text{if } d \leq 10^{-7} \\ 1 & \text{if } d \leq 10^{-7} \end{cases} \tag{F.1}$$

When using an LLM to solve operations, we sample at most 20 tokens (with "</answer>" and the end-of-sequence token stopping the generation), without applying any specific generation parametrization.

Optimal strategies

We then provide further information on the optimal strategies. For $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, we compute the theoretical global utility U of each tool on each operator given that:

- "A" has a rounding precision of 6 on additions, 4 on divisions, and a cost of -0.85
- "B" has a rounding precision of 4 on additions, 5 on divisions, and a cost of -0.025
- "C" has a rounding precision of 5 on additions, 6 on divisions, and a cost of -0.2

We report the per-tool theoretical global utility as a function of β in Figure F.1, the best tool to call in Table F.1, as well as the per-objective theoretical reward in Figure F.2.

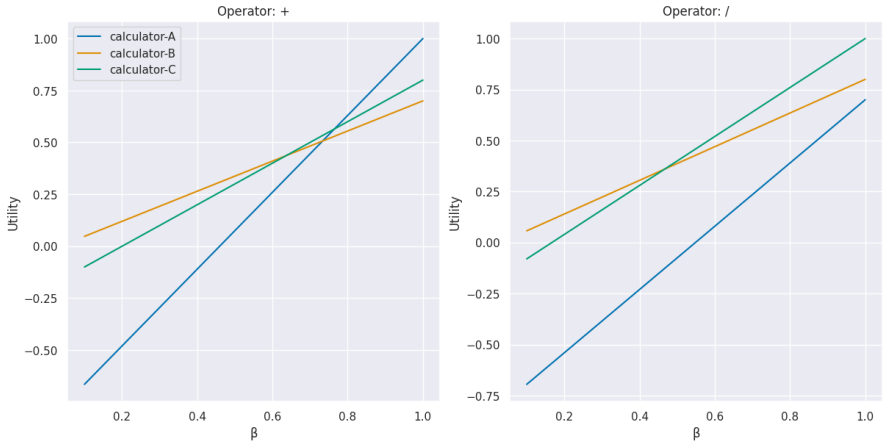


Figure F.1: Per-tool theoretical global utility as a function of β.

Table F.1: Theoretical optimal calculator tool to call per operator for $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

Operator / alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
+	B	B	B	B	B	B	C	A	A	A
/	B	B	B	B	C	C	C	C	C	C

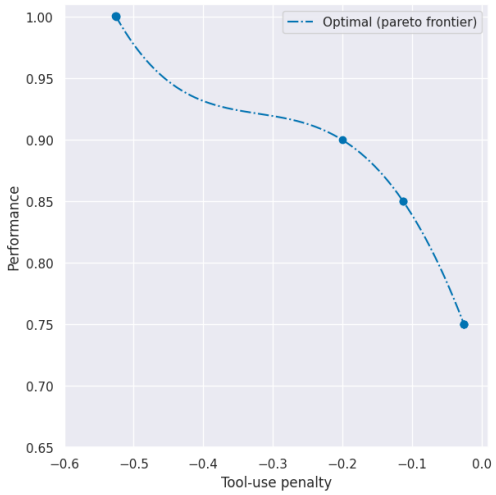


Figure F.2: Theoretical per-objective performance of the optimal strategy for $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

F.1.2 MMLU-Pro

We use the MMLU-Pro dataset from Wang et al. (2024d) in our experiments. We construct our tasks by concatenating the question and four possible answers, each associated with a letter. These answers are deterministically selected around the ground truth answer such that all our experiments use the same task spaces $\mathcal{T}$ and $\mathcal{T}_{test}$. We then filter out tasks containing more than 600 characters. This threshold has been selected such that more than 80% of the "test" split tasks are kept (see Figure F.3).

We augment each task's prompt with the "test" split last sample shown as an example (see Appendix F.1.4). As for maths problems, we expect the answer to be sent between "<answer>X</answer>" tags (with "X" being the answer's letter). For all LLMs, when asked to select a problem's answer, we sample at most 10 tokens (with "</answer>" and the end-of-sequence token stopping the generation), without applying any specific generation parametrization (e.g., nucleus sampling).

In addition to our main LLM, we propose three other LLMs for assistance: Qwen 2.5 14B, Llama 3.1 70B, and Llama expert. The latter is a Llama 3.2 1B that we fine-tuned to generate the ground truth answer for 20 epochs with LoRA on tasks from both $\mathcal{T}$ and $\mathcal{T}_{test}$, that belong to the "maths" and "physics" categories. This constitutes a dataset of 2537 tasks.

Regarding the reward functions, we design R to return 1 if the letter between "<answer></answer>" tags corresponds to the correct answer's letter and 0 otherwise. For C,

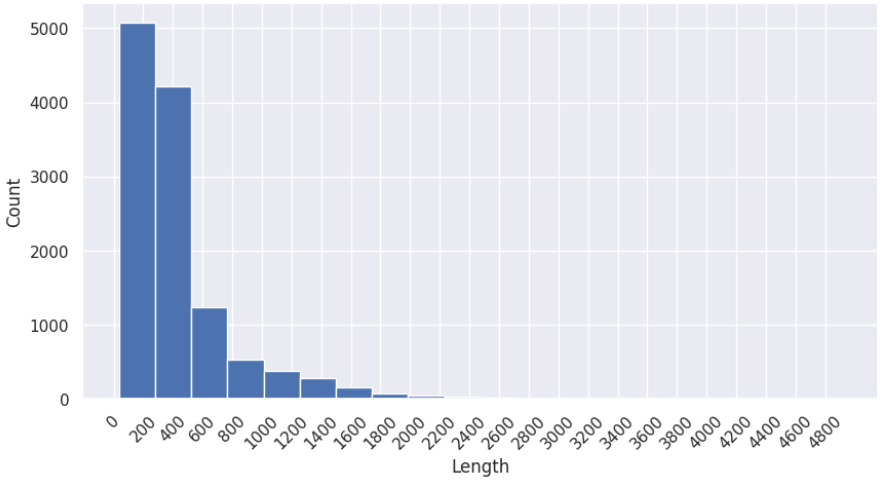


Figure F.3: Repartition of MMLU-Pro tasks in the "test" split by their length.

we apply a per-thousand-tokens-generated cost to each LLM following the cost in dollars applied by online APIs serving those LLMs (e.g., https://www.helicone.ai/llm-cost):

- Our main LLM (i.e., Llama 3.2 1B or Qwen 2.5 0.5B): 0
- Qwen 2.5 14B: 0.00003
- Llama 3.1 70B: 0.00009
- Llama expert (3.2 1B): 0.000003

We multiply this cost by the number of tokens generated divided by 1000 and rescale this cost by multiplying it by -10000 to obtain our reward C. Note that in these experiments, a fixed number of tokens is expected for any LLM's answer (i.e., the selected answer's letter and the "<answer></answer>" tags). Consequently, returning an answer that does not follow the expected format with more tokens will produce a higher penalty.

F.1.3 Methods

This section provides details on the estimators evaluated in this work.

For our two average-based baselines, we do not keep the history of sampled tasks but rather keep, and directly update, the average utility with each new sample with an update weight $\alpha = 0.2$.

For our LLM-based estimator, we use MLPs with two hidden layers of 1024 units (with sigmoid activations). We keep a buffer H of 1024 tasks and sample a batch of 128 tasks every $F = 32$ tasks. We train our estimator on these batches using 4 steps of gradient descent with a learning rate of 10^{-4} and an Adam optimizer.

F.1.4 Prompts

Here, we provide the prompts given to our LLMs. We first show an example for our LLM-based estimator with a single head (Figure F.5) and with two heads (Figure F.4) using separate networks for each arm with a task from MMLU-Pro. We then provide in Figure F.6, the prompt given to our estimator when using shared networks between arms (and two heads). This corresponds to "Shared two heads" described in Appendix F.2. Prompts for maths operations follow the exact same format.

Finally, we provide an example of the prompt given to an LLM when used to solve a task. Figure F.7 shows an example for maths operations, while Figure F.8 shows an example for MMLU-Pro. For maths problems, the examples are fixed, while for MMLU-Pro, the example is the last sample of the dataset's test split (which we use as $\mathcal{T}$).

<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
You will be asked to solve a task. For this, you have access to multiple LLMs that will help you solve the task (llama_3-2_1B, qwen_2-5_14B, llama_expert, llama_3-1_70B). When given a task, you will have to choose which LLM you send the task to. However, each of these LLMs has a specific cost for calling it. Your objective is to minimize your answer's error on a given task (E) while minimizing the cost used (C).
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Now let's solve the following task: You will receive a question along with 4 possible answers. Return the letter associated to the right answer between <answer></answer> markers.
Now, here's the question you must answer:
Question: Typical advertising regulatory bodies suggest, for example that adverts must not: encourage _____, cause unnecessary _____ or _____, and must not cause _____ offence.
A) Safe practices, Distress, Jealousy, Serious
B) Safe practices, Wants, Fear, Serious
C) Unsafe practices, Wants, Fear, Trivial
D) Unsafe practices, Distress, Fear, Serious
<|eot_id|>

Figure F.4: **Example of a prompt given to Llama 3.2 1B to estimate utilities on a task from MMLU-Pro when using two heads.**

F.2 Architectures comparison

In order to implement $E_\theta(\tau, k, \beta)$, we considered four architectures (see Figure F.9):

- Per-k two heads: $E_\theta(\tau, k, \beta) = \beta R_\theta(\tau, k) + (1 - \beta) C_\theta(\tau, k)$. We use one MLP per

<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
You will be asked to solve a task. For this, you have access to multiple LLMs that will help you solve the task (llama_3-2_1B, qwen_2-5_14B, llama_expert, llama_3-1_70B). When given a task, you will have to choose which LLM you send the task to. However, each of these LLMs has a specific cost for calling it. Your objective is to minimize your answer's error on a given task (E) while minimizing the cost used (C).
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Now let's solve the following task: You will receive a question along with 4 possible answers. Return the letter associated to the right answer between <answer></answer> markers.
Now, here's the question you must answer:
Question: Typical advertising regulatory bodies suggest, for example that adverts must not: encourage _____, cause unnecessary _____ or _____, and must not cause _____ offence.
A) Safe practices, Distress, Jealousy, Serious B) Safe practices, Wants, Fear, Serious C) Unsafe practices, Wants, Fear, Trivial D) Unsafe practices, Distress, Fear, Serious
Your global performance will be assessed with the following repartition between the answer's error E and the cost C: 100.0% on E and 0.0% on C.
<|eot_id|>

Figure F.5: **Example of a prompt given to Llama 3.2 1B to estimate utilities on a task from MMLU-Pro when using a single head.**

<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
You will be asked to solve a task. For this, you have access to multiple LLMs that will help you solve the task (llama_3-2_1B, calculator-A, calculator-B, calculator-C). When given a task, you will have to choose which LLM you send the task to. However, each of these LLMs has a specific cost for calling it. Your objective is to minimize your answer's error on a given task (E) while minimizing the cost used (C).
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Now let's solve the following task: You will receive an operation. Compute its results as precisely as possible and return it between <answer></answer> markers.
Now, here's your task: Compute 49.862056/46.695311.
You call calculator-B.
<|eot_id|>

Figure F.6: **Example of a prompt given to Llama 3.2 1B to estimate calculator-B's utility on a maths task when using two heads.**

<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
You are an expert at solving maths operations. When provided an operation, respond with its result.
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
You will receive an operation. Compute its results as precisely as possible and return it between <answer></answer> markers.
Here are some examples:
Compute 4.324672+2.
<answer>6.324672</answer>
Compute 4.324672/2.
<answer>2.162336</answer>
Now, here's your task: Compute 49.862056/46.695311
<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
<answer>

Figure F.7: **Example of a prompt given to Llama 3.2 1B to solve a task from MMLU-Pro.**

arm with two heads to estimate R and C separately with τ provided in the prompt. See Figure F.9a.

- Per-k one head: $E_\theta(\tau, k, \beta)$ is computed with an MLP with a single head per arm k with τ and β provided in the prompt. See Figure F.9b.
- Shared two heads: $E_\theta(\tau, k, \beta) = \beta R_\theta(\tau, k) + (1 - \beta)C_\theta(\tau, k)$. We use an MLP with two heads to estimate R and C separately, with τ and k given in the prompt. See Figure F.9c.
- Shared one head: $E_\theta(\tau, k, \beta)$ is computed with a single-head feedfoward neural network with τ, β, and k provided in the prompt. See Figure F.9d.

We begin by comparing the results of these architectures on maths problems, studying both their evaluation utility and sample efficiency in Figure F.10. We also compare the different architectures with and without LoRA (i.e., keeping the LLM frozen). Our results show that "Per-k two heads" overall leads to the best results on both LLMs. However, not using LoRA adapters leads to better evaluation utility with high β values for Llama 3.2 1B, while it leads to worse results with Qwen 2.5 0.5B.

We then look at results on MMLU-Pro problems in Figure F.11 and obtain similar results. In particular, not using LoRA adapters seems to again lead to better results for Llama 3.2 1B.

<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
You are an assistant that helps solving problems ranging from maths to question answering. You will be given a problem and must answer the correct response. Follow carefully the instructions on how to format your answer.
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
You will receive a question along with 4 possible answers. Return the letter associated to the right answer between <answer></answer> markers.
Here are some examples:
Question: The frequency range of a commercially broadcast FM signal is 88 to 108 MHz, with carrier swing of 125 kHz. Find the percentage modulation of the signal.
A) 83.3%
B) 93.8%
C) 57.1%
D) 100%
<answer>A</answer>
Now, here's the question you must answer:
Question: Typical advertising regulatory bodies suggest, for example that adverts must not: encourage _____, cause unnecessary _____ or _____, and must not cause _____ offence.
A) Safe practices, Distress, Jealousy, Serious
B) Safe practices, Wants, Fear, Serious
C) Unsafe practices, Wants, Fear, Trivial
D) Unsafe practices, Distress, Fear, Serious
<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
<answer>

Figure F.8: **Example of a prompt given to Llama 3.2 1B to solve a maths problem.**

F.3 Additional results on maths problems

In this section, we provide additional results for our experiments with maths problems and calculator tools.

F.3.1 Convergence analysis

We show the regret evolution for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ in Figure F.12. All methods converge to a near-zero regret for $\beta \leq 0.4$, as the optimal strategy is the same for both operators (see Table F.1). However, up to $\beta = 0.8$, using the same strategies for both operators still empirically leads to zero regret, as evidenced by the results of "Average" and Appendix F.3.2. For higher β values, only "Average per-operator" and our approach converge towards a near-zero regret.

We also show the evolution of the per-arm and per-operator estimated utility for

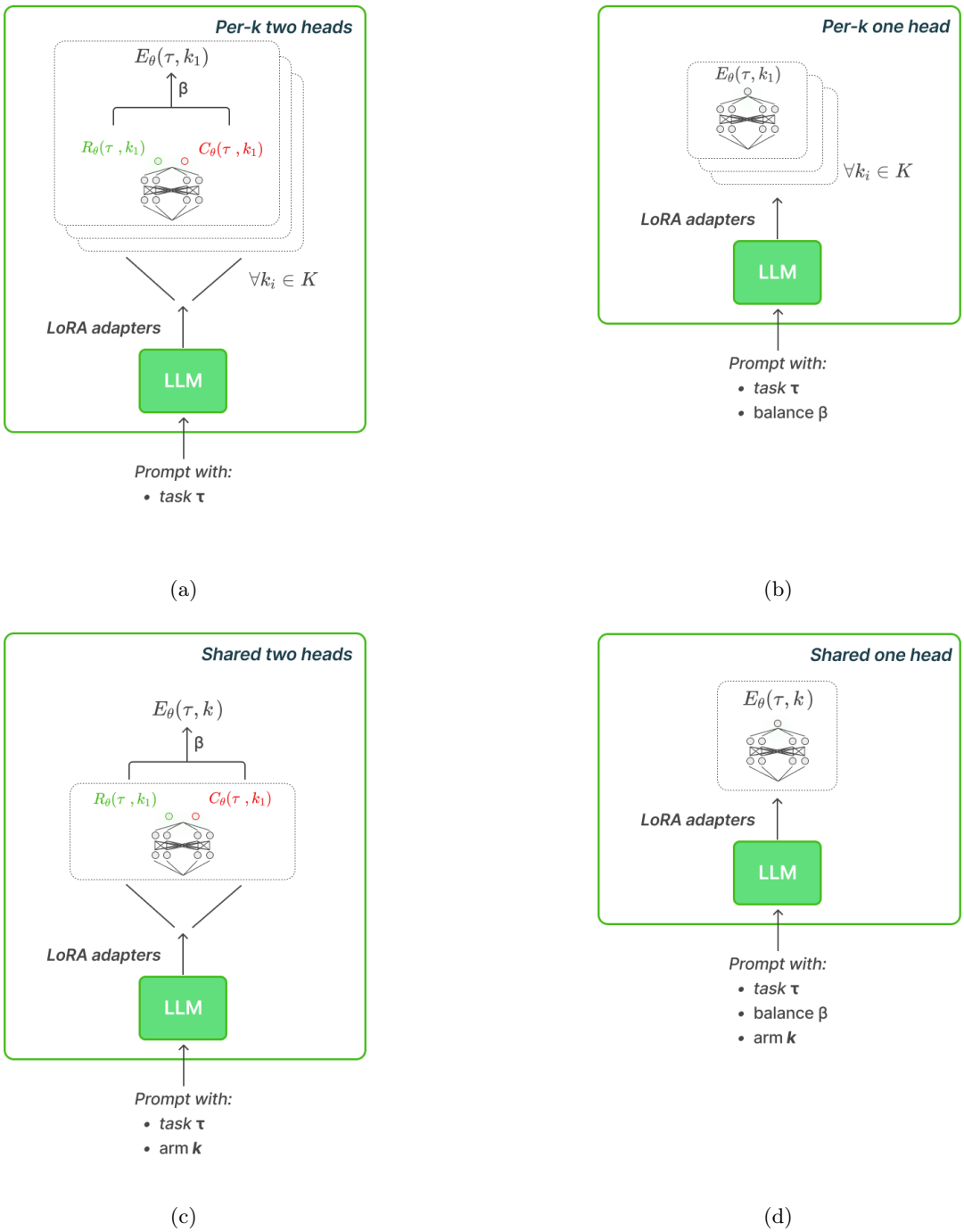
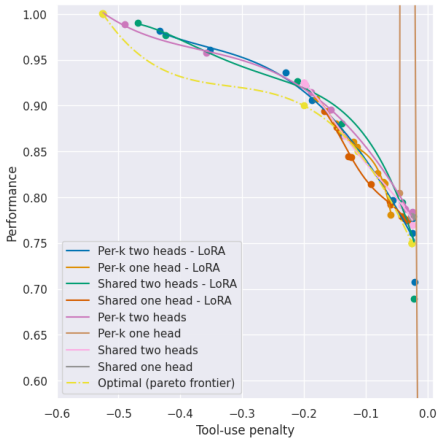
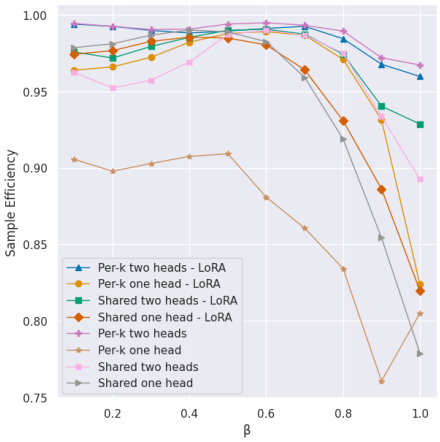


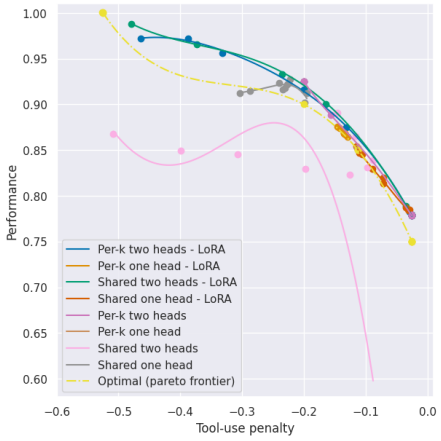
Figure F.9: Different architectures for $E_\theta(\tau, k, \beta)$.



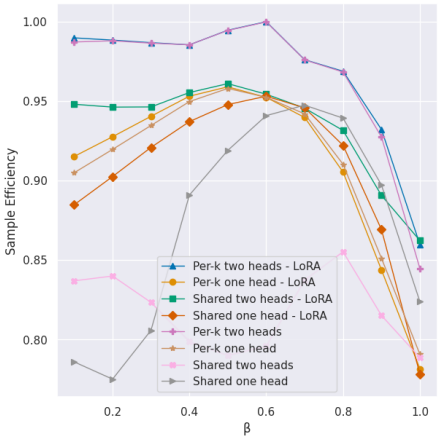
(a) Evaluation utilities of Llama 3.2 1B on maths problems.



(b) Regret sample efficiency of Llama 3.2 1B on maths problems.



(c) Evaluation utilities of Qwen 2.5 0.5B on maths problems.



(d) Regret sample efficiency of Qwen 2.5 0.5B on maths problems.

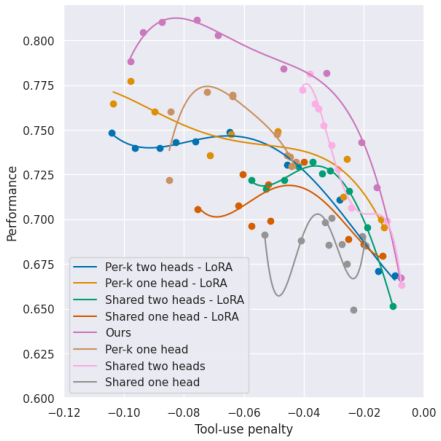
Figure F.10: Comparison of the different architectures for $E_\theta(\tau, k, \beta)$ on maths problems. We show the per-objective final evaluation utility and regret sample efficiency over evaluation tasks for $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ on maths problems. Results are averaged over 32 evaluation tasks and 4 seeds.

$\beta = 1$ in Figure F.13. These results provide more insights as to why our estimator is slower to converge for $\beta = 1$: it tends to underestimate the utility of the optimal tool (i.e., "calculator-A") as this tool is suboptimal for smaller β and slowly converges towards the true utility.

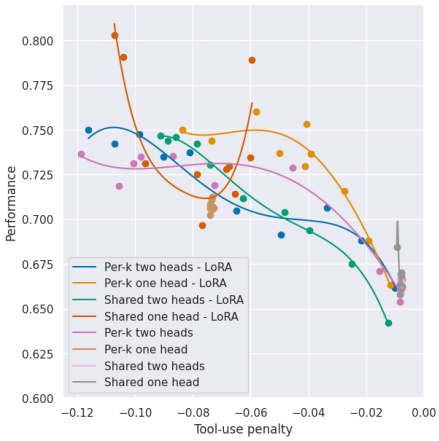
F.3.2 Strategies

In this section, we show the different strategies each estimator evaluated in Section 5.3.3 led to.

We begin by showing the percentage of calls of each arm (i.e., the LLM or the tools) obtained in the evaluation performed after 2048 tasks in Figure F.14. The results are aggregated over the four seeds of each experiment. While Table F.1 shows a single



(a) Evaluation utility of Llama 3.2 1B on MMLU-Pro problems.



(b) Evaluation utility of Qwen 2.5 0.5B on MMLU-Pro problems.

Figure F.11: Comparison of the different architectures for $E_\theta(\tau, k, \beta)$ on MMLU-Pro problems.

theoretical optimal strategy for each β and operator, empirical results from "Average per-operator" show that multiple optimal strategies exist in some cases (e.g., for both additions and divisions at $\beta = 0.6$). This is explained by the fact that, while the theoretical rounding precision of each tool is known, some randomly generated operations can empirically lead to results that are not rounded by a tool (e.g., 8.191039+5.524591 with "calculator-B" leads to an $R = 1$ instead of the theoretical $R = 0.8$). Overall, the results in Figure F.14 show that our LLM-based estimator consistently leads to similar strategies to "Average per-operator", showing that the estimator learned to capture how operators impact tools' utility.

We also show in Figure F.15 the evolution of the strategy when using our estimator for $\beta = 1$.

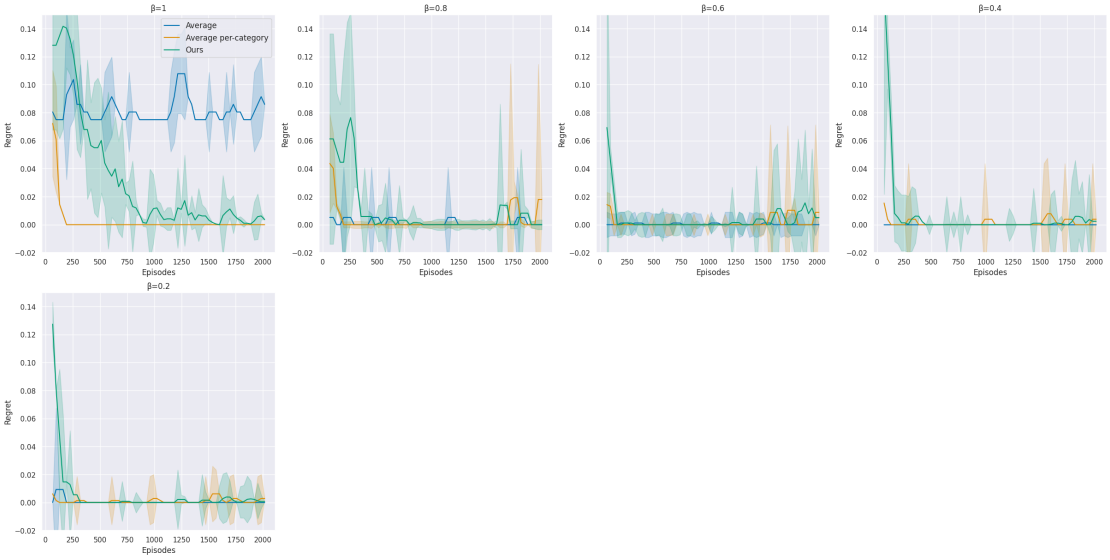
F.4 Additional results on MMLU-Pro

F.4.1 LLMs performances

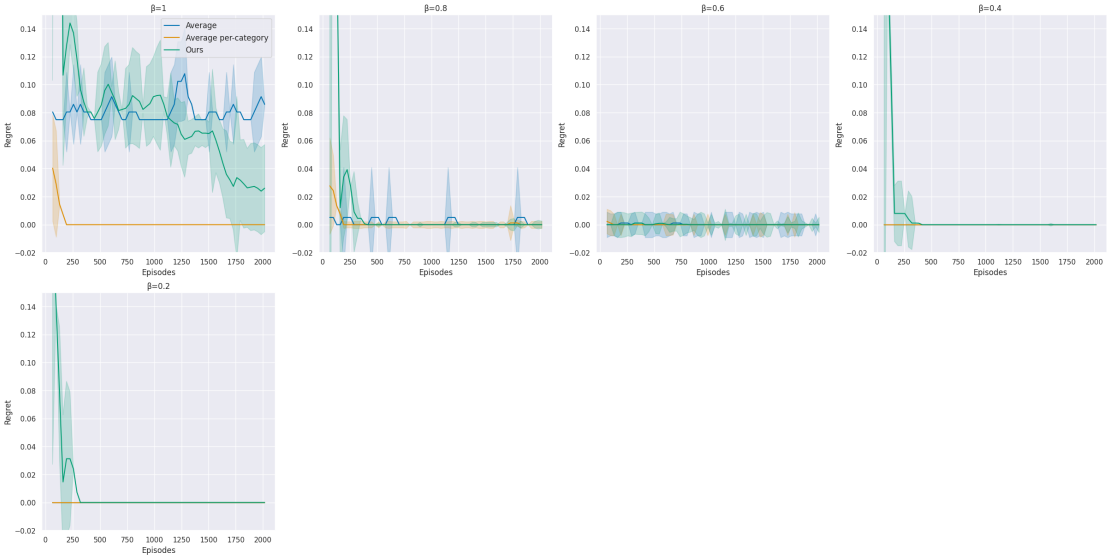
In this section, we report the outcome-based reward R over both $\mathcal{T}$ ("test" split of the dataset) and $\mathcal{T}_{test}$ ("validation" split of the dataset) using a single trial and four seeds for each LLM considered in experiments from Section 5.3.3. We show results in Table F.2 and Table F.3 for the "validation" and "test" splits respectively.

F.4.2 Strategies

In this section, we show the different strategies each estimator evaluated in Section 5.3.3 led to. We report these in Figure F.16, where we show for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ the percentage of calls of each LLM, obtained in the evaluation performed after 2048 tasks. The results are aggregated over the four seeds of each experiment.

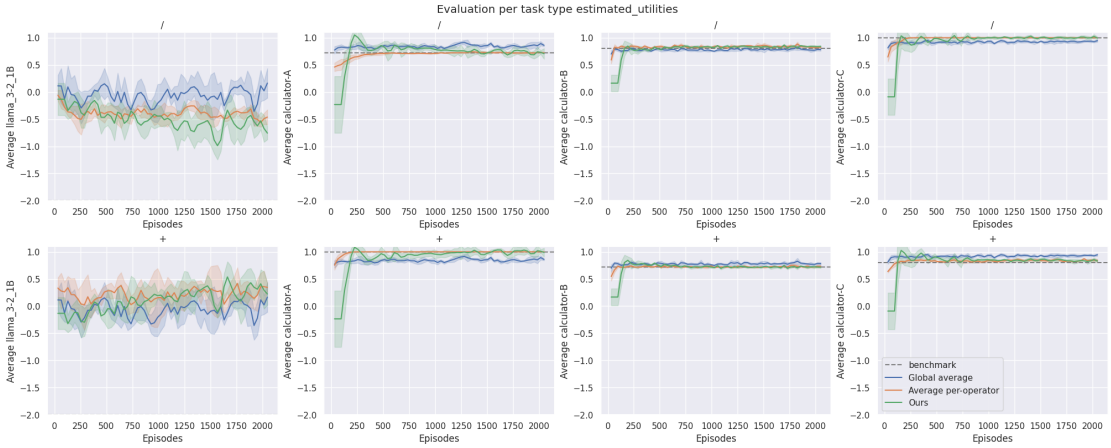


(a) Llama 3.2 1B

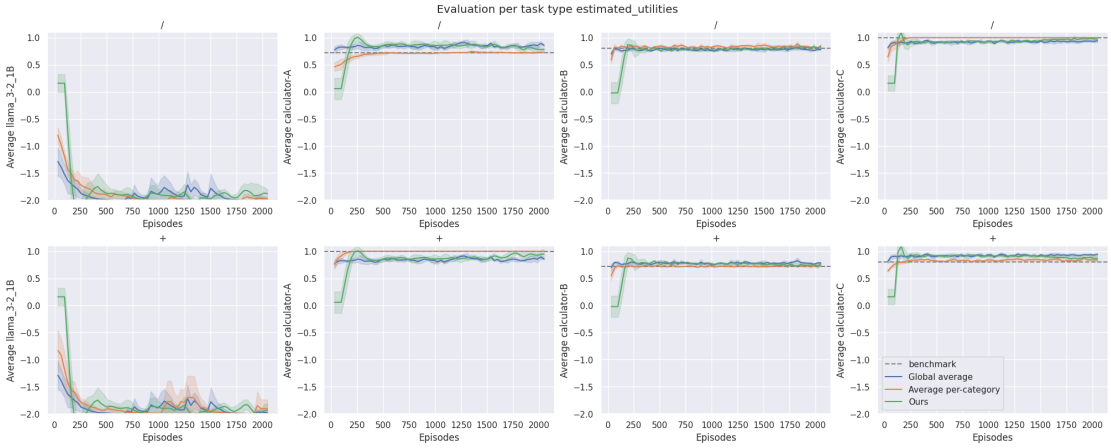


(b) Qwen 2.5 0.5B

Figure F.12: Regret evolution on evaluation tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Results are averaged over 32 evaluation tasks and 4 seeds.



(a) Llama 3.2 1B



(b) Qwen 2.5 0.5B

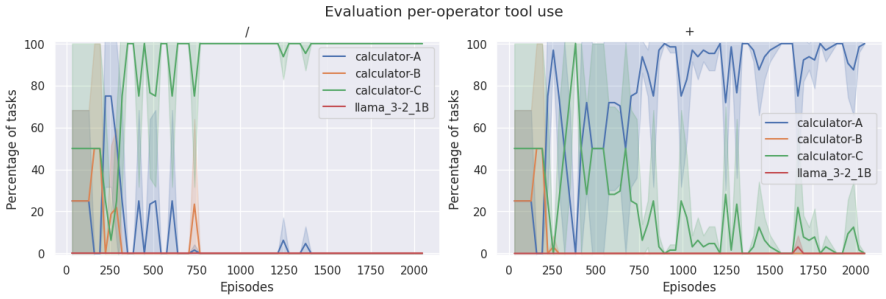
Figure F.13: Evolution of estimated utilities on evaluation tasks for each arm with $\beta = 1$. Results are averaged over 32 evaluation tasks and 4 seeds.

Table F.2: Performance on R over MMLU-Pro "validation" split (used as evaluation set) using four seeds with one trial per problem. We report the average result and standard deviation.

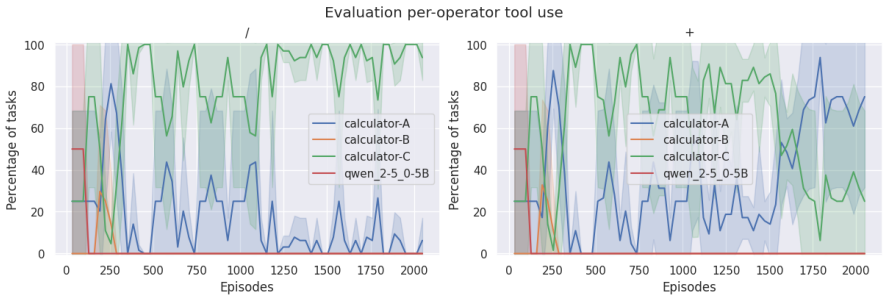
Domain	Llama 3.2 1B	Llama 3.1 70B	Qwen 2.5 14B	Qwen 2.5 0.5B	Llama expert
biology	0.0 (±0.0)	**1.0 (±0.0)**	0.59 (±0.49)	0.0 (±0.0)	0.8 (±0.4)
business	0.12 (±0.32)	0.88 (±0.32)	0.55 (±0.5)	0.0 (±0.0)	**0.95 (±0.22)**
chemistry	0.0 (±0.0)	0.44 (±0.5)	0.4 (±0.49)	0.0 (±0.0)	**0.6 (±0.49)**
computer science	0.0 (±0.0)	**1.0 (±0.0)**	0.44 (±0.5)	0.0 (±0.0)	0.45 (±0.5)
economics	0.0 (±0.0)	**0.96 (±0.2)**	0.6 (±0.49)	0.0 (±0.0)	0.8 (±0.4)
engineering	0.0 (±0.0)	0.32 (±0.47)	0.11 (±0.31)	0.0 (±0.0)	**0.8 (±0.4)**
health	0.04 (±0.2)	0.44 (±0.5)	0.43 (±0.5)	0.0 (±0.0)	**0.74 (±0.44)**
history	0.0 (±0.0)	**0.5 (±0.5)**	0.25 (±0.43)	0.0 (±0.0)	0.5 (±0.5)
law	0.05 (±0.22)	**0.75 (±0.43)**	0.55 (±0.5)	0.0 (±0.0)	0.5 (±0.5)
math	0.0 (±0.0)	0.8 (±0.4)	0.45 (±0.5)	0.0 (±0.0)	**1.0 (±0.0)**
other	0.16 (±0.37)	**0.8 (±0.4)**	0.8 (±0.4)	0.0 (±0.0)	0.6 (±0.49)
philosophy	0.04 (±0.2)	**0.84 (±0.37)**	0.42 (±0.49)	0.0 (±0.0)	0.4 (±0.49)
physics	0.0 (±0.0)	0.5 (±0.5)	0.74 (±0.44)	0.0 (±0.0)	**1.0 (±0.0)**
psychology	0.0 (±0.0)	**1.0 (±0.0)**	0.61 (±0.49)	0.0 (±0.0)	0.15 (±0.36)
Total	*0.03 (±0.0)*	*0.74 (±0.0)*	*0.5 (±0.01)*	*0.0 (±0.0)*	*0.67 (±0.01)*

(a) Llama 3.2 1B (b) Qwen 2.5 0.5B

Figure F.14: Percentage of calls for each arm (i.e., the LLM or tools) obtained in the evaluation performed after 2048 tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Results are averaged over 32 evaluation tasks and 4 seeds.



(a) Llama 3.2 1B



(b) Qwen 2.5 0.5B

Figure F.15: Evolution of tools called by our approach during evaluation for $\beta = 1$ (averaged over 32 evaluation tasks and 4 seeds).

Table F.3: Performance on R over MMLU-Pro "test" split (used as $\mathcal{T}$) using four seeds with one trial per problem. We report the average result and standard deviation.

Domain	Llama 3.2 1B	Llama 3.1 70B	Qwen 2.5 14B	Qwen 2.5 0.5B	Llama expert
biology	0.03 (±0.16)	**0.89 (±0.32)**	0.82 (±0.38)	0.0 (±0.06)	0.61 (±0.49)
business	0.02 (±0.13)	0.48 (±0.5)	0.47 (±0.5)	0.0 (±0.0)	**0.66 (±0.48)**
chemistry	0.0 (±0.0)	0.43 (±0.49)	0.4 (±0.49)	0.0 (±0.03)	**0.59 (±0.49)**
computer science	0.0 (±0.05)	**0.65 (±0.48)**	0.57 (±0.5)	0.0 (±0.05)	0.58 (±0.49)
economics	0.0 (±0.04)	**0.8 (±0.4)**	0.75 (±0.43)	0.01 (±0.09)	0.6 (±0.49)
engineering	0.02 (±0.15)	0.5 (±0.5)	0.39 (±0.49)	0.0 (±0.0)	**0.64 (±0.48)**
health	0.02 (±0.13)	**0.77 (±0.42)**	0.69 (±0.46)	0.01 (±0.08)	0.58 (±0.49)
history	0.04 (±0.2)	**0.84 (±0.36)**	0.72 (±0.45)	0.0 (±0.0)	0.52 (±0.5)
law	0.01 (±0.08)	**0.68 (±0.47)**	0.57 (±0.49)	0.0 (±0.0)	0.6 (±0.49)
math	0.0 (±0.0)	0.52 (±0.5)	0.5 (±0.5)	0.0 (±0.05)	**0.99 (±0.11)**
other	0.03 (±0.18)	**0.71 (±0.46)**	0.64 (±0.48)	0.0 (±0.0)	0.58 (±0.49)
philosophy	0.02 (±0.15)	**0.68 (±0.47)**	0.61 (±0.49)	0.0 (±0.05)	0.54 (±0.5)
physics	0.0 (±0.05)	0.48 (±0.5)	0.47 (±0.5)	0.0 (±0.0)	**1.0 (±0.07)**
psychology	0.02 (±0.13)	**0.84 (±0.37)**	0.79 (±0.41)	0.01 (±0.1)	0.57 (±0.49)

(a) Llama 3.2 1B (b) Qwen 2.5 0.5B

Figure F.16: Percentage of calls for each LLM obtained during the last evaluation performed after 10240 MMLU-Pro tasks for $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Results are averaged over 64 evaluation tasks and 4 seeds.

Appendix G

Jack of All Trades, Master of Some, a Multi-Purpose Transformer Agent

Contents

G.1 Introduction . . . 231
G.2 Related work . . . 232
G.2.1 Transformer for RL . . . 232
G.2.2 Multimodal transformer . . . 233
G.2.3 Multi-task RL . . . 234
G.3 Methodology . . . 234
G.3.1 Model architecture . . . 234
G.3.2 Datasets . . . 237
G.3.3 Training . . . 237
G.4 Experiments and results . . . 238
G.4.1 Text-centric tasks . . . 238
G.4.2 Sequential decision-making tasks . . . 239
G.4.3 Predicting the observations does help . . . 241
G.5 Conclusion . . . 242
G.6 Full results . . . 243
G.7 JAT dataset in depth . . . 251
G.7.1 Sequential decision-making datasets . . . 251
G.8 Image captioning additional examples . . . 257
G.9 Reward as a task determinant . . . 258

G.1 Introduction

Machine learning researchers have long aimed to develop versatile models that can adapt seamlessly to different domains. The recent success of transformers (Vaswani et al., 2017) in NLP, computer vision (CV), and to some extent in RL, has opened new avenues in this quest. In this work, we attempt to extend the boundaries of this success by proposing a single, unified model capable of operating across a wide range of NLP, CV,

and RL tasks using a single set of parameters. This effort not only seeks to challenge the conventional compartmentalization of AI tasks into distinct domains but also aims to establish a more holistic approach to AI model design.

While combining visual and textual tasks has been well-researched, integrating RL tasks remains relatively unexplored and poses distinct challenges. RL tasks are inherently diverse and heterogeneous, making their combination among themselves and with other domains a highly complex exercise. This integration requires dealing with a landscape of different modalities, task complexities, and data volumes across domains and tasks. New questions that arise include: (1) How to design a model and learning method that effectively handles different modalities and data types (sequential decision-making and text-centric)? (2) How to formulate a learning objective that appropriately balances and harmonizes the different modalities, tasks, and domains without bias toward any particular domain or task? (3) How to design a learning strategy that can accommodate the different levels of complexity inherent in different tasks?

These goals are concurrent and, to our knowledge, have only been addressed together by Reed et al. (2022) with the Gato model. Our contributions are characterized by three major advances: (1) Our model features an innovative structure optimized for sequential decision-making tasks. It uniquely assigns each timestep to a corresponding token embedding, resulting in a simpler design. This approach significantly expands the attention window in terms of timesteps compared to Gato (e.g., it is 19 times larger for Atari and more than 25 times larger for Meta-World). (2) In the spirit of open source, we release our code, dataset, and model to the research community. (3) We add observation prediction as an auxiliary task to our model. We demonstrate that this integration significantly contributes to learning a more efficient agent.

Ultimately, our JAT model achieves competitive results on the tasks studied, while being more than 6 times smaller than Gato and relying on a significantly lower training budget. As mentioned above, this new paradigm raises a number of open questions and paves the way for new research. We present a first milestone in this emerging framework and acknowledge the significant potential for improved results.

G.2 Related work

G.2.1 Transformer for RL

Transformer models (Vaswani et al., 2017) are designed to model sequences and, in particular, sequences of words in natural language. However, sequence modeling problems span over a much larger set of domains than only NLP. Several efforts have been made to leverage these models for RL (Li et al., 2023b). In this paper, we focus on modeling RL trajectories (i.e., sequences of observations, actions, and rewards). Modeling such sequences with a transformer was introduced by Chen et al. (2021) and the Decision Transformer (DT) model. In DT, a transformer model is trained with offline RL to take sequences of transitions as input and predict the next action. In particular, Chen et al. (2021) proposed to use returns-to-go (i.e., the return from the current state) to condition actions' generation on both the previous observation and the desired return-to-go. While this has the advantage of explicitly modeling the relations between action selection and

return, using the model at inference requires providing, at every step, a desired return. Liu & Abbeel (2023) proposed to extend this by using hindsight relabeling to better exploit sub-optimal trajectories. Zheng et al. (2022) also extended the DT approach by mixing offline pre-training and online fine-tuning. Finally, Lee et al. (2022) studied how the DT approach scales to a multi-task RL setup where a single policy is learned for multiple games. In terms of sequence structure, some discretize each dimension of the observation and action spaces separately (Reed et al., 2022; Janner et al., 2021; Chebotar et al., 2023), while others associate an embedding with each element of the sequence (Chen et al., 2021; Zheng et al., 2022).

Our work lies in this line of work, as it also leverages transformers to model trajectories. However, our approach (1) uses standard behavioral cloning (BC) instead of conditional BC, relaxing the need to condition the agent by the return-to-go and (2) models a multi-task dataset in which sequences come from very different domains (e.g., control, Atari, visual question answering, see Section G.3.2).

G.2.2 Multimodal transformer

Apart from being widely used in NLP, transformers also thrive in vision and vision-and-language domains. As one of the first works leveraging transformers for vision, Dosovitskiy et al. (2020) introduced Vision Transformer (ViT), a transformer model using image patches for recognition. Following this, a line of work aiming to train multimodal transformers using both text and images emerged, including works such as Flamingo (Alayrac et al., 2022), PaLI (Chen et al., 2023) or IDEFICS (Laurençon et al., 2023). All these models imply the use of an image encoder to obtain image tokens or embeddings that can be given to the transformer alongside text tokens.

These models, typically generating text outputs, are trained for vision-and-language tasks like Visual Q&A. However, recent multimodal transformers focus on decision-making. For example, Jiang et al. (2023b) trained a robot with imitation learning using multimodal prompts to produce motor actions. RT-1 (Brohan et al., 2023) and RT-2 (Zitkovich et al., 2023) use expert demonstrations for real-world robots, with RT-2 building on RT-1 by directly outputting motor actions. Palm-E (Driess et al., 2023) leverages a pre-trained vision language model (VLM) for robotics tasks, producing sequences of text instructions executed by control policies.

Finally, our approach is largely inspired by Gato (Reed et al., 2022), which proposed to train a transformer on both vision-and-language and decision-making tasks without relying on any pre-trained model. The resulting model is therefore smaller than the ones leveraging large VLMs (e.g., Palm-E, RT-2) while still being able to perform both vision-and-language and decision-making tasks. In this paper, we first propose to build a dataset that resembles Gato's dataset, except we only use open-source data sources and release all demonstrations, as well as expert policies we used to obtain these demonstrations. Then, we also leverage a multimodal transformer along with imitation learning for our model, but introduce several improvements, notably in the processing of sequential data and support for continuous values (see Section G.3.1).

G.2.3 Multi-task RL

The quest for a general agent has long been a goal of RL (Bellemare et al., 2013). However, most works have chosen to use a different neural network for each environment. Recent research has revived interest in this objective and explores it through several approaches.

One such approach involves directly extending online learning to multi-task environments (Espeholt et al., 2018; Yu et al., 2020; Song et al., 2020a). These works highlight the potential for positive transfer in multi-task learning, meaning that learning across tasks can be mutually beneficial due to underlying commonalities. However, they also acknowledge the risk of negative transfer, where inter-task interference can impair training. Studies have investigated methods to limit this risk, such as that by Yang et al. (2020), which proposed refined gradient management techniques to mitigate these detrimental effects.

An alternative approach is policy distillation, which involves condensing the behaviors of expert agents into a singular, unified policy (Rusu et al., 2016; Parisotto et al., 2016). While these studies also report positive transfer across tasks (Rusu et al., 2016), they also identify instances of negative transfer. Subsequent research has focused on strategies to minimize this negative transfer (Teh et al., 2017). The reliance on the availability of policies to distill is a limitation. This constraint is notably addressed in (Chen et al., 2021), which proposes conditioning the distilled policy on the desired return thus allowing the use of any policy, including those of non-expert agents. This strategy has been adapted to the multi-task setting by Lee et al. (2022).

Despite the diversity of research in this area, most studies are limited to multi-task learning within a single domain, in comparison to the Gato model (Reed et al., 2022), which learns a large number of domains in a single network. It is the closest baseline to our work.

G.3 Methodology

In this section, we introduce the JAT model, detailing our architectural choices that underpin its effectiveness and highlighting its ability to handle different modalities in sequential decision-making and text-centric tasks. We present the associated dataset, which is notable for its groundbreaking diversity across domains and modalities. Finally, we discuss the learning strategy used in depth.

G.3.1 Model architecture

Embedding mechanism

The model is designed to handle two main categories of data: tasks involving sequential decision-making and text-centric tasks. In text-centric tasks, the model currently supports two modalities: text and image. Although the current version of the model supports image generation, we focus only on tasks that involve text generation. To ease reading,

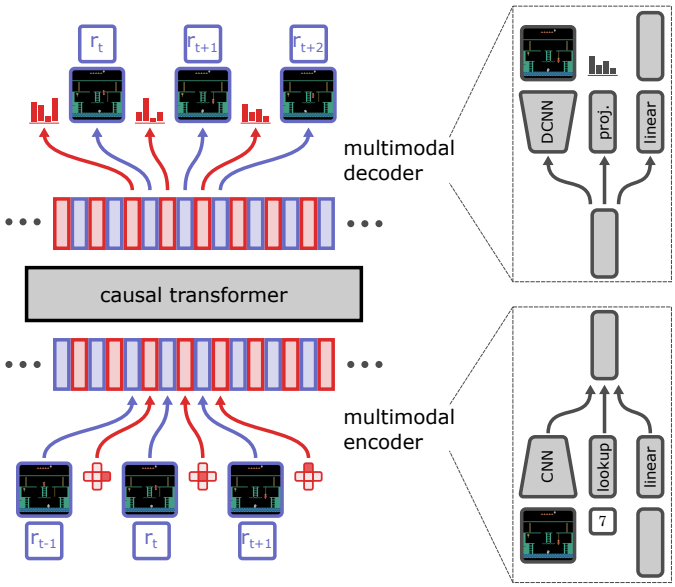


Figure G.1: Architecture of the JAT network. For sequential decision-making tasks, observations and rewards on the one hand, and actions on the other, are encoded and interleaved. The model generates the next embedding autoregressively with a causal mask, and decodes according to the expected modality.

we will refer to it as *text-centric tasks* in the remainder of this paper. Each of these two categories requires a slightly different approach to the embedding process.

In both cases, the resulting sequence is truncated to match the maximum permissible input size of the inner transformer model. Any truncated portion is not discarded; instead, it forms the basis of a new sample. This process may be repeated if necessary, ensuring that no valuable information is lost.

Sequential decision-making tasks – For sequential decision-making tasks, the data comprises a sequence of observations, actions, and rewards. At the embedding stage, these sequences are processed to produce an interleaved sequence of observation embeddings (augmented with the corresponding reward) and action embeddings, denoted as $[\phi(s_0, 0.0), \phi(a_0), \phi(s_1, r_1), \phi(a_1), \ldots]$. Unlike DT (Chen et al., 2021) and Gato (Reed et al., 2022), each timestep is consistently associated with two embeddings: one for the observation and the other for the action, regardless of the modality. This enables JAT to better handle high-dimensional observations, and to provide a much wider, constant attention window in terms of timesteps. As an example, this multiplies the size of the attention window in terms of timesteps by more than 25 for Meta-World. The embedding method employed at a specific timestep is modality-dependent (with H the hidden size of the model):

- Continuous observation: The reward value is appended to the observation vector. This augmented vector is then padded to achieve a uniform length of 377, corresponding to the maximum augmented observation size in the dataset. The embedding vector is

subsequently obtained by passing this padded vector through a linear layer with an output size of H. This layer is consistently used across all timesteps.

- Discrete observation: The observation consists of a vector of integers, each of which is encoded into a continuous vector of size H using a lookup table. Subsequently, a linear layer is applied to reduce the dimensionality to $\lfloor H/50 \rfloor$. Following vector flattening, another linear layer is applied, resulting in an output size of $H - 1$. Lastly, the reward is added to the resulting vector.
- Image observation: The input image is first resized to a uniform dimension of 84×84 using bicubic approximation, normalized, and padded to ensure 4 channels. The image encoder consists of a series of three blocks, each consisting of a convolutional layer, an instance normalization layer, and an attention layer. The output of the last block is flattened and passed through a linear layer, resulting in an embedding vector of size H.
- Continuous action: The process is similar to that of continuous observations, with the exception of the reward component. Notably, the linear layer is shared with the one used for continuous observations.
- Discrete action: In the case of discrete actions, the process is slightly different due to the nature of the input: a discrete action is represented by a single integer, as opposed to a vector of integers for discrete observations. The input is directly mapped to a continuous vector of size H using the same lookup table employed for discrete observations.

Text-centric tasks – For text-centric tasks, each sample includes text, accompanied or not by an image.

- Image data: We employ the ViT architecture, as originally proposed by Dosovitskiy et al. (2020). The image is first cropped to its central square, and resized to 224×224. The image is then normalized and divided into non-overlapping patches of 16×16. Each patch is linearly embedded in a vector of size H.
- Text data: We use the GPT-2 tokenization strategy (Radford et al., 2019), utilizing a byte-pair encoding (Sennrich et al., 2016, BPE) specifically designed for unicode characters. This approach ensures comprehensive and granular tokenization. The tokenizer produces a vocabulary of 50,257 tokens. For efficient implementation, we use the Hugging Face integration. Each token is mapped to an embedding vector using a lookup table, where each unique token in the vocabulary is associated with a distinct vector. Notably, this lookup table is shared with the one employed for discrete values in sequential decision-making tasks.

When a sample includes both images and text, the embeddings are arranged so that the image embeddings precede the text embeddings. This specific order is essential for image captioning task because of the causal masking applied by the model's internal transformer. The concatenated image-text embeddings form a unified representation for subsequent processing steps.

Transformer architecture

The JAT model is based on a transformer architecture using EleutherAI's implementation of GPT-Neo (Black et al., 2022). It takes as input the embedding sequence whose

computation was described in the previous section. The model uses a dual attention mechanism whose design is inspired by the Longformer (Beltagy et al., 2020): global attention with a window size of 512 tokens for full context understanding, and local attention with a fixed window of 256 tokens. The transformer's feed-forward components consist of 12 layers and 12 heads with an intermediate dimensionality of 8192 and a hidden size of 768. They are designed to be causal, meaning a causal mask is applied during training and inference.

Output processing and loss

The internal causal transformer outputs a sequence of embeddings, each encoding the basis for predicting subsequent elements in different data modalities. As we predict multiple modalities within a single sequence, we use the appropriate decoders and corresponding loss functions for each modality. When an embedding encodes an image, we use a transposed convolutional neural network (Zeiler et al., 2010) for prediction. When an embedding represents a continuous vector, we use a continuous linear layer for prediction. For both image and continuous vector prediction, we compute the loss using Mean Square Error (MSE). When an embedding represents a discrete value, we assign scores to each discrete candidate using a linear projection layer and compute the loss using cross-entropy. Notably, we use the same projection layer for text tokens and discrete sequential values (like action for Atari and BabyAI). To compute the overall loss of the sequence, we average the individual losses computed for each element. For sequential decision-making task, we apply a weighting between the loss related to observations and the loss related to actions. We show in Section G.4.3 that predicting the observations does help learning, and thereby solves one of the common open questions of (Chen et al., 2021) and (Reed et al., 2022).

G.3.2 Datasets

In this work, we have collected a wide range of datasets, classified into two main groups: sequential decision-making datasets and textual datasets. The former includes a series of interaction sequences, each consisting of observations, actions, and subsequent rewards, generated by so-called expert agents, details of which are given in the Appendix G.7. The latter includes large corpora of textual data and image-text pairs. In order to promote the emerging field of general-purpose AI models, we have made these datasets, together with the expert agents and the full set of code required to generate them, available to the public as open resources in our Hugging Face repository, accessible at the following URL `https://huggingface.co/jat-project`. To the best of our knowledge, this compilation is unprecedented in terms of the variety of tasks and the volume of data, representing a valuable new contribution to the field.

G.3.3 Training

Overall training procedure

The model was trained for 250,000 steps. We distributed the training across 8 GPUs NVIDIA V100 using the Trainer from the Hugging Face Transformers library (Wolf et al., 2020) in conjunction with Accelerate. This training lasted approximately 9 days. For practical reasons, each batch is made up of data from a single dataset. We use a constant batch size of 20 and accumulate over 2 steps, resulting in an effective batch size of 320. We use the AdamW optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate starts at $5 \cdot 10^{-5}$ and linearly decays to zero throughout the training process.

Task-specific weight adjustments

Each task presents a unique training challenge. To allow for balanced learning of all tasks, we introduced custom weight modifications. The choice of these weights is made heuristically. The learning would surely benefit from a more precise and systematic method for choosing these weights.

Sample weight Some tasks required more updates for effective convergence. To allow proportionate progress of all tasks during learning, these tasks are sampled more frequently. Specifically, Oscar, Conceptual-Captions, and Wikipedia were assigned a sample weight of 10.0, while others have a sample weight of 1.0.

Loss weight Some control tasks require increased accuracy of actions. To allow for more strongly penalizing the error for these tasks, we assigned loss weights. In MuJoCo tasks, the loss weight is typically set at 10.0, except for the Pendulum task (20.0) and the Double Pendulum task (50.0). In Meta-World tasks, a uniform loss weight of 50.0 is used.

G.4 Experiments and results

In this section, we discuss the results of our experiments. First, we provide a brief overview of the model's performance on text-centric tasks. We then present the results of the sequential decision-making tasks, showing the different levels of mastery across the different domains within our study. Finally, we provide a comprehensive analysis highlighting the benefits of incorporating the prediction of the next observation as an auxiliary task during the learning process.

G.4.1 Text-centric tasks

We present a showcase of JAT's capabilities, with a particular focus on text completion and image captioning. It's important to note that JAT is much smaller and has a much lower training budget than the specialized models for these tasks. Therefore, instead of comparing it to these expert systems, we want to demonstrate its intrinsic capabilities. Figure G.2 shows a selection of captioning results to illustrate how the model interprets and describes visual data. Additional examples are given in Appendix G.8. Figure G.3, meanwhile, shows a series of text-based interactions that provide insight into its ability to complete text prompts. These examples were chosen to highlight the model's basic

the flag was removed from the building after the fire broke out

: and beverage type at the beach. here are some of the most beautiful things i have ever seen.

: the food is good for the body! :) - a-baked chicken. person. p̃hoto by author

Figure G.2: JAT image captioning examples. The theme is usually correct, although the relevance is sometimes limited.

capabilities in these areas, providing a realistic view of its current state of development and potential for future enhancements. In addition, we provide a demos[1] for direct interaction and experimentation, allowing users to experience its functionalities.

Input	Model completion
The weather today is	a great time to the city of New York City. The city is a great place to stay in.
In the future, cars will	be able to drive cars to the market. The new car will be built in the new market for the new car.
My favorite book is	a book by the author of the book.

Figure G.3: JAT text completion examples. The syntax is generally correct, the completion is on-topic, although the generated text may be wrong.

G.4.2 Sequential decision-making tasks

We save checkpoints regularly during training. We evaluate each checkpoint on all the tasks on which it has been trained. Unlike Gato, the evaluation does not require any data to be used as a prompt. We show empirically in Appendix G.9 that despite the absence of a prompt, and even in the worst case of our study, the agent still manages to identify the requested task. For each task, we collect 10 evaluation episodes and normalize by the average expert score of the dataset for this task. For the final checkpoint, we use 100 evaluation episodes. We then aggregate the results by domain. Figure G.4 shows the evolution of the aggregate score for each domain during learning, and Figure G.5 focuses on Atari, showing the human normalized score for each environment. The final results are presented in detail in Appendix G.6.

The final agent achieves an average expert normalized interquartile mean (IQM) of 65.8%, demonstrating the network's ability to effectively mimic expert agents across a

[1] https://huggingface.co/spaces/jat-project/text-completion

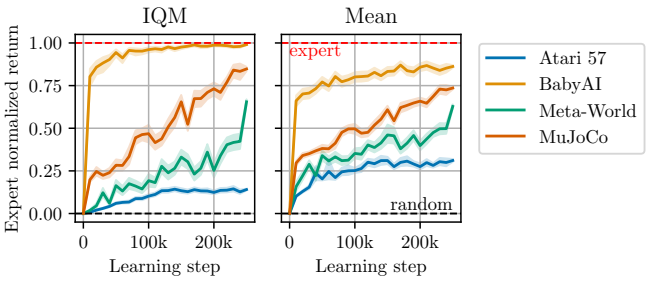


Figure G.4: Aggregated expert normalized scores with 95% Confidence Intervals (CIs) for each RL domain as a function of learning step.

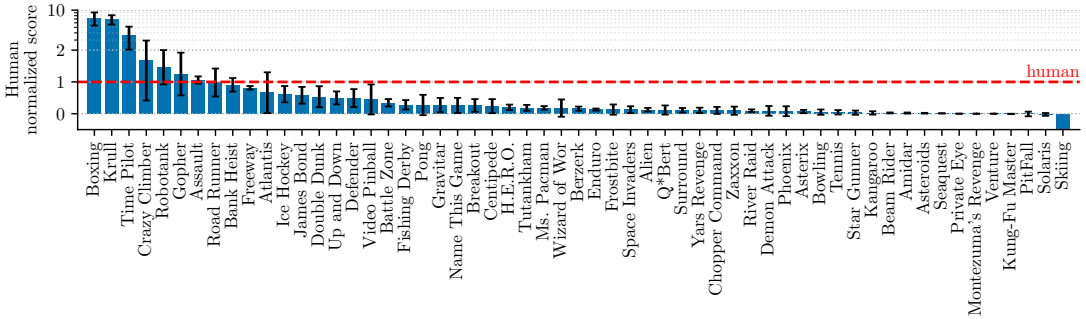


Figure G.5: Human normalized scores for the JAT agent on the Atari 57 benchmark.

wide range of tasks. The agent achieves 14.1% of the expert's score on the Atari 57 benchmark, corresponding to 37.6% of human performance, and exceeding the average human level in 21 games. For the BabyAI benchmark, JAT achieves a normalized score of 99.0%. This score falls below 50% for only one task, namely Move Two Across S8N9. For this benchmark, however, there is no guarantee that the expert score can be achieved, since the bot used for dataset collection has access to the full state of the environment, while the interacting agents only have access to a partial observation. Finally, in the MuJoCo and Meta-World, JAT records scores of 84.8% and 65.5%, respectively. Although JAT reaches expert level for a fair number of Meta-World tasks, we note that some, such as basketball, have not been learned at all. Insofar as the action and observation spaces are identical for all tasks in this benchmark, these failures may be due to task indeterminacy, which we explore in more detail in Appendix G.9. Future research will have to confirm this hypothesis. We also note that some domains are mastered more quickly than others; in particular, BabyAI achieves a score of 90% after only 30,000 learning steps. We hypothesize that the high semantic similarity of the tasks enables a strong positive transfer, without, however, providing any proof of this. The Appendix G.6 presents the final results in detail.

Although the results achieved are commendable, for a fair comparison, we limit our benchmarking to Gato only, as it is the only truly comparable baseline. Reed et al. (2022) present results only for the 1.18 billion parameter version of Gato, which is 6 times larger than JAT. Its results are normalized to expert performance. Since we don't have access to the normalization parameters, we estimated scores for the random agents, which may

not be exactly the same as those used by Reed et al. (2022), and used our expert scores for normalization, even though they obviously do not match those used by Reed et al. (2022). Therefore, comparisons of these normalized scores should be interpreted with great caution. On the Atari benchmark, JAT achieves an average normalized score of 31.1% outperforming Gato, which reports a score of 30.9%. For BabyAI, JAT achieves an average normalized score of 86.2%, close to the Gato score of 93.2%. Our study, however, is made with 39 tasks versus the 46 used in Gato's training, with the specific seven additional tasks in their study remaining unidentified. Since our evaluation includes all of the hardest tasks mentioned in their study, the seven missing tasks are likely to be easier, suggesting a harder test scenario in our study. For Meta-World, JAT achieves an average normalized score of 62.8%, which is below the 87.0% reported for Gato. On the MuJoCo benchmark, JAT achieves an average normalized score of 73.6%. While Gato's training doesn't use MuJoCo, it's worth noting that they use the DMC benchmark, which shares some similarities. For reference, on the DMC benchmark (Tassa et al., 2018), Gato achieves an average score of 63.6%.

G.4.3 Predicting the observations does help

The model's main task is to predict the actions that maximize the sum of future rewards. Its ability to predict future observations is therefore not the main concern. However, can this ability contribute to better prediction of actions or accelerate the learning process? Two contrasting hypotheses emerge: firstly, learning to predict observations could serve as an auxiliary objective, directing the learning process towards a deeper understanding of the environment, which could lead to improved and faster learning. Conversely, this prediction learning could serve as a distracting objective: instead of excelling in action prediction, the model might only achieve moderate performance in both action and observation prediction. This could slow down the learning process, resulting in a lower overall performance score. Reed et al. (2022) choose not to predict the observation, but does not study the influence of this prediction on learning.

To answer this question, we use a loss function that combines observation loss ($\mathcal{L}_{\text{obs}}$) and action loss ($\mathcal{L}_{\text{act}}$), balanced by a weighting parameter κ. The function is defined as:

$$\mathcal{L} = \kappa \cdot \mathcal{L}_{\text{obs}} + (1 - \kappa) \cdot \mathcal{L}_{\text{act}} \tag{G.1}$$

We select a range of values for κ and train the model on a subset of 6 dataset tasks from different domains (Freeway, Pong, ButtonPressWall, WindowClose, Ant and DoubleInvertedPendulum). Figure G.6 compares the results at the end of training for the different values of κ.

In our study of the κ coefficient and its impact on learning, we find an interesting balance. When set to the highest value in our range ($\kappa = 0.5$), the learning process seems to be somewhat hindered by the additional objective. On the other hand, at lower κ values, this added task of predicting observations doesn't significantly impact learning, leading to scores that are similar to the base score of $94.5 \pm 1.1\%$, which we get when predicting observations isn't part of the objective. The sweet spot appears to be around $\kappa = 0.005$. Learning to predict observations doesn't distract but actually improves the agent's learning efficiency, achieving a near-optimal score of $99.1 \pm 0.4\%$.

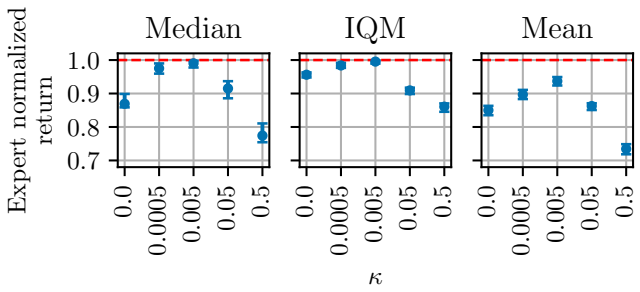


Figure G.6: Aggregate measures with 95% CIs for the study on the influence of observation prediction learning for selected tasks. The results presented cover the selected range of κ values and are based on 100 evaluations per task. Optimal κ selection can significantly improve agent performance.

This finding highlights that adding observation prediction into the learning process is beneficial, provided it's balanced correctly.

G.5 Conclusion

In this study, we introduced JAT, a novel multimodal framework for general RL agents. JAT features the ability to handle diverse tasks of varying complexity using a single set of parameters. Its innovations include a new transformer-based structure that efficiently addresses sequential decision-making, CV, and NLP tasks. We also show that joint learning of observation prediction significantly improves performance in sequential decision-making tasks. We've open-sourced our training dataset, which includes a wide range of sequential decision-making data as well as extensive language and visual data. We believe that JAT represents an important and valuable step towards general-purpose RL models.

This study reveals several avenues for improvement. A primary challenge is the joint learning of tasks characterized by high heterogeneity. Our dataset features variations in size, task complexity, and accuracy requirements for optimal performance. Our current approach, which uses basic sample and loss weighting, partially addresses this challenge. A refinement of task sampling could potentially account for task difficulty or impact on the model's improvement. For this, Automatic Curriculum Learning methods, discussed in Chapter 5 as well as Appendix H, might prove useful.

Another important challenge lies in the use of imitation learning. First, improving the quality of expert data presents a clear opportunity for progress. For instance, in the Asterix task, our model's expert score (3699.6) remains far below that of specialized agents such as R2D2 (999,153.3) (Kapturowski et al., 2019). Leveraging the strongest RL agent available for each individual task could significantly raise the quality of our dataset, and in turn, improve the performance of the distilled generalist agent. Furthermore, while our current method employs basic BC, incorporating more advanced imitation learning techniques could yield further gains. More importantly, as discussed throughout this manuscript, active learning through interaction with an environment is key—whereas imitation learning comes with inherent limitations. Across multiple contributions, we

emphasized how RL and direct interaction drive the acquisition of functional competence in language processing. Although JAT focuses solely on pre-training, a natural next step would be to fine-tune the model using RL via online interactions across multiple domains.

G.6 Full results

This appendix contains a detailed view of the results of the trained JAT agent. The score of the random agent for Atari games is sourced from (Mnih et al., 2015). In other domains, this score is approximated by averaging the returns from 1,000 episodes, where the agent selects actions uniformly across its action space. The expert scores represent the average return in the dataset for the task. Meanwhile, the raw score is the average return achieved by the trained agent, based on 100 evaluation episodes. Both these scores, along with the trained agent, are accessible as open-source[2]. The normalized score is derived by comparing the agent's return to the expert's, calculated using the formula: $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$. It's important to note that in instances where the *expert*, inaccurately named, does not fully master the task and thus scores similarly or lower than the random agent, the normalized score must be interpreted cautiously. Specifically, if this score falls below that of the random agent, as in the case of Bowling, normalization is not applied. The results for Atari are presented in Table G.1 and Figure G.7, for BabyAI in Table G.2 and Figure G.8, for Meta-World in Table G.3 and Figure G.9, and for MuJoCo in Table G.4 and Figure G.10.

[2] https://huggingface.co/jat-project/jat

Table G.1: Comparison of performance scores across tasks on Atari 57. The table presents the episodic return (score) achieved by a random agent (from (Mnih et al., 2015)), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT (raw)	JAT (normalized)
Alien	227.8	16912.5 ± 7087.4	1474.9 ± 588.7	0.07 ± 0.04
Amidar	5.8	2164.7 ± 1229.5	104.9 ± 103.5	0.05 ± 0.05
Assault	222.4	15699.1 ± 9572.1	1650.1 ± 821.0	0.09 ± 0.05
Asterix	210.0	3699.6 ± 2421.3	800.0 ± 584.9	0.17 ± 0.17
Asteroids	719.0	177011.1 ± 35334.2	1385.3 ± 507.5	0.00 ± 0.00
Atlantis	12850.0	320679.6 ± 418247.4	66980.0 ± 158449.7	0.18 ± 0.51
Bank Heist	14.2	1322.4 ± 60.8	948.3 ± 199.9	0.71 ± 0.15
Battle Zone	236.0	295592.6 ± 161961.0	17420.0 ± 6071.5	0.06 ± 0.02
Beam Rider	363.9	29589.3 ± 16133.0	797.3 ± 328.3	0.01 ± 0.01
Berzerk	123.7	57085.3 ± 13104.5	687.3 ± 331.9	0.01 ± 0.01
Bowling	23.1	20.4 ± 7.3	22.4 ± 5.6	N/A
Boxing	0.1	98.0 ± 3.8	90.1 ± 23.0	0.92 ± 0.24
Breakout	1.7	703.0 ± 203.6	8.8 ± 5.6	0.01 ± 0.01
Centipede	2090.9	11624.3 ± 4918.3	5589.9 ± 2567.3	0.37 ± 0.27
Chopper Command	811.0	90990.6 ± 270876.9	2417.0 ± 1489.9	0.02 ± 0.02
Crazy Climber	10780.5	179296.9 ± 39862.1	97639.0 ± 26184.7	0.52 ± 0.16
Defender	2874.5	351958.3 ± 40466.8	39323.5 ± 15203.0	0.10 ± 0.04
Demon Attack	152.1	92195.2 ± 26174.8	815.3 ± 989.7	0.01 ± 0.01
Double Dunk	-18.6	20.9 ± 3.6	14.4 ± 10.0	0.84 ± 0.25
Enduro	0.0	2292.2 ± 147.5	108.5 ± 42.7	0.05 ± 0.02
Fishing Derby	-91.7	7.2 ± 25.1	-30.4 ± 24.4	0.62 ± 0.25
Freeway	0.0	33.9 ± 0.3	27.5 ± 1.6	0.81 ± 0.05
Frostbite	65.2	13196.1 ± 4341.0	2769.6 ± 1445.6	0.21 ± 0.11
Gopher	257.6	81676.2 ± 46329.5	5340.6 ± 2547.1	0.06 ± 0.03
Gravitar	173.0	3986.6 ± 1729.0	1269.5 ± 903.0	0.29 ± 0.24
H.E.R.O.	1027.0	44677.4 ± 1754.4	11709.6 ± 3233.5	0.24 ± 0.07
Ice Hockey	-11.2	25.2 ± 5.8	7.5 ± 5.6	0.51 ± 0.15
James Bond	29.0	27786.9 ± 33819.2	327.5 ± 123.2	0.01 ± 0.00
Kangaroo	52.0	574.0 ± 636.9	378.0 ± 344.0	0.62 ± 0.66
Krull	1598.0	11439.8 ± 1218.3	10720.5 ± 1284.1	0.93 ± 0.13
Kung-Fu Master	258.5	32392.8 ± 10006.6	288.0 ± 255.1	0.00 ± 0.01
Montezuma's Revenge	0.0	393.5 ± 50.4	0.0 ± 0.0	0.00 ± 0.00
Ms. Pacman	307.3	6896.1 ± 2032.0	1573.1 ± 484.0	0.19 ± 0.07
Name This Game	2292.3	22991.2 ± 2473.1	7523.3 ± 2471.4	0.25 ± 0.12
Phoenix	761.5	424583.2 ± 97649.2	2197.9 ± 1795.4	0.00 ± 0.00
PitFall	-229.4	-1.4 ± 4.5	-6.7 ± 19.0	0.98 ± 0.08
Pong	-20.7	21.0 ± 0.2	13.7 ± 13.3	0.82 ± 0.32
Private Eye	24.9	100.0 ± 0.0	44.0 ± 49.6	0.25 ± 0.66
Q*Bert	163.9	42971.4 ± 85070.7	1951.5 ± 2577.2	0.04 ± 0.06
River Raid	1338.5	14800.9 ± 7924.6	3758.5 ± 1536.7	0.18 ± 0.11
Road Runner	11.5	77942.8 ± 6088.6	6407.0 ± 4847.4	0.08 ± 0.06
Robotank	2.2	80.5 ± 13.3	11.3 ± 5.5	0.12 ± 0.07
Seaquest	68.4	2597.3 ± 386.1	804.0 ± 403.3	0.29 ± 0.16
Skiing	-17098.0	-10738.1 ± 111.1	-16231.5 ± 6060.5	0.14 ± 0.95
Solaris	1236.3	1353.7 ± 517.0	1286.6 ± 446.7	0.43 ± 3.81
Space Invaders	148.0	29425.3 ± 23623.9	325.4 ± 163.4	0.01 ± 0.01
Star Gunner	664.0	360588.6 ± 49207.7	4379.0 ± 3027.2	0.01 ± 0.01
Surround	-10.0	9.4 ± 0.8	2.7 ± 4.7	0.65 ± 0.24
Tennis	-23.8	11.1 ± 7.6	-13.5 ± 3.8	0.30 ± 0.11
Time Pilot	3568.0	69583.3 ± 29838.7	13028.0 ± 5222.6	0.14 ± 0.08
Tutankham	11.4	291.2 ± 30.4	85.7 ± 61.8	0.27 ± 0.22
Up and Down	533.4	429418.3 ± 7187.4	17768.7 ± 10322.0	0.04 ± 0.02
Venture	0.0	0.0 ± 0.0	0.0 ± 0.0	N/A
Video Pinball	0.0	441507.9 ± 283264.6	11917.4 ± 8204.3	0.03 ± 0.02
Wizard of Wor	563.5	49333.3 ± 16157.1	2544.0 ± 2902.4	0.04 ± 0.06
Yars Revenge	3092.9	270262.9 ± 161816.0	12532.7 ± 8062.8	0.04 ± 0.03
Zaxxon	32.5	73097.2 ± 14825.8	6902.0 ± 3206.1	0.09 ± 0.04

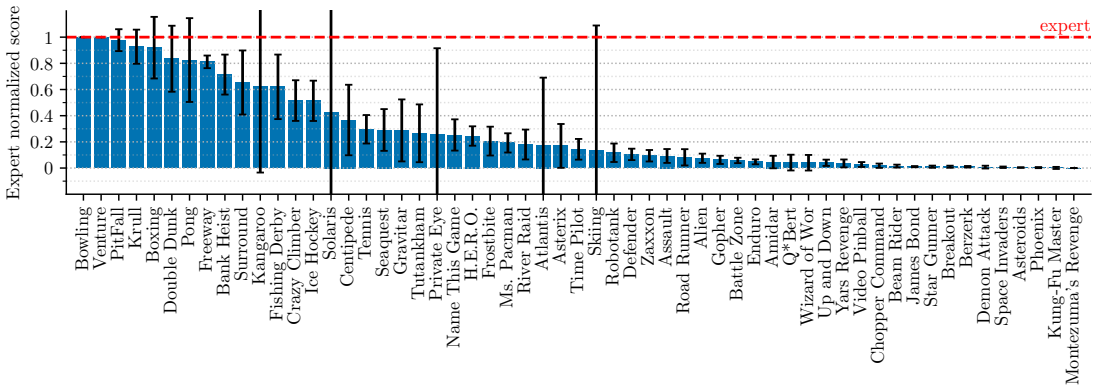


Figure G.7: Expert normalized episodic return for the JAT agent on the Atari 57 benchmark.

Table G.2: Comparison of performance scores across tasks on BabyAI. The table presents the episodic return (score) achieved by a random agent (averaged over 1,000 episodes), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT (raw)	JAT (normalized)
Action Obj Door	0.37 ± 0.39	0.99 ± 0.01	0.94 ± 0.14	0.93 ± 0.23
Blocked Unlock Pickup	0.00 ± 0.02	0.95 ± 0.01	0.95 ± 0.01	1.00 ± 0.01
Boss Level	0.06 ± 0.21	0.94 ± 0.05	0.52 ± 0.43	0.53 ± 0.49
Boss Level No Unlock	0.06 ± 0.19	0.94 ± 0.05	0.48 ± 0.43	0.48 ± 0.48
Find Obj S5	0.08 ± 0.23	0.95 ± 0.04	0.95 ± 0.04	1.01 ± 0.05
Go To	0.13 ± 0.29	0.92 ± 0.07	0.83 ± 0.27	0.89 ± 0.34
Go To Door	0.45 ± 0.38	0.99 ± 0.00	0.99 ± 0.02	0.99 ± 0.04
Go To Imp Unlock	0.07 ± 0.22	0.83 ± 0.13	0.60 ± 0.41	0.70 ± 0.54
Go To Local	0.16 ± 0.30	0.93 ± 0.04	0.88 ± 0.14	0.94 ± 0.19
Go To Obj	0.13 ± 0.27	0.93 ± 0.03	0.93 ± 0.03	1.00 ± 0.04
Go To Obj Door	0.53 ± 0.39	0.99 ± 0.01	0.96 ± 0.10	0.94 ± 0.21
Go To Red Ball	0.17 ± 0.30	0.93 ± 0.04	0.92 ± 0.05	0.99 ± 0.07
Go To Red Ball Grey	0.12 ± 0.27	0.92 ± 0.05	0.91 ± 0.07	0.99 ± 0.08
Go To Red Ball No Dists	0.14 ± 0.28	0.93 ± 0.03	0.93 ± 0.03	1.00 ± 0.04
Go To Red Blue Ball	0.12 ± 0.27	0.92 ± 0.05	0.88 ± 0.11	0.95 ± 0.14
Go To Seq	0.08 ± 0.23	0.94 ± 0.05	0.72 ± 0.34	0.74 ± 0.40
Key Corridor	0.00 ± 0.00	0.91 ± 0.01	0.86 ± 0.16	0.94 ± 0.18
Mini Boss Level	0.07 ± 0.21	0.89 ± 0.10	0.61 ± 0.39	0.65 ± 0.47
Move Two Across S8N9	0.00 ± 0.00	0.96 ± 0.01	0.02 ± 0.13	0.02 ± 0.13
One Room S8	0.08 ± 0.21	0.92 ± 0.03	0.92 ± 0.04	1.00 ± 0.04
Open	0.10 ± 0.24	0.95 ± 0.05	0.94 ± 0.11	0.98 ± 0.13
Open Door	0.23 ± 0.34	0.99 ± 0.00	0.99 ± 0.00	1.00 ± 0.01
Open Doors Order N4	0.16 ± 0.30	0.99 ± 0.01	0.95 ± 0.17	0.95 ± 0.20
Open Red Door	0.08 ± 0.21	0.92 ± 0.03	0.91 ± 0.03	1.00 ± 0.04
Open Two Doors	0.08 ± 0.20	0.98 ± 0.00	0.98 ± 0.00	1.00 ± 0.00
Pickup	0.08 ± 0.22	0.92 ± 0.07	0.76 ± 0.32	0.82 ± 0.38
Pickup Above	0.02 ± 0.09	0.91 ± 0.07	0.90 ± 0.08	0.99 ± 0.09
Pickup Dist	0.10 ± 0.24	0.86 ± 0.21	0.90 ± 0.07	1.05 ± 0.09
Pickup Loc	0.08 ± 0.23	0.91 ± 0.04	0.86 ± 0.16	0.94 ± 0.19
Put Next S7N4	0.00 ± 0.03	0.96 ± 0.01	0.85 ± 0.22	0.88 ± 0.23
Put Next Local	0.00 ± 0.05	0.92 ± 0.03	0.63 ± 0.36	0.69 ± 0.40
Synth	0.11 ± 0.26	0.93 ± 0.06	0.77 ± 0.33	0.80 ± 0.40
Synth Loc	0.13 ± 0.29	0.94 ± 0.06	0.79 ± 0.33	0.81 ± 0.40
Synth Seq	0.07 ± 0.20	0.95 ± 0.04	0.52 ± 0.44	0.52 ± 0.50
Unblock Pickup	0.08 ± 0.22	0.91 ± 0.08	0.74 ± 0.32	0.79 ± 0.39
Unlock	0.03 ± 0.15	0.87 ± 0.10	0.52 ± 0.42	0.58 ± 0.50
Unlock Local	0.01 ± 0.09	0.98 ± 0.01	0.98 ± 0.01	1.00 ± 0.01
Unlock Pickup	0.00 ± 0.00	0.75 ± 0.04	0.76 ± 0.04	1.01 ± 0.05
Unlock To Unlock	0.00 ± 0.00	0.96 ± 0.00	0.80 ± 0.35	0.83 ± 0.37

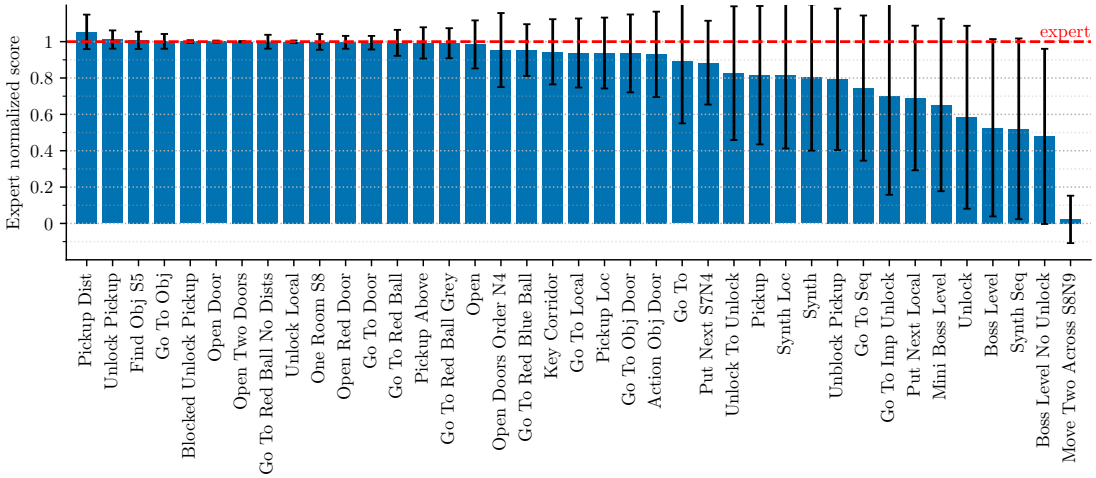


Figure G.8: Expert normalized episodic return for the JAT agent on the BabyAI benchmark.

Table G.3: Comparison of performance scores across tasks on Meta-World. The table presents the episodic return (score) achieved by a random agent (averaged over 1,000 episodes), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT (raw)	JAT (normalized)
Assembly	45.3 ± 4.1	246.0 ± 3.5	238.0 ± 34.6	0.96 ± 0.17
Basketball	2.8 ± 1.2	628.0 ± 2.0	1.6 ± 0.4	-0.00 ± 0.00
BinPicking	1.9 ± 0.4	425.6 ± 101.9	200.0 ± 222.1	0.47 ± 0.52
Box Close	76.4 ± 17.9	512.5 ± 107.8	462.6 ± 172.0	0.89 ± 0.39
Button Press	31.7 ± 5.2	643.1 ± 12.8	560.0 ± 182.6	0.86 ± 0.30
Button Press Topdown	29.0 ± 10.4	490.2 ± 27.2	266.2 ± 77.2	0.51 ± 0.17
Button Press Topdown Wall	29.0 ± 10.5	497.2 ± 31.4	275.8 ± 88.6	0.53 ± 0.19
Button Press Wall	9.0 ± 4.0	675.4 ± 15.0	638.3 ± 123.0	0.94 ± 0.18
Coffee Button	31.7 ± 6.4	731.1 ± 29.3	298.0 ± 285.6	0.38 ± 0.41
Coffee Pull	4.1 ± 0.4	259.9 ± 88.5	41.0 ± 69.8	0.14 ± 0.27
Coffee Push	4.2 ± 0.8	496.8 ± 118.2	153.1 ± 218.9	0.30 ± 0.44
Dial Turn	29.6 ± 16.7	793.6 ± 80.1	758.4 ± 120.4	0.95 ± 0.16
Disassemble	40.3 ± 7.5	42.8 ± 6.3	40.7 ± 9.9	0.17 ± 3.91
Door Close	5.3 ± 1.3	529.7 ± 27.2	524.3 ± 33.2	0.99 ± 0.06
Door Lock	112.3 ± 28.6	811.5 ± 34.1	696.3 ± 198.6	0.84 ± 0.28
Door Open	56.4 ± 11.2	581.9 ± 19.7	577.5 ± 53.7	0.99 ± 0.10
Door Unlock	94.2 ± 15.6	802.9 ± 17.1	768.3 ± 91.8	0.95 ± 0.13
Drawer Close	116.7 ± 253.1	867.9 ± 4.5	596.7 ± 223.4	0.64 ± 0.30
Drawer Open	126.8 ± 25.2	493.0 ± 2.5	485.9 ± 36.8	0.98 ± 0.10
Faucet Close	253.1 ± 22.9	753.9 ± 13.4	367.8 ± 91.1	0.23 ± 0.18
Faucet Open	244.1 ± 23.3	705.8 ± 7.1	566.1 ± 169.9	0.70 ± 0.37
Hammer	95.3 ± 9.0	693.2 ± 34.6	667.7 ± 89.4	0.96 ± 0.15
Hand Insert	2.8 ± 3.5	740.5 ± 36.7	688.1 ± 187.7	0.93 ± 0.25
Handle Press	80.4 ± 110.2	855.9 ± 72.7	735.0 ± 252.0	0.84 ± 0.32
Handle Press Side	57.0 ± 39.5	861.1 ± 20.0	64.5 ± 73.7	0.01 ± 0.09
Handle Pull	10.3 ± 13.5	669.4 ± 24.8	556.6 ± 161.7	0.83 ± 0.25
Handle Pull Side	2.1 ± 2.8	384.7 ± 102.9	195.1 ± 187.2	0.50 ± 0.49
Lever Pull	60.3 ± 15.8	612.0 ± 38.9	280.9 ± 234.8	0.40 ± 0.43
Peg Insert Side	1.7 ± 0.4	315.2 ± 140.1	254.3 ± 158.4	0.81 ± 0.51
Peg Unplug Side	4.7 ± 2.8	456.1 ± 81.7	80.6 ± 145.5	0.17 ± 0.32
Pick Out Of Hole	1.5 ± 0.2	219.6 ± 88.9	2.1 ± 0.1	0.00 ± 0.00
Pick Place	1.6 ± 1.0	419.1 ± 98.2	135.8 ± 200.1	0.32 ± 0.48
Pick Place Wall	0.0 ± 0.0	450.6 ± 64.1	43.7 ± 129.7	0.10 ± 0.29
Plate Slide	74.6 ± 13.8	527.0 ± 155.3	481.5 ± 190.2	0.90 ± 0.42
Plate Slide Back	33.5 ± 11.2	718.2 ± 87.4	196.9 ± 1.7	0.24 ± 0.00
Plate Slide Back Side	34.3 ± 11.5	729.6 ± 69.1	703.7 ± 117.3	0.96 ± 0.17
Plate Slide Side	22.6 ± 17.4	662.8 ± 102.8	122.6 ± 24.6	0.16 ± 0.04
Push	5.5 ± 2.4	750.6 ± 44.0	702.4 ± 157.6	0.94 ± 0.21
Push Back	1.2 ± 0.2	85.0 ± 107.1	82.2 ± 108.0	0.97 ± 1.29
Push Wall	6.1 ± 3.2	748.9 ± 10.6	158.8 ± 224.6	0.21 ± 0.30
Reach	149.7 ± 44.7	681.4 ± 133.7	332.2 ± 171.5	0.34 ± 0.32
Reach Wall	143.3 ± 36.6	746.1 ± 104.2	631.6 ± 224.0	0.81 ± 0.37
Shelf Place	0.0 ± 0.0	241.3 ± 24.6	92.1 ± 112.0	0.38 ± 0.46
Soccer	5.7 ± 4.6	375.2 ± 140.2	291.6 ± 161.8	0.77 ± 0.44
Stick Pull	2.6 ± 1.4	523.6 ± 18.9	480.1 ± 119.3	0.92 ± 0.23
Stick Push	2.8 ± 1.0	627.9 ± 10.2	303.2 ± 298.8	0.48 ± 0.48
Sweep	11.2 ± 7.3	494.8 ± 43.3	16.7 ± 18.7	0.01 ± 0.04
Sweep Into	12.5 ± 10.7	799.2 ± 19.1	793.3 ± 47.5	0.99 ± 0.06
Window Close	57.5 ± 7.1	591.3 ± 38.6	414.3 ± 207.4	0.67 ± 0.39
Window Open	43.4 ± 2.1	590.8 ± 57.1	577.3 ± 63.9	0.98 ± 0.12

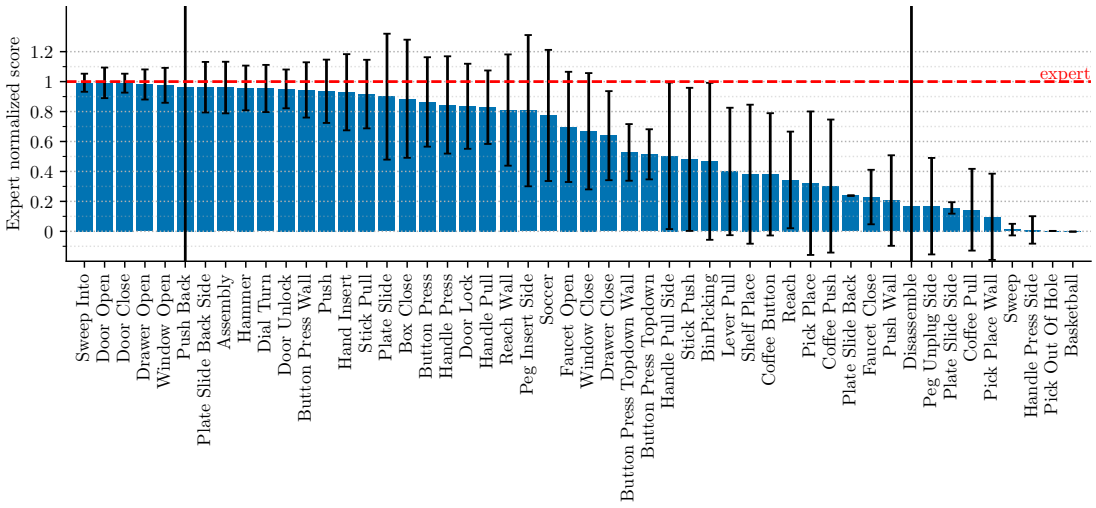


Figure G.9: Expert normalized episodic return for the JAT agent on the Meta-World benchmark.

Table G.4: Comparison of performance scores across tasks on MuJoCo. The table presents the episodic return (score) achieved by a random agent (averaged over 1,000 episodes), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT (raw)	JAT (normalized)
Ant	-59.9 ± 99.6	5846.4 ± 942.6	5110.5 ± 1720.8	0.88 ± 0.29
Inverted Double Pendulum	57.5 ± 17.5	9338.7 ± 352.6	8663.7 ± 1259.4	0.93 ± 0.14
Half Cheetah	-285.0 ± 79.8	7437.8 ± 173.3	6595.9 ± 244.4	0.89 ± 0.03
Hopper	18.4 ± 17.1	1858.7 ± 534.1	1409.0 ± 385.6	0.76 ± 0.21
Humanoid	122.0 ± 35.3	6281.0 ± 1795.8	712.6 ± 120.6	0.10 ± 0.02
Inverted Pendulum	6.1 ± 3.5	475.4 ± 179.0	117.4 ± 22.0	0.24 ± 0.05
Pusher	-149.7 ± 7.4	-25.2 ± 6.7	-25.0 ± 6.3	1.00 ± 0.05
Reacher	-43.0 ± 3.9	-5.7 ± 2.5	-5.9 ± 2.4	0.99 ± 0.06
Humanoid Standup	33135.8 ± 2481.9	273574.2 ± 85253.3	116736.7 ± 22765.5	0.35 ± 0.09
Swimmer	0.8 ± 10.7	92.2 ± 4.4	94.0 ± 4.1	1.02 ± 0.04
Walker 2d	2.7 ± 6.1	4631.2 ± 1059.0	4381.3 ± 851.1	0.95 ± 0.18

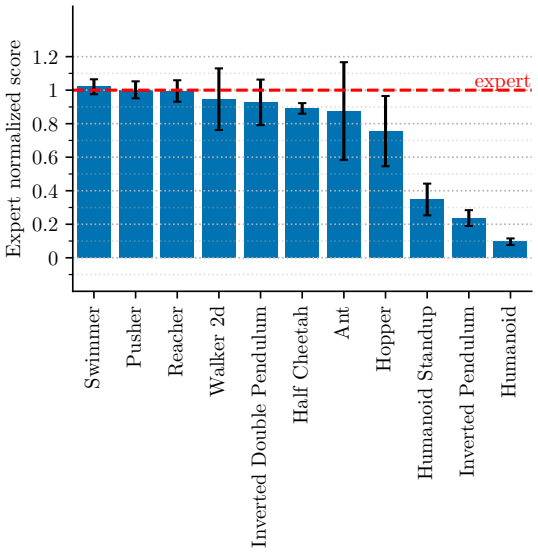


Figure G.10: Expert normalized episodic return for the JAT agent on the MuJoCo benchmark.

G.7 JAT dataset in depth

G.7.1 Sequential decision-making datasets

For each decision-making environment, we collect a set of interactions using expert agents. Detailed scores are available in the Appendix G.6.

Atari – We use the 57 games from the Arcade Learning Environment (Bellemare et al., 2013) as a benchmark in our research, amassing roughly 500,000 interactions per game. Episode lengths varied significantly depending on the specific game. For each game, we trained a dedicated agent for 2 billion steps using the asynchronous implementation of Proximal Policy Optimization (Schulman et al., 2017) from Sample Factory (Petrenko et al., 2020). The expert agents achieve above human performance on 43 tasks[3].

BabyAI – BabyAI stands out in our study due to its unique characteristic of being partially observable and its dual-modality observations (Chevalier-Boisvert et al., 2019, 2023). Using the bot provided with the BabyAI paper (Chevalier-Boisvert et al., 2019), we gathered 100,000 episodes for 39 of its available settings. Each interaction consists of a text observation (mission), a discrete observation (7×7 symbolic representation of the agent's field of view), an action, and a reward.

Meta-World – Meta-World's MT50 benchmark provides a set of 50 diverse and challenging robot manipulation tasks (Yu et al., 2020). Similar to the methodology used for Atari, we trained one agent per task using the asynchronous PPO (Schulman et al., 2017) implementation of (Petrenko et al., 2020). The trained agents solved most of the tasks, except for Assembly and Disassemble, where they failed to reach the expected performance. We limit the number of timesteps per episode to 100, which proved to be sufficient for solving the tasks. Without this limit, much of the subsequent dataset would consist of the stabilization phases of the agents after goal attainment, reducing its relevance. We then used the trained agents to generate 10,000 episodes per environment.

MuJoCo – We included the MuJoCo locomotion benchmark suite (Todorov et al., 2012; Brockman et al., 2016) comprising 11 continuous control tasks into our study due to its diverse challenges in domain complexity and task difficulty, and its wide recognition in the research literature. Following our methodologies for Atari and Meta-World, we individually trained agents for each task using asynchronous PPO (Schulman et al., 2017) from Sample Factory (Petrenko et al., 2020). These agents successfully solved all tasks, achieving scores that meet or exceed the current highest standards. Subsequently, we employed these agents to generate 10,000 episodes per environment.

Each sample in this dataset is an episode. This episode consists of a list of observations, actions, and rewards, the nature and size of which depend on the task. In Figure G.11, we represent for each Atari game the average return of the episodes of the dataset normalized

[3]The expert score is below the human score for Asterix, Bowling, Centipede, Fishing Derby, Kangaroo, Montezuma's Revenge, Ms. Pacman, Pitfall, Private Eye, River Raid, Seaquest, Skiing, Solaris, Venture

by the human score from (Mnih et al., 2015). Notably, for 43 games the average score is higher than the human score, and for 31 games the average score is more than twice the human score. It should be noted, however, that for 7 games (Bowling, Montezuma's Revenge, PitFall, Private Eye, Seaquest, Solaris and Venture) the average score is less than 10% of the human score.

We also plot the distribution of returns for each task, which provides a more detailed picture than a simple average. Figure G.12 shows this distribution for Atari, Figure G.13 for BabyAI, Figure G.14 for Meta-World, and Figure G.15 for MuJoCo.

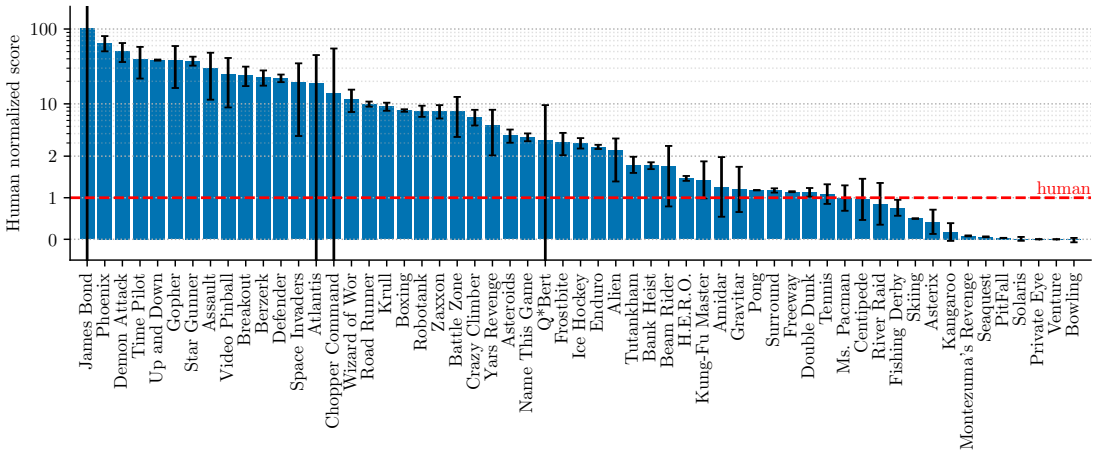


Figure G.11: Human normalized dataset scores for Atari.

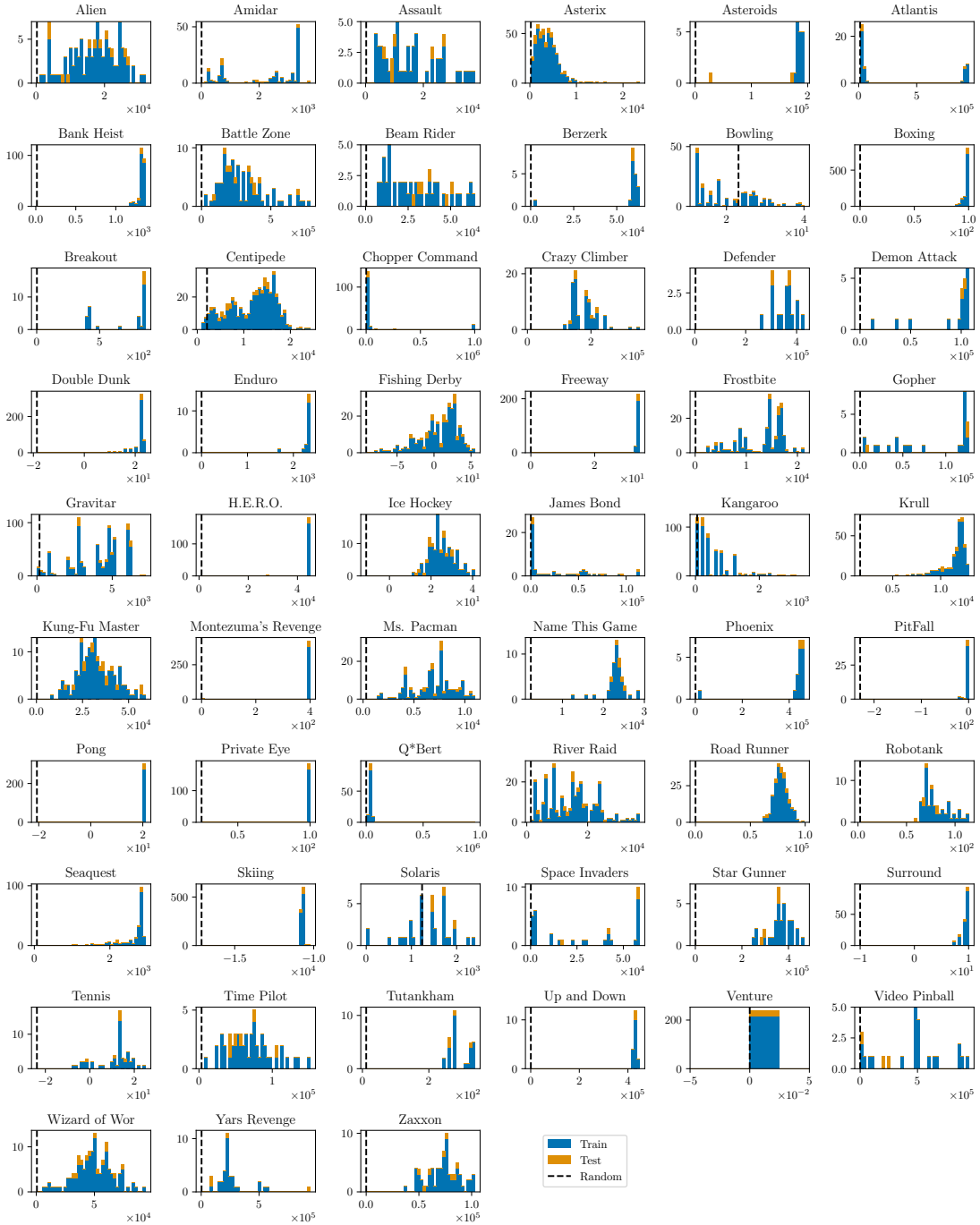


Figure G.12: Atari dataset return distribution.

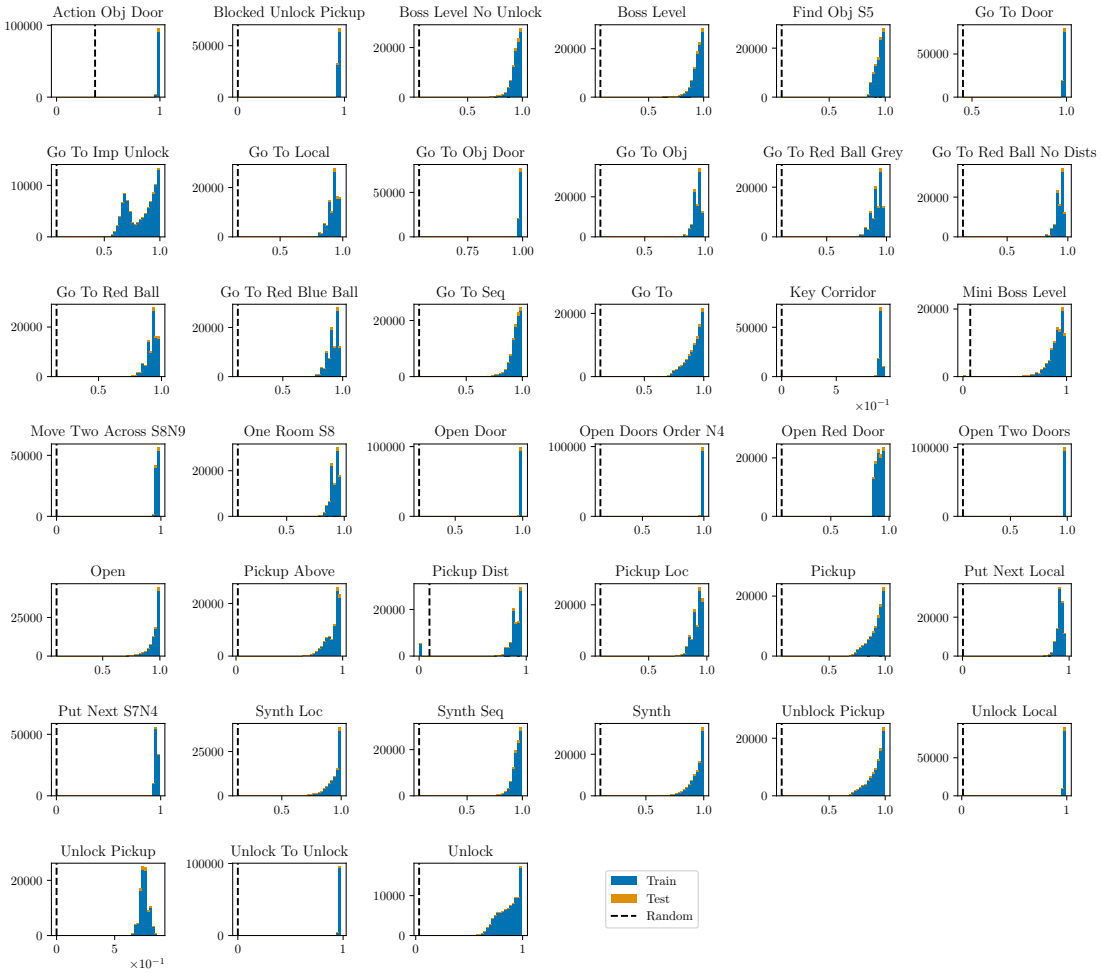


Figure G.13: BabyAI dataset return distribution.

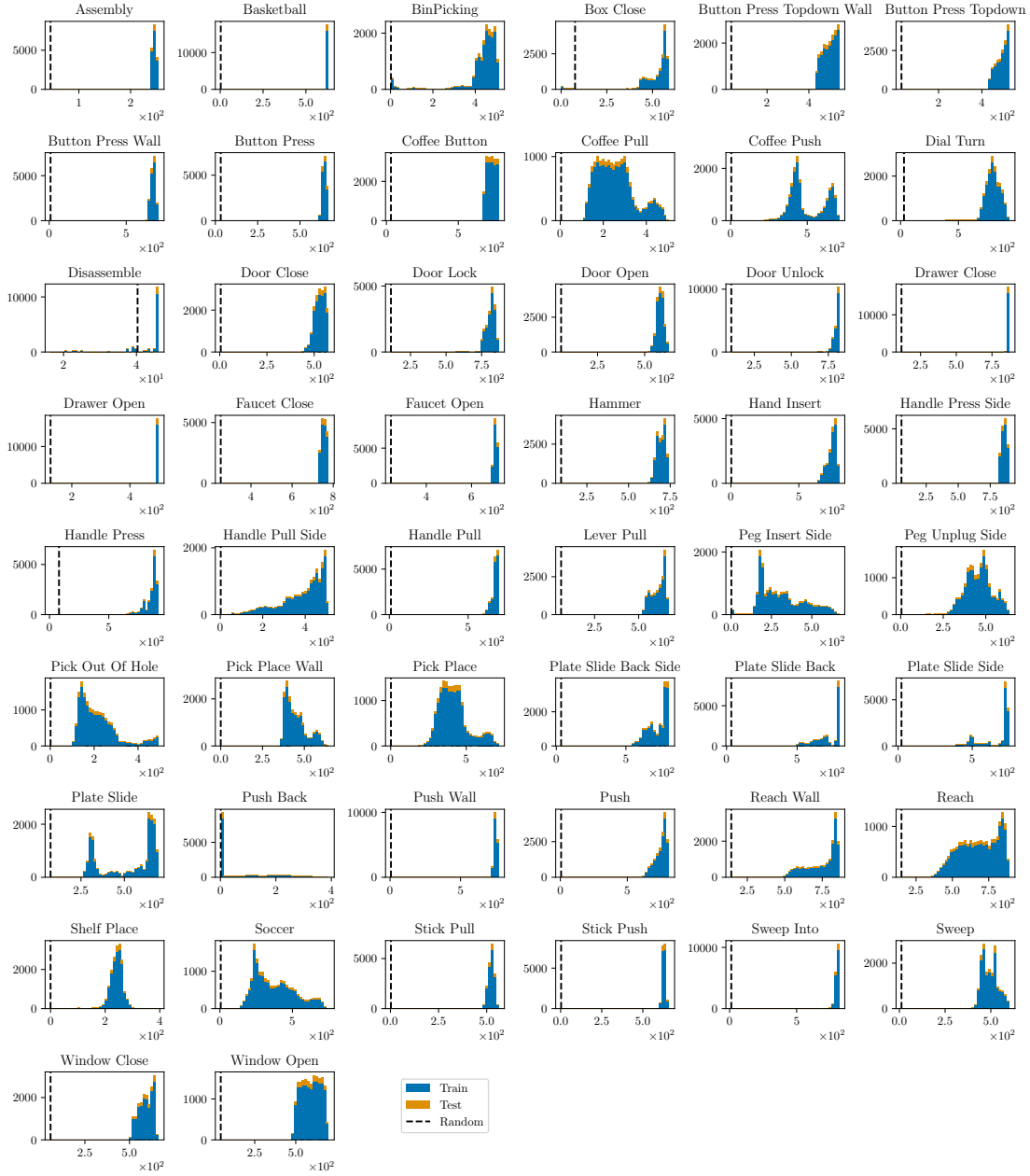


Figure G.14: Meta-World dataset return distribution.

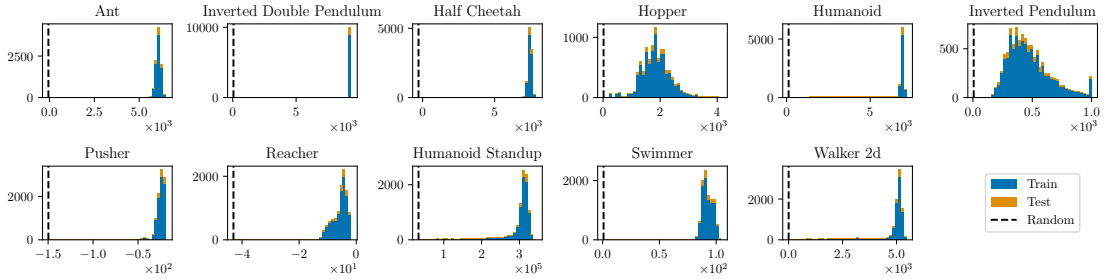


Figure G.15: MuJoCo dataset return distribution.

Text-Centric Datasets

Oscar – Common Crawl-based text documents have been widely used in the past to create datasets for Language Modeling (Radford et al., 2019; Brown et al., 2020; Raffel et al., 2020). We chose to leverage the unshuffled deduplicated English subset of the OSCAR[4] corpus (Ortiz Suárez et al., 2020) for our language modeling objective. As such crawled internet data needs to be cleaned before using it for training Language Models (e.g., deduplication, filtering out machine-generating content), we reused both the cleaning and deduplication pipeline from the ROOTS corpus (Laurençon et al., 2022). The initial dataset was shuffled, split into a training (95%) and test (5%) set, and evenly split into 30 shards on which the cleaning and deduplication pipelines were applied to reduce the memory needs. Shards were then concatenated back together, leading to a final dataset of 245 million documents (compared to 304 million documents in the initial dataset).

Conceptual-Captions – We include the Conceptual-Captions dataset (Sharma et al., 2018), as it is a key resource for image captioning and visual understanding tasks. It contains over 2.6 million training examples and over 12,000 test examples, with a wide range of web-sourced images, each paired with a descriptive caption.

OK-VQA – We include the OK-VQA dataset (Marino et al., 2019) because it is an essential resource for visual question answering tasks that focus on the intersection of visual perception and knowledge-based reasoning. With over 14,000 samples, it contains a wide range of images, each associated with questions that require not only visual understanding, but also external knowledge for an accurate answer.

Wikipedia – The Wikipedia dataset, built from the Wikipedia dump[5], contains over 6 million English language samples as of March 1, 2022. It offers a wide range of topics and a wealth of information. By using this dataset, we aim to improve the language processing capabilities of our model and provide access to extensive reservoir of encyclopedic knowledge.

[4]Its original version from 2019: https://huggingface.co/datasets/oscar
[5]https://dumps.wikimedia.org/

G.8 Image captioning additional examples

tattoo on the left inner forearm. ∼ artist. ∼ ∼ photo sharing website

- shaped cloud formation over a city. ∼ photo by person. #zn. - #zn.0 # christmas

and illustration of the new year. photo by person.

s and drawings on the ceiling of a building.

- cut emerald - cut diamonds are a perfect addition to any home.

day in the green forest.

for the first time! by person, the man who is now on the right.

s are part of the annual event. organisation

ging it up : the model was spotted wearing a pair of black jeans, a white t - shirt and black t - shirt

s and other souvenirs at the market.

s of the day : flowers

dog in a bedroom.

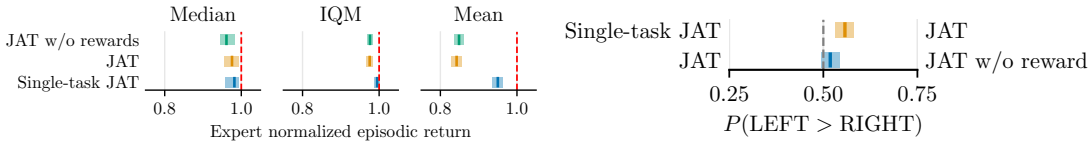
Figure G.16: JAT image captioning additional examples.

G.9 Reward as a task determinant

In multi-task learning, different environments may share identical dynamics and observational structures while differing in their ultimate goals (i.e., reward functions). Initially, the agent cannot distinguish the specific task it is facing. In most cases, this problem does not arise. For BabyAI, for example, the goal is an explicit part of the observation. For Atari, a single frame is sufficient to determine the game, and therefore the goal. In our dataset, the only domain that could be challenging in this respect is Meta-World, for which the structure of observations and dynamics is consistent across tasks. Note also that even in this case, it should be possible for the agent in some instances to infer the task from the initial conditions. We confirm this hypothesis in the following experiment.

To solve the problem of task indeterminacy, Gato introduces a method of pre-empting the sequence with an expert demonstration (prompt) to guide the agent. While this approach is effective, it imposes an important limitation: a demonstration must be available, and this demonstration must be sufficiently complete to clearly define the task. In the JAT model, we adopt a less restrictive and simpler approach by incorporating the reward signal directly into the observation encoding. We believe that this integration can, in most cases, provide the agent with sufficient context to remove ambiguity about the task at hand.

To support our hypothesis on the effectiveness of integrating reward signals into observations, we conducted an experiment with three different settings. First, to create a baseline where task indeterminacy is absent, we trained individual agents, each on a specific task from a random subset of 10 Meta-World tasks. This single-task training ensures that each agent is perfectly matched to its respective task, without any ambiguity. Next, we introduced a degree of indeterminacy by training a single model on the same 10 tasks without access to the reward signal, presenting a scenario that simulates a worst-case uncertainty condition. We compare these two settings with our full JAT model, i.e., with access to the reward signal, trained on the same selection of tasks. We compared the performance of these three scenarios, with the results detailed in Figures G.17 following the recommendations of (Agarwal et al., 2021).



(a) Aggregated metrics. (b) Probability of improvement.

Figure G.17: Results of the reward ablation. The vertical bars are the estimated values and the shaded areas are the 95% stratified bootstrap CIs. The experiments were conducted on a selection of 10 tasks from the Meta-World benchmark. Displayed are the results of an ablation study on our JAT model variations: Single-task JAT with each task learned by a dedicated agent; JAT without rewards where the training omits reward signals; and the full JAT model integrating reward signals. Results are based on 100 evaluations per task.

Firstly, it's notable that the JAT model trained on a single task surpasses other settings, thus demonstrating the existence of a negative impact of task indeterminacy. However, this impact is actually very minor, and even in the most unfavorable setting (JAT without reward), the normalized IQM score reaches $97.6 \pm 0.7\%$. This confirms the previously formulated intuition that the task can generally be inferred from the initial conditions. Then, when comparing the JAT model with and without access to the reward, we observe a probability of improvement from the former over the latter of $51.8 \pm 2.5\%$, indicating that the addition of the reward has a significant, albeit small, positive effect on resolving indeterminacy. Lastly, the most significant gap is observed in the average score. This can be attributed to the fact that this metric accounts for outliers. Here, the outliers are the tasks suffering from indeterminacy, for which the agent often fails to resolve the task.

In summary, the key insight from this study is that complex solutions like prompting are often not required to address this task indetermination issue, as it typically presents a minimal challenge. Furthermore, in instances where the problem does manifest, implementing a straightforward strategy like incorporating the reward into the observation proves to be an effective measure for mitigation.

Appendix H

TeachMyAgent: a Benchmark for Automatic Curriculum Learning in Deep RL

Contents

H.1 Introduction . . . 261
H.2 Related work . . . 264
H.3 ACL baselines . . . 265
H.4 The TeachMyAgent benchmark . . . 266
H.4.1 Environments . . . 266
H.4.2 Learners . . . 268
H.5 Experiments . . . 268
H.5.1 Experimental details . . . 268
H.5.2 Challenge-specific comparison with Stump Tracks . . . 269
H.5.3 Global performance analysis using the Parkour . . . 271
H.6 Open-Source release of TeachMyAgent . . . 272
H.7 Discussion and Conclusion . . . 273
H.8 Details on ACL baselines . . . 274
H.8.1 Implementation details . . . 274
H.8.2 Hyperparameters tuning . . . 279
H.9 Environment details . . . 280
H.9.1 Stump Tracks . . . 281
H.9.2 Parkour . . . 281
H.9.3 Morphologies . . . 286
H.10 Experimental details . . . 287
H.10.1 Deep RL Students . . . 287
H.10.2 General experimental setup . . . 288
H.10.3 Stump Tracks variants . . . 288
H.10.4 Parkour experiments . . . 291
H.11 Additional results . . . 293
H.11.1 Original Stump Tracks . . . 293
H.11.2 Challenge-specific comparison . . . 294
H.11.3 Parkour . . . 303

H.1 Introduction

Inspired by how structured and gradual human learning is, curriculum learning has long been identified as a key component for many machine learning problems (Selfridge et al., 1985; Elman, 1993; Bengio et al., 2009; Cangelosi & Schlesinger, 2015) in order to organize samples shown during learning. While such a curriculum can be hand-designed by human experts on the problem, the field of automatic curriculum learning (ACL) (Graves et al., 2017; Portelas et al., 2020b) focuses on designing teacher algorithms able to autonomously sequence learning problem selection so as to maximize agent performance (e.g., over a set of samples in supervised learning, or game levels in Deep RL).

Parallel to this, Deep RL researchers have been increasingly interested in finding methods to train generalist agents (Rajeswaran et al., 2017; Zhang et al., 2018; Cobbe et al., 2019) to go beyond initial successes on solving single problems, e.g., individual Atari games (Mnih et al., 2015) or navigation in fixed scenarios (Lillicrap et al., 2016; Haarnoja et al., 2018). Many works proposed novel Deep RL learning architectures able to successfully infer multi-purpose action policies when given an experience stream composed of randomly sampled *tasks* (Schaul et al., 2015; Hessel et al., 2018; Cobbe et al., 2019; Hessel et al., 2019). Here and thereafter *tasks* denote learning problems in general, for instance, multiple mazes to solve (a.k.a environments) or, in the context of robotic manipulation, multiple state configurations to obtain (i.e., goals) (Portelas et al., 2020b). To compare existing and future multi-task Deep RL agents, Cobbe et al. (2020) proposed a suite of 16 atari-like environments, all relying on procedural content generation (PCG) to generate a wide diversity of learning situations. The high-diversity induced by PCG has been identified as particularly beneficial to foster generalization abilities in Deep RL agents (Justesen et al., 2018; Risi & Togelius, 2020; OpenAI et al., 2019).

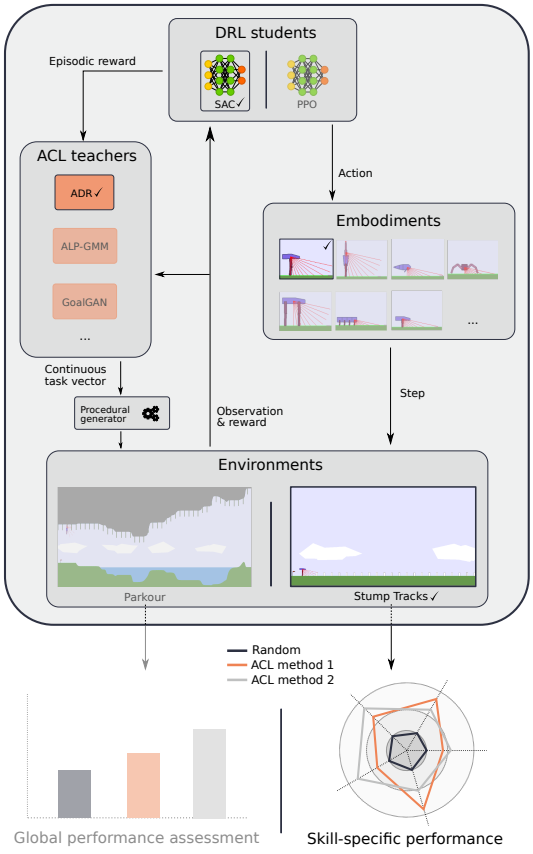


Figure H.1: **TeachMyAgent**: A benchmark to study and compare teacher algorithms in continuous procedural environments.

An important aspect not covered by these prior works is that they all rely on proposing randomly selected tasks to their agent, i.e., they do not consider using a curriculum in learning. One can argue that random task selection is inefficient, especially when considering complex continuous task spaces, which can feature subspaces of varying difficulties ranging from trivial to unfeasible. Following this observation, many works attempted to train multi-task agents by pairing them with ACL algorithms (Portelas

et al., 2020b; Narvekar et al., 2020). The advantages of ACL over random task sampling for Deep RL agents have been demonstrated in diverse experimental setups, such as domain randomization for sim2real robotics (OpenAI et al., 2019; Mehta et al., 2020), video games (Salimans & Chen, 2018), or navigation in virtual environments (Florensa et al., 2018; Portelas et al., 2020a; Racaniere et al., 2020).

While this diversity of potential application domains and implementations of ACL hints at a promising future for this field, it also makes comparative analysis complicated, which limits large-scale adoption of ACL. For instance, depending on the ACL approach, the amount of required expert knowledge on the task space can range from close to none – as in Portelas et al. (2020a) – to a high amount of prior knowledge, e.g., initial task sampling subspace and predefined reward range triggering task sampling distribution shifts, as in OpenAI et al. (2019). Additionally, some ACL approaches were tested based on their ability to master an expert-chosen target subspace (Klink et al., 2020) while others were tasked to optimize their performance over the entire task space (Baranes & Oudeyer, 2009; Florensa et al., 2018; Portelas et al., 2020a). Besides, because of the considerable computational cost and implementation efforts necessary for exhaustive comparisons, newly proposed ACL algorithms are often compared to only a subset of previous ACL approaches Mehta et al. (2020); Portelas et al. (2020a); Racaniere et al. (2020). This computation bottleneck is also what prevents most works from testing their ACL teachers on a diversity of Deep RL students, i.e., given a set of tasks, they do not vary the student's learning mechanism nor the characteristics that may influence its learning dynamics (e.g., its embodiment). Designing a unified benchmark platform, where baselines would be shared allowing one to only run their approach and compare it to established results, could drive progress in this space.

Inspired by how the MNIST dataset (Deng, 2012), or the ALE Atari games suite (Bellemare et al., 2013), respectively catalyzed supervised learning and single-task RL research, we propose to perform this much-needed in-depth ACL benchmarking study. As such, we introduce *TeachMyAgent 1.0*[1], a teacher testbed featuring a) two procedural Box2D[2] environments with challenging task spaces, b) a collection of pre-defined agent embodiments, and c) multiple Deep RL student models. The combination of these three components constitutes a large panel of diverse teaching problems. We leverage this benchmark to characterize the efficiency of an ACL algorithm on the following key teaching challenges:

1. *Mostly unfeasible task spaces* - While using PCG systems to generate tasks allows proposing rich task spaces to Deep RL agents, which is good for generalization, such large spaces might contain a predominant amount of unfeasible (or initially unfeasible) tasks. A teacher algorithm must then have the ability to quickly detect and exploit promising task subspaces for its learner.

2. *Mostly trivial task spaces* - On the contrary, the task space might be mostly trivial and contain only few challenging subspaces, which is a typical scenario when dealing with a skilled student (e.g., that is already trained, or that has an advantageous

[1] http://developmentalsystems.org/TeachMyAgent/

[2] 2D game engine, used in OpenAI gym (Brockman et al., 2016)

embodiment). In that case, the teacher has to efficiently detect and exploit the small portion of subspaces of relevant difficulty.

3. *Forgetting students* - Deep RL learners are prone to catastrophic forgetting Kirkpatrick et al. (2017), i.e., to overwrite important skills while training new ones. This has to be detected and dealt with by the teacher for optimal curriculum generation.
4. *Robustness to diverse students* - Being able to adapt curriculum generation to diverse students is an important desideratum to ensure a given ACL mechanism has good chances to transfer to novel scenarios.
5. *Rugged difficulty landscapes* - Another important property for ACL algorithms is to be able to deal with task spaces for which the optimal curriculum is not a smooth task distribution sampling drift across the space but rather a series of distribution jumps, e.g., as in complex PCG-task spaces.
6. *Working with no or little expert knowledge* - Prior knowledge over a task space w.r.t. a given student is a costly information gathering process that needs to be repeated for each new problem/student. Relying on as little expert knowledge as possible is therefore a desirable property for ACL algorithms (especially if aiming for out-of-the-lab applications).

To precisely assess the proficiency of an ACL algorithm on each of these challenges independently, we extend a Box2D walker environment from Portelas et al. (2020a) into multiple unit-test variants, one per challenge, inspired by the structure of *bsuite* (Osband et al., 2020), a recent benchmark for RL agents. The second environment of our benchmark is the *Parkour* environment, inspired by Wang et al. (2020). It features a complex task space whose parameters seed a neural network-based procedural generation of a wide diversity of environments, in which there exists drastically different learning curricula depending on the agent's embodiment (see fig. H.1). To assess the ability of existing ACL methods to robustly adapt to diverse students, we consider a random black-box student scenario in the Parkour environment, i.e., the morphology (e.g., walker or climber) of the learner is randomly selected for each new training run.

Scope – More precisely, we conduct an in-depth comparative study of ACL approaches suited for generalist Deep RL agents in single-agent scenarios. We do not include works on self-play/multi-agent setups Hernandez et al. (2019); Hernandez-Leal et al. (2018) nor single-agent population-based approaches (Forestier et al., 2022; Wang et al., 2020). Also, we are interested in the problem of task selection from a continuous parameter space encoding the procedural generation of tasks. We do not consider ACL methods for discrete task sets Matiisen et al. (2017), sets of task spaces Forestier et al. (2022); Colas et al. (2019), or intrinsic reward learning (Pathak et al., 2019; Burda et al., 2019b). We assume this continuous space is given and relatively low-dimensional, as it already poses strong teaching challenges: we therefore leave the analysis of approaches that autonomously learn task representations for subsequent work Pong et al. (2020); Jabri et al. (2019); Kovač et al. (2023).

Our main contributions are:

- Identification of multiple challenges to be tackled by ACL methods, enabling multi-dimensional comparisons of these algorithms.

- TeachMyAgent *1.0*, a set of teaching problems (based on PCG environments) to study and compare ACL algorithms when paired with Deep RL students.
- Comparative study of representative existing ACL approaches, including both skill-specific unit-tests and global performance assessments, which highlights the competitiveness of methods not using expert knowledge and shows that our Parkour environment largely remains an open problem for current state-of-the-art ACL.
- Release of an open-source Python package, featuring 1) all environments, embodiments and Deep RL students from TeachMyAgent, 2) all studied ACL algorithms, that we either adapt to our API when code is available or re-implement from scratch if not open-sourced, 3) our experimental results as baselines for future works, and 4) tutorials & reproducibility scripts.

H.2 Related work

Many environment suites already exist to benchmark Deep RL algorithms: some of them leverage video games, which provide challenging discrete action spaces, e.g., Atari 2600 Games as in Bellemare et al. (2013) or Sonic The Hedgehog levels in Nichol et al. (2018). To study and develop Deep RL agents suited for complex continuous control scenarios, the community predominantly used the MuJoCo physics engine (Todorov et al., 2012). The Deep Mind Lab (Beattie et al., 2016) provides customizable puzzle-solving environment, particularly well suited to study goal-conditioned policies learning from pixels in rich 3D environments. At the intersection of Deep RL and Natural Language Processing, benchmark environments such as TextWorld (Côté et al., 2019) or BabyAI (Chevalier-Boisvert et al., 2019) were also designed to provide a testbed to develop an autonomous agent receiving linguistic goals and/or interacting using language. The *bsuite* benchmark (Osband et al., 2020) leverages unit-tests to assess the core capabilities of Deep RL methods (e.g., generalization, memory). In all these previous works, the Deep RL agent is learning in one or a few environments presented randomly and/or intrinsically chooses goals within those predefined environments, and the long-term community objective is to find more efficient learning architectures. On the contrary, the objective of TeachMyAgent is to foster the development of new teacher algorithms whose objective is, given a task space and a Deep RL student, to most efficiently organize the learning curriculum of their Deep RL student such that its performance is maximized over the task set. In other words, it is not about finding efficient learning architectures but about finding efficient curriculum generators.

Perhaps closest to our work is the Procgen benchmark (Cobbe et al., 2020), which features several Atari-like environments, all having unique procedural generation systems, allowing for the generation of a wide diversity of learning situations, particularly well suited to assess the generalization abilities of Deep RL agents. While they rely on an uncontrolled, random procedural generation, we assume control over it, which enables the use of ACL methods to select parameters encoding task generation. An interesting future work, parallel to ours, would be to modify the Procgen benchmark to allow direct control over the procedural generation.

Because of the current lack of any ACL benchmark, most recently proposed ACL

algorithms have relied on designing their own set of test environments. Florensa et al. (2018) used a custom MuJoCo Ant maze in which the ACL approach is in control of which end-position to target. Klink et al. (2020) used another MuJoCo Ant maze and ball-catching environment featuring a simulated Barrett WAM robot. While these previous works studied how to control goal selection in a given fixed environment, we are interested in the arguably more challenging problem of controlling a rich parametric procedural generation. Portelas et al. (2020a) already studied ACL in Stump Tracks, a procedural Box2D environment that we include and extend in TeachMyAgent, however, it did not perform an extensive comparative study as what we propose in the present work. Racaniere et al. (2020) also used procedural generation to test their ACL approach, however they only compared their ACL algorithm to Goal-GAN (Florensa et al., 2018), and did not open-source their environments. Additionally, in contrast with all previously cited ACL works, in TeachMyAgent we propose an in-depth analysis of each approach through multiple unit-test experiments to fully characterize each teacher.

H.3 ACL baselines

In the following paragraphs, we succinctly frame and present all the ACL algorithms that we compare using TeachMyAgent. More detailed explanations are left to Section H.8.

Framework – Given a Deep RL student s and an n-dimensional task-encoding parameter space $\mathcal{T} \in \mathbb{R}^n$ (i.e., a task space), the process of automatic curriculum learning aims to learn a function $\mathcal{A} : \mathcal{H} \mapsto \mathcal{D}(\mathcal{T})$ mapping any information retained about past interactions with the task space to a distribution of tasks.

One can define the optimization objective of an ACL policy given an experimental budget of E episodic tasks as:

$$\max_{\mathcal{A}} \int_{\mathrm{T} \sim \mathcal{D}_{target}} P_{\mathrm{T}}^{E} \, \mathrm{dT}, \tag{H.1}$$

with $\mathcal{D}_{target}$ the distribution of test tasks over the task space and P the post-training performance (e.g., episodic reward, exploration score) of a student s on task T after E episodes. Since it is usually difficult to directly optimize for this objective, various surrogate objectives have been proposed in the literature. See Portelas et al. (2020b) for a review and classification of recent ACL works.

Expert-knowledge – To ease the curriculum generation process, multiple forms of expert knowledge have been provided in current ACL approaches. We propose to gather them in three categories: 1) use of initial task distribution $\mathcal{D}_{init}$ to bootstrap the ACL process, 2) use of a target task distribution $\mathcal{D}_{target}$ to guide learning, and 3) use of a function interpreting the scalar episodic reward sent by the environment to identify mastered tasks (*Reward mastery range*). For each implemented ACL method, we highlight its required prior knowledge over the task space w.r.t a given Deep RL agent in table H.1. We hope that this classification will ease the process of selecting an ACL method for researchers and engineers, as available expert knowledge is (arguably) often what conditions algorithmic choices in machine learning scenarios.

Table H.1: Expert knowledge used by the different ACL methods. We separate knowledge required (REQ.) by algorithms, optional ones (OPT.), and knowledge not needed (empty cell).

Algorithm	$\mathcal{D}_{init}$	$\mathcal{D}_{target}$	Reward mastery range
ADR	req.		req.
ALP-GMM	opt.		
Covar-GMM	opt.		
Goal-GAN	opt.		req.
RIAC			
SPDL	req.	req.	
Setter-Solver		opt.	req.

Implemented baselines – We compare seven ACL methods, chosen to be representative of the diversity of existing approaches, that can be separated in three broad categories. First, we include three methods relying on the idea of maximizing the Learning Progress (LP) of the student: RIAC (Baranes & Oudeyer, 2009), Covar-GMM (Moulin-Frier & Oudeyer, 2013) and ALP-GMM (Portelas et al., 2020a). We then add in our benchmark Goal-GAN (Florensa et al., 2018) and Setter-Solver (Racaniere et al., 2020), both generating tasks using deep neural networks and requiring a binary reward for mastered/not mastered tasks, pre-defined using expert knowledge. Finally, we append to our comparison two ACL algorithms using the idea of starting from an initial distribution of tasks and progressively shifting it regarding the student's capabilities: ADR (OpenAI et al., 2019) (inflating a task distribution from a single initial task based on student mastery at each task distribution's border) and SPDL (Klink et al., 2020) (shifting its initial distribution towards a target distribution). We also add a baseline teacher selecting tasks uniformly random over the task space (called Random).

H.4 The TeachMyAgent benchmark

In the following section, we describe available environments and learners in TeachMyAgent. We propose two Box2D environments with procedural generation allowing to generate a wide variety of terrains. Both our environments are episodic, use continuous action/observation spaces and return scalar rewards. In addition, we provide two Deep RL algorithms as well as multiple agent morphologies. An experiment is thus constituted of an ACL method, an environment and a learner (i.e., an embodied Deep RL algorithm).

H.4.1 Environments

Stump Tracks environment – Stump Tracks is an extension of a parametric Box2D environment initially presented in Portelas et al. (2020a). The learning policy is embodied into a walker agent whose motors are controllable with torque (i.e., continuous action space). The observation space is composed of lidar sensors, head position and joint positions. The walker is rewarded for going forward and penalized for torque usage.

An episode lasts 2000 steps at most, and is terminated if the agent reaches the end of the track or if its head collides with the environment (in which case a −100 reward is received). A 2D parametric PCG is used for each new episode: it controls the height and spacing of stumps laid out along the track (see fig. H.2 and app. H.9). We chose to feature this environment as its low-dimensional task space is convenient for visualizations and modifications. We derive multiple variants of Stump Tracks (e.g., by extending the task space boundaries or shuffling it) to design our unit-tests of ACL challenges (see sec. H.1 and sec. H.5).

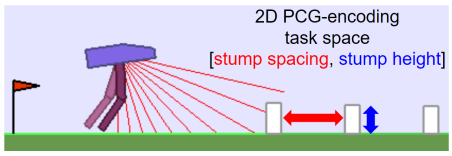


Figure H.2: *Stump Tracks*, a simple parametric env. to study ACL algorithms with Deep RL students.

Parkour environment – Inspired by both Stump Tracks and another Box2D environment from Wang et al. (2020), we present the parametric Parkour environment: a challenging task space with rugged difficulty landscape, few prior knowledge definable, and requiring drastically different learning curricula depending on the agent's embodiment.

It features an uneven terrain (see figure H.1) composed of a ground and ceiling encoded through a Compositional Pattern-Producing Network (CPPN) (Stanley, 2007). This CPPN, whose weights and architecture are kept fixed, takes an additional input vector of bounded real numbers, which acts as the parameters controlling terrain generation. This neural network-based generation enables the creation of a task space with a rugged difficulty landscape (see Section H.9), requiring time-consuming exploration from an expert to seek trivial subspaces. We propose three versions of this task space (i.e., three possible bounds for the CPPN's input vector): easy, medium (used in the experiments of this work), and hard. The Parkour environment also features graspable objects, called "creepers", creating a niche for climbing morphologies. Similarly to the stumps in Stump Tracks, the creepers' generation is controlled by their height and the space between them. The Parkour's task space also contains a dimension controlling the "water" level of the track, ranging from 0 (no water) to 1 (entire parkour under water). Water adds new physic rules aiming to imitate (in a simplified way) physics of water.

The resulting 6D task space (3 for the CPPN's input, 2 for creepers and 1 for water) creates a rich environment in which the optimal curriculum will largely depend on the agent's embodiment (e.g., swimming agents need high levels of water, while climbers and walkers need low levels). Note that, as in Stump Tracks, each episode lasts 2000 steps, agents are rewarded for moving forward (and penalised for using torque) and have access to lidars, head position, joint positions, and also additional information (see Section H.9).

H.4.2 Learners

Embodiments – As aforementioned, we introduce new morphologies using swimming and climbing locomotion (e.g., fish, chimpanzee, see figure H.1). TeachMyAgent also features the short walker and quadrupedal walker from Portelas et al. (2020a) as well as new walking morphologies such as the spider and the millipede (see figure H.1).

Deep RL algorithms – To benchmark ACL algorithms, we rely on two different state-of-the-art Deep RL algorithms: 1) Soft-Actor-Critic (Haarnoja et al., 2018) (SAC), a now classical off-policy actor-critic algorithm based on the dual optimization of reward and action entropy, and 2) Proximal Policy Optimization (PPO) (Schulman et al., 2017), a well-known on-policy Deep RL algorithm based on approximate trust-region gradient updates. We use OpenAI Spinningup's implementation[3] for SAC and OpenAI Baselines' implementation[4] for PPO. See Section H.10 for implementation details.

H.5 Experiments

We now leverage TeachMyAgent to conduct an in-depth comparative study of the ACL algorithms presented in Section H.3. After discussing experimental details, we undergo two separate experiments, aiming to answer the following questions:

- How do current ACL methods compare on each teaching challenges proposed in sec. H.1 ?
- How do current ACL methods scale to a complex task space with limited expert knowledge ?

H.5.1 Experimental details

For both our environments, we train our Deep RL students for 20 million steps. For each new episode, the teacher samples a new parameter vector used for the procedural generation of the environment. The teacher then receives the cumulative episodic reward that can be potentially turned into a binary reward signal using expert knowledge (as in GoalGAN and Setter-Solver). Additionally, SPDL receives the initial state of the episode as well as the reward obtained at each step, as it is designed for non-episodic RL setup. Every 500000 steps, we test our student on a test set composed of 100 pre-defined tasks and monitor the percentage of test tasks on which the agent obtained an episodic reward greater than 230 (i.e., "mastered" tasks), which corresponds to agents that were able to reach the last portion of the map (in both Stump Tracks and Parkour). We compare performance results using Welch's t-test as proposed in Colas et al. (2018), allowing us to track statistically significant differences between two methods. We perform a hyperparameter search for all ACL conditions through grid-search (see Section H.8),

[3] https://spinningup.openai.com
[4] https://github.com/openai/baselines

while controlling that an equivalent number of configurations are tested for each algorithm. See Section H.10 for additional experimental details.

H.5.2 Challenge-specific comparison with Stump Tracks

First, we aim to compare the different ACL methods on each of the six challenges we identified and listed in Section H.1. For this, we propose to leverage the Stump Tracks environment to create five experiments, each of them designed to highlight the ability of a teacher in one the first five ACL challenges (see Section H.10 for details):

- *Mostly unfeasible task space*: growing the possible maximum height of stumps, leading to almost 80% of unfeasible tasks.
- *Mostly trivial task space*: allowing to sample stumps with negative height introducing 50% of new trivial tasks.
- *Forgetting student*: resetting the Deep RL model twice throughout learning (i.e., every 7 million steps).
- *Diverse students*: using multiple embodiments (short bipedal and spider) and Deep RL students (SAC and PPO).
- *Rugged difficulty landscape*: Applying a random transformation to the task space such that feasible tasks are scattered across the space (i.e., among unfeasible ones).

Additionally, in order to compare methods on the last challenge (i.e., the need of prior knowledge), we propose to perform each of our five experiments in three conditions:

- *No expert knowledge*: None of the prior knowledge listed in table H.1 is given. Hence only methods not requiring it can run in this setup.
- *Low expert knowledge*: Only reward mastery range information is accessible. We consider this as low prior knowledge as, while it requires some global knowledge about the task space, it does not require assumptions on the difficulty of specific subspaces of the task space.
- *High expert knowledge*: All the expert knowledge listed in table H.1 is given.

Note that in the *No expert knowledge* and *Low expert knowledge* setups, SPDL (and ADR in *Low expert knowledge*) uses an initial task distribution randomly chosen as a subset of the task space. Moreover, in order to make a fair comparison in the *High expert knowledge* condition, we modified the vanilla version of Covar-GMM and ALP-GMM such that they can use an expert-given initial task distribution.

Using these 15 experiments (5 challenges in 3 expert knowledge setups), we here introduce what is, to our knowledge, the first unit-test like experiment of ACL methods, allowing one to compare teachers in each of the challenges we previously introduced. Moreover, performing each of the five experiments in three expert knowledge setups allows to show how the (un)availability of expert knowledge impacts performance for

each method, which is hard to infer from each approach's original paper as they tend to focus only on the most ideal scenario. See Section H.10 for a detailed explanation of each experimental setup.

To conduct our analysis, each ACL method is used in 15 experiments with 32 seeds, except ADR, GoalGAN and Setter-Solver which cannot run in the *No expert knowledge* setup (i.e., only 10 experiments). We then calculate the aforementioned percentage of mastered test tasks on our test set (identical for all experiments), and average it over seeds. Performance results of all conditions can be visualized in figure H.3 as a ratio of the Random teachers' performance, our lower-baseline.

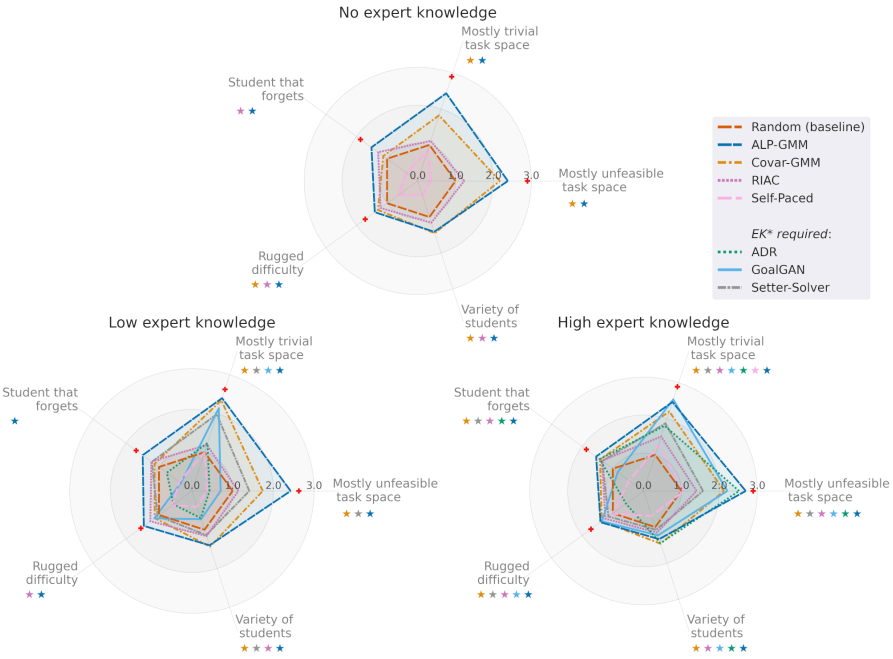


Figure H.3: *EK: Expert Knowledge*. Post-training performance of each ACL method as a ratio of Random's results on multiple teaching challenges, done with 3 different expert knowledge levels. We use ✚ to show estimations of upper-bound performances in each challenge, except for *Variety of students* (see Section H.11.1). On each axis, we indicate which method performed significantly better than Random ($p < 0.05$) using colored stars matching each method's color (e.g., ★ for Covar-GMM, ★ for ADR). See Section H.11.2 for details.

Results – We gather the results in figure H.3 as well as in Section H.11.2.

Expert-knowledge-free methods – Using these, one can see, first, that methods not requiring any expert knowledge (e.g., ALP-GMM or Covar-GMM) obtain very similar performances in *No expert knowledge* and in *High expert knowledge* setups (although expert knowledge does benefit them in terms of sample efficiency, see Section H.11.2 for details). Comparing their performance without prior knowledge to the results obtained by other teachers when they have access to high expert knowledge shows how competitive expert-knowledge-free methods can be.

Expert knowledge dependency – The *Low expert knowledge* setup highlights the dependence of methods relying on an initial distribution of easy tasks (e.g., ADR and GoalGAN), as it is not given in this scenario. As a result, in this setup, ADR obtains end performances not significantly different from Random in all challenges, and GoalGAN only outperforms Random in the mostly trivial task space ($p < 0.05$). This has to be compared with their performance on the *High expert knowledge* setup, in which both approaches reach the top 3 results on 3/5 challenges.

ADR & GoalGAN – Both ADR and GoalGAN have one strong weakness in a challenge (*Rugged difficulty* for ADR and *Forgetting student* for GoalGAN) that lead them to a performance worse than Random (significantly for ADR with $p < 0.05$) in all expert knowledge setups. For ADR, it can be explained by the fact that its expansion can get stuck by subspaces of very hard (or unfeasible) difficulty, and for GoalGAN, by its inability to adapt quickly enough to the student's regressing capabilities because of its inertia to update its sampling distribution (updating the buffer and training the GAN). We provide a more in-depth analysis of these two cases in Section H.11.2.

SPDL – One can see that SPDL obtains poor performance in our experimental setup: its end performance is significantly inferior to Random in 11/15 experiments ($p < 0.05$). This can be explained by the fact that SPDL, by design, optimizes performance over a Gaussian target distribution, while our test set is uniformly sampled over the task space (see Section H.8 for details and potential fixes). However, following the publication of our work, Klink et al. (2024) proposed an updated version of SPDL that uses optimal transport. They showed that this allows their method to handle uniform target distributions, leading to strong results in TeachMyAgent.

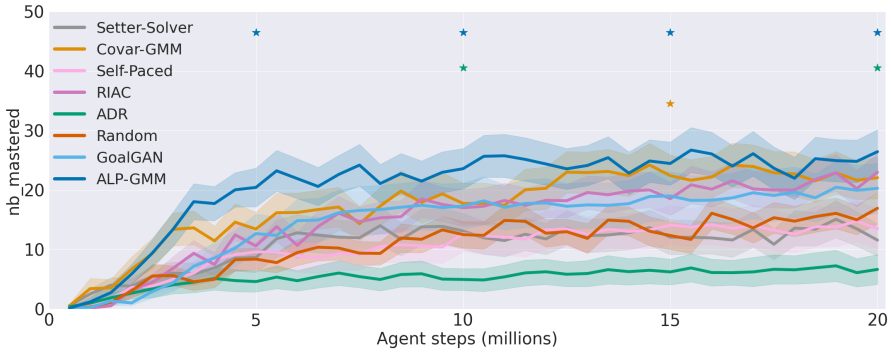


Figure H.4: Averaged performance (48 seeds, with standard error of the mean) for each ACL method on Parkour. We calculate every 5 million steps, for which the method obtained statistically different ($p < 0.05$) results from Random and indicate it with a star.

H.5.3 Global performance analysis using the Parkour

The second experiment we propose aims to more broadly benchmark ACL methods' performance in the Parkour environment, which features most of the previously discussed ACL challenges: 1) most tasks are unfeasible, 2) before each run, unknown to the teacher,

the student's embodiment is uniformly sampled among three morphologies (bipedal walker, fish and chimpanzee), requiring the teacher to adapt curriculum generation to a diversity of student profiles, and 3) tasks are generated through a CCPN-based PCG, creating a rich task space with rugged difficulty landscape and hardly-definable prior knowledge (see Section H.9).

We perform 48 seeded experiments (i.e. 16 seeds per morphology). To evaluate performance, three test sets were hand-designed (one per embodiment) such that each contains an even distribution between easy, medium and hard tasks. In terms of expert knowledge for teachers, we only give reward mastery range. Without straightforward initial easy task distribution to give to teachers requiring such knowledge (ADR and SPDL), we set it randomly over the space for each new run. See Section H.10 for details.

Results – We present the evolution of performance of each teacher averaged over all seeds (and thus all embodiments) in figure H.4 and gather the detailed results in Section H.11.3. Interestingly, one can observe that best-performing methods do not use expert knowledge. This is explained by the fact that few prior knowledge is provided to the teachers in these experiments and, as shown in the challenge-specific experiments, most methods using expert knowledge heavily rely on them to reach high-performance. However, one can see that, while SPDL and Setter-Solver remain at the performance level of Random, GoalGAN's performance along training is (mostly) not significantly different from those of Covar-GMM and RIAC, two methods not relying on expert knowledge, as opposed to GoalGAN. On his side, ADR seems to plateau very fast and finally reach an average performance significantly worse than Random ($p < 0.05$, see figure H.14). Indeed, as the difficulty landscape of the Parkour environment is rugged, and the initial "easy" task distribution randomly set, ADR is unable to progressively grow its sampling distribution towards feasible subspaces. Finally, when looking specifically to each embodiment type, results show the incapacity of all teachers to make the Deep RL student learn an efficient policy with the climbing morphology (i.e., at most 1% of mastered tasks by the end of training across all teachers), although we are able to show that high-performing policies can be learned when considering a subspace of the task space (see our case study in Section H.11.3). This might be due to the complexity of learning the climbing gait w.r.t walking or swimming, as it requires for instance good coordination skills between the arms and the grasping actions. For the two other morphologies (bipedal walker and fish), results obtained are also low (respectively less than 60% and 50%) and have a high variance (especially for the fish) considering that our test sets contain feasible tasks. This makes the Parkour environment an open challenge for future work on designing ACL algorithms.

H.6 Open-Source release of TeachMyAgent

With the open-source release of TeachMyAgent (version *1.0*), we hope to provide a tool that can be used as a step towards thorough comparison and better understanding of current and future ACL methods. TeachMyAgent's documented repository features the code of our environments, embodiments, Deep RL students, as well as implementations of all ACL methods compared in this work. All of these parts use APIs we provide

such that one can easily add its ACL method, learning algorithm, and new embodiment or environment. We hope this will foster community-driven contributions to extend TeachMyAgent in order to broaden its impact and adapt it to the future of ACL. We also provide the code we used to reproduce our experiments, as well as Jupyter notebooks that allow us to generate all the figures shown in this work. Finally, we release the results of our benchmark, allowing one to load them and compare its ACL method against baselines without having to reproduce our large-scale experiments.

H.7 Discussion and Conclusion

In this work, we presented TeachMyAgent *1.0*, a first extensive testbed to design and compare ACL algorithms. It features unit-test environments to assess the efficiency of a given teacher algorithm on multiple core skills and the Parkour environment, which provides a challenging teaching scenario that has yet to be solved. We used TeachMyAgent to conduct a comparative study of existing ACL algorithms. Throughout our experiments, we identified that 1) current ACL approaches not using expert knowledge matched and even outperformed (e.g., ALP-GMM) other approaches using high amounts of expert knowledge, and 2) the Parkour environment is far from solved, which makes it a good candidate as a testbed when designing new ACL approaches.

Limitations & future work. – An obvious extension of this work is the addition of recent ACL approaches proposed during or after our experimental campaign (Dennis et al., 2020; Zhang et al., 2020; Jiang et al., 2021b,a; Parker-Holder et al., 2022; Rutherford et al., 2024). So far, all studied ACL algorithms have struggled to detect feasible task subspaces in Parkour, hinting that more research is needed to improve the "progress niche detection" ability of current teacher algorithms.

TeachMyAgent currently only features environments with low-dimensional PCG systems. Designing new environments with higher-dimensional PCG, which might require to learn low-dimensional representations on which to apply ACL algorithms, is an interesting avenue. Besides, our current list of environments only studies 2D locomotion tasks inspired by ALP-GMM's original paper (Portelas et al., 2020a) as well as other works on Deep RL and 2D locomotion (Ha, 2019; Song et al., 2018; Gaier & Ha, 2019; Wang et al., 2019, 2020). While we put maximal effort into building a thorough and fair analysis of ACL methods, we believe extending TeachMyAgent with other environments (e.g., ProcGen Cobbe et al. (2019), robotic manipulation) would make the benchmark even more informative. Moreover, TeachMyAgent currently only focuses on maximizing the student's performance over the complete task space. We believe it is worth investigating other objectives considered in the ACL literature, such as maximizing the performance over a pre-defined hard subspace (Klink et al., 2020, 2024), or maximizing the worst case generalization performance (Dennis et al., 2020; Parker-Holder et al., 2022; Rutherford et al., 2024)

Finally, TeachMyAgent considers experiments and methods that scaffold the learning of a single student. Recently, Portelas et al. (2020c) advocated for considering the problem of *Meta ACL*: algorithms *learning to learn to teach* across a sequence of students. TeachMyAgent could be extended to also consider Meta ACL teachers.

H.8 Details on ACL baselines

In this section, we provide details about our implementations of ACL methods and their hyperparameter tuning.

H.8.1 Implementation details

Random – We use as baseline a random teacher, which samples tasks using a uniform distribution over the task space.

ADR – OpenAI et al. (2019) introduced *Automatic Domain Randomization* (ADR), an ACL method relying on the idea of *Domain Randomization* (Tobin et al., 2017; Peng et al., 2018). Instead of sampling tasks over the whole task space, ADR starts from a distribution centered on a single example easy for the student and progressively grows the distribution according to the learning agent's performance. Using this mechanism, it increases the difficulty of the tasks proposed to the student while still sampling in previously seen regions in order to reduce potential forgetting.

This sampling distribution P_ϕ is parameterized by $\phi \in \mathbb{R}^{2d}$ (with d the number of dimensions of the task space). For each dimension, a lower and upper boundary are set $\phi = \{\phi_i^L, \phi_i^H\}_{i=1}^d$, allowing uniformly sampling on each dimension using these boundaries and obtain a task λ:

$$P_\phi(\lambda) = \prod_{i=1}^{d} U(\phi_i^L, \phi_i^H)$$

At the beginning, ϕ is centered on a single example (i.e. $\phi_i^L = \phi_i^H \; \forall i$). Then, at each episode, 1) ADR starts by sampling a new task $\lambda \sim P_\phi$. Following this, 2) ADR chooses with a probability p_b whether to modify λ in order to explore the task space or not. It thus samples a value ϵ uniformly in $[0; 1]$ and checks whether $\epsilon < p_b$. If this is not the case, ADR sends λ to the environment.

Otherwise, 3) ADR selects uniformly one of the dimensions of the task space, which we will call j as an example. Following this, 4) one of the two boundaries ϕ_j^L or ϕ_j^H is selected (50% chances for each boundary). Finally, 5) ADR replaces the j-th value of λ by the selected boundary and sends λ to the environment.

Moreover, ADR keeps a buffer D_i^L and D_i^H for each dimension i in the task space. Every time ϵ is greater than p_b and a value of λ is replaced by one of the selected boundary, ADR stores the episodic reward obtained at the end of the episode in the buffer associated to the selected boundary (e.g., the episodic reward is stored in D_k^L if the k-th value of lambda was replaced by ϕ_k^L).

Every time one of the buffers' size reaches m, the average $\overline{p}$ of episodic reward stored is calculated. Then, $\overline{p}$ is compared to two thresholds t_L and t_H (being hyperparameters of ADR) in order to know whether the boundary associated with the buffer must be reduced or increased.

As an example, say that D_k^L's size reached m, meaning that ϕ_k^L is the associated dimension (i.e., a λ sampled got its k-th value replaced by ϕ_k^L m times). Its average

episodic reward $\overline{p}$ is calculated. It is first compared to t_L and, if $\overline{p} < t_l$, ϕ_k^L is increased by Δ (as ϕ_k^L is a lower boundary, this means that the task space is reduced). Similarly, if $\overline{p} > t_l$, ϕ_k^L is decreased by Δ (expanding the task space).

If instead of D_k^L we take D_k^H, our task space has to be expanded or reduced in the same way: if $\overline{p} < t_L$ then ϕ_k^H is reduced by Δ (as it is now an upper boundary of the task space) and if $\overline{p} > t_H$ then ϕ_k^H is increased by Δ. Finally, note that whenever one buffer's size reaches m, it is then emptied.

As no implementation was provided by the authors, we propose here an implementation that is as close as possible to the algorithms given in OpenAI et al. (2019).

RIAC – Proposed in Baranes & Oudeyer (2009), Robust Intelligent Adaptive Curiosity is based on the recursive splitting of the task space in hyperboxes, called regions. One region is split into two whenever a pre-defined number max_s of sampled tasks originate from the region. The split value is chosen such that there is maximal Learning Progress (LP) difference between the two regions, while maintaining a size min_d (i.e., a ratio of the size of the whole task space) for each region. The number of possible splits to attempt is parameterized by n. We reuse the implementation and the value of the hyperparameters not mentioned here from Portelas et al. (2020a). RIAC does not require expert knowledge.

Covar-GMM – Covar-GMM was proposed in Moulin-Frier & Oudeyer (2013). As for RIAC, it does not require any expert knowledge and is based on learning progress. The core idea of Covar-GMM is to fit a Gaussian Mixture Model (of maximum size max_k) every n episodes on recently sampled tasks *concatenated with both a time dimension and a competence dimension*. The Gaussian from which to sample a new task is then chosen proportionally to its respective learning progress, defined as the positive correlation between time and competence. Additionally, in order to preserve exploration, Covar-GMM has a probability r_p of uniformly sampling a task instead of using one of its Gaussians. We use the implementation and hyperparameters from Portelas et al. (2020a), which uses Absolute Learning Progress (ALP) instead of LP.

Moreover, as mentioned in Section H.5, we modified the implementation to make it use expert knowledge (i.e., an initial distribution) when provided. Hence, instead of uniformly sampling tasks over the whole task space during the bootstrap phase at the beginning of training, Covar-GMM samples tasks from an initial Gaussian distribution of tasks provided by the expert.

ALP-GMM – ALP-GMM is an ACL algorithm inspired by Covar-GMM, proposed in Portelas et al. (2020a). Instead of relying on time competence correlation, which only allows to compute ALP over a single GMM fit, it computes a per-task ALP from the entire history of sampled tasks using a knn-based approach similar to those proposed in Forestier et al. (2022). Recent tasks are periodically used to fit a GMM on recently sampled tasks *concatenated with their respective ALP value*. The Gaussian from which to sample is then selected based on its mean ALP dimension. ALP-GMM does not require expert knowledge and has the same hyperparameters as Covar-GMM. We reused the implementation and hyperparameters (except max_k, n and r_p) provided by Portelas et al.

(2020a).

Additionally, as for Covar-GMM, we added the possibility to ALP-GMM to bootstrap tasks for an initial Gaussian distribution if the latter is provided, instead of uniformly bootstrapping tasks.

Goal-GAN – Another teacher algorithm we included in this benchmark is GoalGAN (Florensa et al., 2018), which relies on the idea of sampling goals (initially states to reach in the environment) where the agent performs neither too well nor too badly, called *Goals Of Intermediate Difficulty* (GOID). However, as this goal generation introduces a curriculum in the agent's learning, one can see the goal selection process as a task selection process. We will thus call them tasks instead of goals in the following description. For sampling, Florensa et al. (2018) proposed to use a modified version of a *Generative Adversarial Network* (GAN) (Goodfellow et al., 2014) where the generator network is used to generate tasks for the student given a random noise, and the discriminator is trained to classify whether these tasks are of "intermediate difficulty". To define such an "intermediate difficulty", GoalGAN uses a binary reward signal defining whether the student succeeded in the proposed task. As our environments return scalar rewards, this implies a function interpreter hand-designed by an expert (in our case, we set a threshold on the scalar reward, as explained in Section H.10). For each task sampled, the teacher proposes it multiple times ($n_{rollouts}$) to the student and then calculates the average of successes obtained (lying in $[0; 1]$). Using a lower threshold R_{min} and an upper threshold R_{max}, GoalGAN calculates if the average lies in this interval of tasks, neither too easy (with an average of successes very high) nor too hard (with an average of successes very low). If this is the case, this task is labeled as 1 for the discriminator (0 otherwise). This new task is then stored in a buffer (except if it already exists in the buffer a task at an euclidean distance smaller than ϵ from our new task). Every time a task has to be sampled, in order to prevent the GAN from forgetting previously seen GOIDs, the algorithm has the probability p_{old} of uniformly sampling from the buffer instead of using the GAN. Finally, the GAN is trained using the tasks previously sampled every n episode.

Note that, in order to help the GAN generate tasks in a feasible subspace of the task space at the beginning of training, GoalGAN can also pre-train its GAN using trivial tasks. In the original paper, as tasks are states, the authors proposed to use the student to interact with the environment and use collected states as achievable tasks. However, in our case, this is not possible. We thus chose to reuse the same trick as the one in (Klink et al., 2020), which uses an initial Gaussian distribution to sample tasks and label them as positives (i.e., tasks of intermediate difficulty) to pre-train the GAN. See Section H.10 for more details on this initial distribution.

We reused and wrapped the version[5] of GoalGAN implemented by Klink et al. (2020), which is a slightly modified implementation of the original one made by Florensa et al. (2018). Our generator network takes an input that has the same number of dimensions as our task space, and uses two layers of 256 neurons with ReLU activation (and TanH activation for the last layer). Our discriminator uses two layers of 128 neurons. For ϵ, we used a distance of 10% on each dimension of the task space. As per Florensa et al.

[5] https://github.com/psclklnk/spdl

(2018), we set R_{min} to 0.25 and R_{max} to 0.75. Finally, as in the implementation made by Klink et al. (2020), we set the amount of noise δ added to each goal sampled by the generator network as a proportion of the size of the task space.

Self-Paced – Proposed by Klink et al. (2020), *Self-Paced Deep Reinforcement Learning* (SPDL) samples tasks from a distribution that progressively moves towards a target distribution. The intuition behind it can be seen as similar to the one behind ADR, as the idea is to start from an initial task space and progressively shift it towards a target space, while adapting the pace to the agent's performance. However, here, all task distributions (initial, current, and target) are Gaussian distributions. SPDL thus maintains a current task distribution from which it samples tasks and changes it over training. This distribution shift is seen as an optimization problem using a dual objective: 1) maximizing the agent's performance over the current task space, while 2) minimizing the Kullback-Leibler (KL) divergence between the current task distribution and the target task distribution. This forces the task selection function to propose tasks where the agent performs well while progressively going towards the target task space.

Initially designed for non-episodic RL setups, SPDL, unlike all our other teachers, receives information at every step of the student in the environment. After an offset of n_{OFFSET} first steps, and then every n_{STEP} steps, the algorithm estimates the expected return for the task sampled $\mathrm{E}_{p(c)}[J(\pi, c)]$ using the value estimator function of the current student (with $p(c)$ the current task distribution, π the current policy of the student, and $J(\pi, c)$ the expected return for the task c with policy π).

With this, SPDL updates its current sampling distribution in order to maximize the following objective w.r.t. the current task distribution $p(c)$:

$$\max_{p(c)} \mathrm{E}_{p(c)}[J(\pi, c)]$$

Additionally, a penalty term is added to this objective function, such that the KL divergence between $p(c)$ and the target distribution $\mu(c)$ is minimized. This penalty term is controlled by an α parameter automatically adjusted. This parameter is first set to 0 for K_α optimization steps and is then adjusted in order to maintain a constant proportion ζ between the KL divergence penalty and the expected reward term (see Klink et al. (2020) for more details on the way α is calculated). This optimization step is made such that the shift of distribution is not bigger than ϵ (i.e., $s.t. D_{\text{KL}}(p(c)||q(c)) \leq \epsilon$ with a shift from $p(c)$ to $q(c)$).

We reused the same implementation made by Klink et al. (2020) and wrapped it to our teacher architecture. However, as shown in Section H.5, using a Gaussian target distribution does not match with our Stump Tracks test set, where tasks are uniformly sampled over the whole task space. In order to solve this issue, some adaptations to its architecture could be explored (e.g., using a truncated Gaussian as target distribution to get closer to a uniform distribution). Klink et al. (2024) proposed an updated version of SPDL that uses optimal transport. They showed that this allows their method to handle uniform target distributions, leading to strong results in TeachMyAgent.

For the value estimators, we used the value network of both our PPO and SAC implementations (with the value network sharing its weights with the policy network for

PPO). For the calculation of α, we chose to use the average reward, as in the experiments of Klink et al. (2020). We did not use the lower bound restriction on the standard deviation of the task distribution σ_{LB} proposed in Klink et al. (2020) as our target distributions were very large (see Section H.10).

Setter-Solver – Finally, the last ACL algorithm we implemented here is Setter-Solver (Racaniere et al., 2020). In a very similar way to Goal-GAN, this method uses two neural networks: a *Judge* (replacing the discriminator) and a *Setter* (replacing the generator), outputting a task given a feasibility scalar in $[0; 1]$. During the training, the *Judge* is trained to output the right feasibility given a task sampled, and is used in the *Setter*'s losses to encourage the latter to sample tasks where the predicted feasibility was close to the real one. The *Setter* is also trained to sample tasks the student has succeeded (i.e. using a binary reward signal as Goal-GAN) while maximizing an entropy criterion encouraging it to sample diverse tasks.

For the implementation, Racaniere et al. (2020) provided code to help reproducibility that implements both the *Setter* and *Judge*, but did not include losses or optimization functions. Therefore, we provide here our own implementation of the full Setter-Solver algorithm, trying to be as close as possible to the paper's details. We reused the code provided for the two neural networks and modified it to add losses, optimizers, and some modifications to better integrate it to our architecture. We kept the tricks added in the code provided by authors that uses a non-zero uniform function to sample the feasibility and a clipped sigmoid in the *Setter*'s output. Concerning the generator network, we kept the hyperparameters of the paper (i.e., a RNVP (Dinh et al., 2017) with three blocks of three layers) except the size of hidden layers n_{HIDDEN} that we optimized. We also reused the three layers of 64 neurons architecture for the *Judge* as per the paper. Note that we used an Adam optimizer with a learning rate of $3 \cdot 10^{-4}$ for both the *Setter* and the *Judge*, while this was not specified for the *Judge* in Racaniere et al. (2020). We optimized the upper bound δ of the uniformly sampled noise that is added to succeed tasks in the validity *Setter*'s loss, as well as the update frequency n.

We did not use the conditioned version of the *Setter* or *Judge*. Indeed, first we generate the task before obtaining the first observation in our case, as opposed to Racaniere et al. (2020), and also because the first observation of an embodiment is always the same, as both our environments have a startpad (see Section H.9). Finally, we did not use the additional target distribution (called *desired goal distribution* in the original paper) loss that uses a Wassertein discriminator (Arjovsky et al., 2017) to predict whether a task predicted belongs to the target distribution. Indeed, as shown in Racaniere et al. (2020), using the targeted version of Setter-Solver offers more sample efficiency but leads to similar final results. Moreover, in our case, a target distribution is known only in the *High expert knowledge* setup of the challenge-specific experiments, in addition to having this part not implemented at all in the code provided by the authors. We thus leave this upgrade to future work.

H.8.2 Hyperparameters tuning

In order to tune the different ACL methods to our experiments, we chose to perform a grid search using our Stump Tracks environment with its original task space. As the Parkour is partly extended from it, in addition of the challenge-specific experiments, this environment offered us an appropriate setup. Each point sampled in the grid-search was trained for 7 million steps (instead of the 20 million used in our experiments) with 16 seeds in order to reduce the (already high) computational cost. At the end of training, we calculated the percentage of mastered tasks on the test set for each seed. The combination of hyperparameters having the best average over its seeds was chosen as the configuration for the benchmark.

In order to make the grid-search as fair as possible between the different ACL methods, given that the number of hyperparameters differs from one method to another, we sampled the same number of points for each teacher: 70 ($\pm$10). The hyperparameters to tune for each teacher, as well as their values, were chosen following the recommendations given by their original paper.

Moreover, we chose to tune the teachers in what we call their "original" expert knowledge version (i.e., they have access to the same amount of prior knowledge as the one they used in their paper). Hence, teachers requiring expert knowledge use our high expert knowledge setup, and algorithms such as ALP-GMM use no expert knowledge.

Table H.2 shows the values we tested for each hyperparameter and the combinations that obtained the best result.

Table H.2: Hyperparameters tuning of the ACL methods.

ACL method	Hyperparameter	Possible values	Best value
ADR	t_L	$[0, 50]$	0
ADR	t_H	$[180, 230, 280]$	180
ADR	p_b	$[0.3, 0.5, 0.7]$	0.7
ADR	m	$[10, 20]$	10
ADR	Δ	$[0.05, 0.1]$	0.1
RIAC	max_s	$[50, 150, 250, 350]$	150
RIAC	n	$[25, 50, 75, 100]$	75
RIAC	min_d	$[0.0677, 0.1, 0.1677, 0.2]$	0.1
Covar-GMM	n	$[50, 150, 250, 350]$	150
Covar-GMM	max_k	$[5, 10, 15, 20]$	15
Covar-GMM	r_p	$[0.05, 0.1, 0.2, 0.3]$	0.1
ALP-GMM	n	$[50, 150, 250, 350]$	150
ALP-GMM	max_k	$[5, 10, 15, 20]$	10
ALP-GMM	r_p	$[0.05, 0.1, 0.2, 0.3]$	0.05
GoalGAN	δ	$[0.01, 0.05, 0.1]$	0.01
GoalGAN	n	$[100, 200, 300]$	100
GoalGAN	p_{old}	$[0.1, 0.2, 03]$	0.2
GoalGAN	$n_{rollouts}$	$[2, 5, 10]$	2
SPDL	n_{OFFSET}	$[100000, 200000]$	200000
SPDL	n_{STEP}	$[50000, 100000]$	100000
SPDL	K_α	$[0, 5, 10]$	0
SPDL	ζ	$[0.05, 0.25, 0.5]$	0.05
SPDL	ϵ	$[0.1, 0.8]$	0.8
Setter-Solver	n	$[50, 100, 200, 300]$	100
Setter-Solver	δ	$[0.005, 0.01, 0.05, 0.1]$	0.05
Setter-Solver	n_{HIDDEN}	$[64, 128, 256, 512]$	128

H.9 Environment details

In this section, we give details about our two environments, their PCG algorithm, as well as some analysis about their task space. Note that our two environments follow the OpenAI Gym's interface and provide after each step, in addition of the usual information (observation, reward, and whether the episode terminated), a binary value set to 1 if the cumulative reward of the episode reached 230. Additionally, we provide extra information and videos of our environments and embodiments, as well as policies learned at http://developmentalsystems.org/TeachMyAgent/.

H.9.1 Stump Tracks

We present the Stump Tracks environment, an extended version of the environment introduced by Portelas et al. (2020a). We only use two of the initially introduced dimensions of the procedural generation of task: stumps' height μ_s and spacing Δ_ss. As in Portelas et al. (2020a), μ_s is used as the mean of a Gaussian distribution with standard deviation 0.1. Each stump has thus its height sampled from this Gaussian distribution and is placed at distance Δ_s from the previous one. We bound differently this task space depending on the experiment we perform, as explained in Section H.10.

We kept the same observation space with 10 values indicating distance of the next object detected by lidars, head angle and velocity (linear and angular), as well as information from the embodiment (angle and speed of joints and also whether the lower limbs have contact with the ground). For information concerning the embodiment, the size of observation depends on the embodiment, as the number of joints varies (see below in H.9.3). We also kept the action space controlling joints with a torque.

H.9.2 Parkour

We introduce the Parkour, a Box2D parkour track inspired by the Stump Tracks and the environment introduced in Wang et al. (2020). It features different milieus in a complex task space.

Procedural generation

CPPN-encoded terrain – First, similarly to the Stump Tracks, our Parkour features a ground (that has the same length as the one in Stump Tracks) where the agent starts at the leftmost side and has to reach the rightmost side. However, this ground is no longer flat and rather, as in Wang et al. (2020), generated using a function outputted by a neural network called CPPN (Stanley, 2007). This network takes in input a x position and outputs the associated y position of the ground. Using this, one can slide the CPPN over the possible x positions of the track in order to obtain the terrain. This method has the advantage of being able to easily generate non-linear and very diverse terrains as shown in Wang et al. (2020), while being light and fast to use as this only needs inference from the network. While CPPNs are usually used in an evolutionary setup where the architecture and weights are mutated, we chose here to rather initialize an arbitrary architecture and random weights and keep them fixed. For this architecture, we chose to use a four layers MLP with 64 units per layer and an alternation of TanH and Softplus activations (except for the output head which uses a linear activation) inspired by Ha (2016). Weights were sampled from a Gaussian distribution with mean 0 and standard deviation of 1. In addition of its x input, we added to our network three inputs that are set before generating the terrain as parameters controlling the generation. This vector θ of size 3 acts in a similar way as noise vector does in GANs for instance. Its size was chosen such that it allows for analysis of the generation space and maintains the overall task space's number of dimensions quite small. As for the parameters in Stump Tracks, we bounded the space of values an ACL method could sample in θ. For this, we

provide three hand-designed setups (easy, medium and hard) differing from the size of the resulting task space and the amount of feasible tasks in it (see Section H.10.4).

Moreover, in addition of the y output of the ground, we added another output head y_c in order to create a ceiling in our tracks. As in Stump Tracks, the terrain starts with a flat startpad region (with a fixed distance between the ground and the ceiling) where the agent appears. Once $Y = (y_i)_{i \in X}$ and $Y_c = (y_{c_i})_{i \in X}$ generated by the CPPN, with X all the possible x positions in the track, we align them to their respective startpad:

$$y_i = y_i + startpad_g - y_0 \; \forall i \in Y$$

$$y_{c_i} = y_{c_i} + startpad_c - y_{c_0} \; \forall i \in Y_c$$

with $startpad_g$, $startpad_c$ being respectively the y position of the ground startpad and ceiling startpad, and y_0, y_{c_0} respectively the first y position of the ground and the ceiling outputted by our CPPN.

Using this non-linear generator (i.e., CPPN) allows us to have an input space where the difficulty landscape of the task space is rugged. Indeed, in addition of generating two non-linear functions for our ground and ceiling, the two latter can cross each other, creating unfeasible tasks (see figure H.6). Additionally, our CPPN also makes the definition of prior knowledge over the input space more complex, as shown in figure H.5.

Finally, as shown in figure H.6, we smoothed the values of Y and Y_c by a parameter δ ($= 10$ in the training distribution) in order to make the roughness of the terrains adapted to our embodiments.

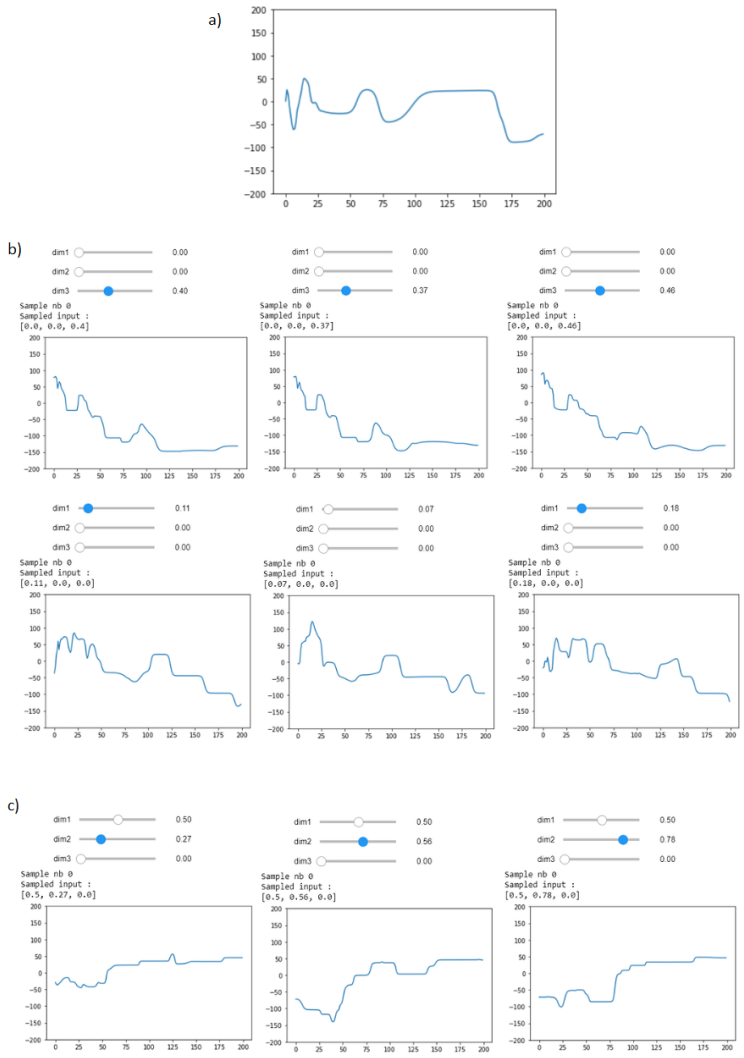


Figure H.5: Overview of the input space of θ. First, in a) one can see the function generated when all the values of the input vector are set to zero. Secondly, in b) we can see that small changes over the space lead to similar functions and that big changes lead to very different results, showing that local similarity is maintained over the task space. Finally, c) shows how the difficulty landscape of θ can be rugged, as moving along the second dimension leads to terrains having a very different difficulty level.

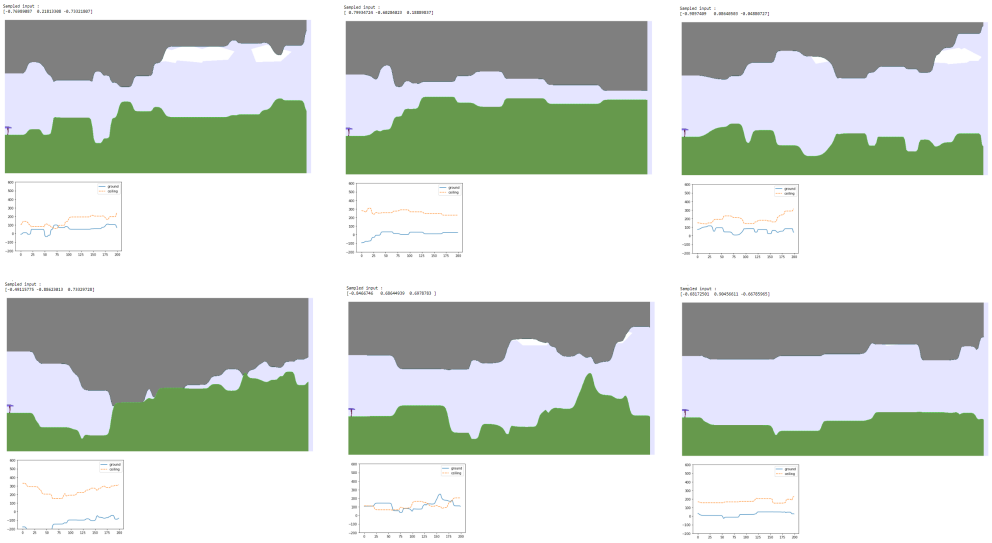


Figure H.6: Here are some examples of generated tasks in the Parkour environment. While most of them seem too hard for a classic bipedal walker, the bottom left task is a good example of an unfeasible task, no matter which embodiment is used.

Creepers – Once the terrain is generated, we add what we call "creepers". Similarly to the stumps, we create objects at distance Δ_c from one another and of height sampled using a Gaussian distribution of mean μ_c and standard deviation 0.1 (the width can also be controlled but was fixed to 0.25 in our experiments). However, creepers are not obstacles for agents as stumps but rather graspable objects that embodiments can go through. Moreover, even though not used in our experiments, we provide the possibility to make creepers more realistic by dividing every creeper in multiple rectangles of height at most 1 linked with a rotating joint. As shown on our website, this creates creepers on which the climbers can swing.

Water – Finally, we added a last dimension to our task space controlling the "water" level. Water is simulated using a rectangle object that the agent can go through and in which physics change (see below). This rectangle's width equals the terrain's width and its height is controlled by a parameter $\tau \in [0; 1]$ with 0 being an arbitrary lower limit the ground can reach and 1 the highest point of the current ceiling (generated by the CPPN for the current task).

Physics

As previously mentioned, we introduced creepers and water along with new physics. First, in order to make our creepers graspable by the agents, we added sensors to the end of limb of certain embodiments (see Section H.9.3 below). Every time one of these sensors enters in contact with a creeper, we look at the action in the action space of the agent that is associated to this sensor. If its value is greater than 0, we create a rotational

joint between the sensor and the creeper at the contact point. As long as this action is greater than 0, the joint remains. As soon as the action goes negative or equals 0, we delete the joint (releasing the agent's limb from the creeper) and start watching again for contact. Note that, in order to better see whether the agent grasps a creeper, we color its sensors in red when a joint exists and in yellow otherwise (see our website). Additionally, in order to help the learning agent, we also make the ceiling graspable.

Secondly, concerning the water, we simulated a buoyancy force when an object enters water given its density compared to the water's density (set to 1). In addition, we implemented a "drag" and a "lift" force that simulate the resistance applied when an object moves in water and slows down the movement. Finally, we added a "push" force applied to an object having an angular velocity. This force simulates the fact that applying a torque to a rotational joint makes the object attached to the joint "push" the water and move (i.e., have a linear force applied). With these forces, we were able to simulate in a simplified way some physics of water, which resulted in very natural policies learned from our agents (see our website).

Finally, we simulated the fact that each embodiment is suited for one (or several) milieu, creating types of agents. Indeed, we first consider swimming agents that die (i.e., the actions sent by the Deep RL student to the environment no longer have effects on the motors of the embodiment) after spending more than 600 consecutive steps outside water. On the contrary, the two other types, climbers and walkers, cannot survive underwater for more than 600 consecutive steps. Both walkers and swimmers are allowed to have collisions with their body (including their head in the Parkour), whereas climbers are not allowed to touch the ground with any part of their body. Note that, while walkers appear with their legs touching the ground, swimmers appear a bit above the ground and climbers appear with all of their sensors attached to the ceiling (see figure H.1).

All of these physics introduce the fact that an ACL teacher has to propose tasks in the right milieu for the current embodiment (i.e., mostly underwater for swimmers so that they do not die, with creepers and a ceiling high enough for climbers so that they do not touch the ground or die in water and with no water for walker so that they do not drown) in order to make the student learn.

Observation and action space

As in Stump Tracks, the agent is rewarded for moving forward and penalized for torque usage. An episode lasts 2000 steps unless the agent reaches the end of the track before or if a part of its body touches the ground if the embodiment is a climber. We also reused the 10 lidars per agent that were used in the Stump Tracks with all the lidars starting from the center of the head of the morphology. However, we modified them such that three configurations of covering exist (see the three tasks shown in figure H.1):

- 90° from below the agent to ahead of it (used by walkers, as in Stump Tracks)
- 180° from below the agent to above it (used by swimmers)
- 90° from ahead of the agent to above it (used by climbers)

Moreover, in addition of the distance to the next object detected by each lidar, we added an information concerning the type of object detected by the lidar (-1 if water, 1 if creeper, 0 otherwise) such that the agent knows whether the object detected is an obstacle or can be passed through. Note also that once their origin point overlaps an object (e.g., water), lidars no longer detect it. Hence the lidars of an agent underwater no longer detect water (which would have made lidars useless as they would have only detected water). Therefore, in order to inform the Deep RL student whether the embodiment is underwater, we added an observation that is set to 1 if the agent's head is under the water level and 0 otherwise. Similarly, we added a binary observation telling whether the agent is dead or not (i.e., the actions we send to its motors no longer have impact). In addition, we kept the same information concerning the agent's head as in Stump Tracks (angle, linear velocity and angular velocity) as well as observations for each motor (angle and speed of joint as well as contact information for some of the attached limb). Finally, we added two binary observations per sensor (if the agent has sensors) telling whether the sensor has contact with a graspable surface and whether it is already attached with a joint. Without considering the information about motors and sensors which depend on the morphology, all of the information listed above create an observation vector of size 26. Note that, additionally, we provide the information to the teacher at each step whether the cumulative reward of the episode has reached 230 for the users using a binary reward.

Finally, for the action space, we kept the same behaviour as the one used in Stump Tracks (i.e., each agent has motors which are controlled through a torque value in $[-1; 1]$). Moreover, we added an action in $[-1; 1]$ per sensor for climbers to say whether this sensor must grasp (if it has contact with a graspable surface) or release.

H.9.3 Morphologies

We included in our benchmark the classic bipedal walker as well as its two modified versions introduced in Portelas et al. (2020a): the short bipedal and the quadrupedal. For these three agents, we kept in their implementation the additional penalty for having an angle different than zero on their head, which was already in Portelas et al. (2020a). Additionally, we created new walkers such as the spider or the millipede shown in figure H.1. See our repository and website for the exhaustive list of embodiments we provide.

We introduce another type of morphologies: climbers. We propose two agents: a chimpanzee-like embodiment, as well as its simplified version without legs (reducing the action space to simplify the learning task). These two agents have two arms with two sensors at their extremity allowing them to grasp creepers or the ceiling.

Both walkers and climbers have a density of 1 on their legs and arms, and a density of 5 on their body and head, making them simply "sink" in water.

Finally, we created swimming morphologies with each of their body part having the same density as the water, making them in a zero-gravity setting when fully underwater. We propose a fish-like embodiment (see figure H.1) with a fin and a tale that can wave its body to move (as well as moving its fin).

Note that we also included an amphibious bipedal walker allowed to survive both underwater and outside water. This gave interesting swimming policies as shown on our

website (http://developmentalsystems.org/TeachMyAgent/).

H.10 Experimental details

In this section, we give details about the setups of our experiments.

H.10.1 Deep RL Students

We used the 0.1.1 version of OpenAI Spinningup's implementation of SAC that uses Tensorflow, as in Portelas et al. (2020a). We modified it such that a teacher could set a task at each reset of the environment. We also kept the same hyperparameters as the ones used in Portelas et al. (2020a):

- A two layers feedforward network with 400/300 units per hidden layer (ReLU activation) for both the value and policy network (using TanH activation on the output layer for the latter)
- An entropy coefficient of 0.005
- A learning rate of 0.001
- A mini-batch update every 10 steps using 1000 randomly sampled experiences from a buffer of size 2 million

For PPO, we used OpenAI Baselines' (Tensorflow) implementation. We used the same two-layer neural network as in SAC for the policy and value networks (which share weights). We modified the runner sampling trajectories from the environment in order to use a single synchronous runner instead of multiple asynchronous ones. We used the environment's wrappers proposed in the OpeanAI Baselines' implementation to clip the actions and normalize the observations and rewards. We added a test environment (as well as a teacher that sets tasks) to test the agent's performance every 500000 steps (as done with SAC). We also normalize the observations and rewards in this test environment using the same running average as the one used in the training environment, so that agent does not receive different information from both environments. We send to the teacher and monitor the original values of reward and observation sent by the environment instead of normalized ones. We set the λ factor of the Generalized Advantage Estimator to 0.95, the clipping parameter ϵ to 0.2 and the gradient clipping parameter to 0.5. Finally, we tuned the following hyperparameters using a grid-search on Stump Tracks for 10 million steps with stumps' height and spacing respectively in $[0; 3]$ and $[0; 6]$:

- Size of experiences sampled between two updates: 2000
- Number of epochs per update: 5
- Learning rate: 0.0003
- Batch size: 1000

- Value function coefficient in loss: 0.5
- Entropy coefficient in loss: 0.0

Note that for both our Deep RL students, we used $\gamma = 0.99$.

H.10.2 General experimental setup

We call an experiment the repetition, using different seeds, of the training of a Deep RL student for 20 million steps using tasks chosen at every reset of the environment by a selected ACL teacher. The seed is used to initialize the state of random generators used in the teacher, Deep RL student and environment. We provide to the teacher the bounds (i.e., a *min* and *max* value for each dimension) of the task space before starting the experiment. The Deep RL student then interacts with the environment and asks the ACL teacher to set the task (i.e., a vector controlling the procedural generation) at every reset of the environment. Once the episode ended, the teacher received either the cumulative reward or a binary reward (set to 1 if the episodic reward is greater than 230) for GoalGAN and Setter-Solver. Teachers like SPDL can additionally access the information sent by the environment at every step, allowing non-episodic ACL methods to run in our testbed.

Every 500000 steps of the Deep RL student in the environment, we test its performance on 100 predefined tasks (that we call test set). We monitor the episodic reward obtained on each of these tasks. We also monitor the average episodic reward obtained on the tasks seen by the student during the last 500000 steps. We ask the teacher to sample 100 tasks every 250000 steps of the Deep RL student and store these tasks to monitor the evolution of the generated curriculum (see at `http://developmentalsystems.org/TeachMyAgent/`). For this sampling, we use the non-exploratory part of our teachers (e.g., ALP-GMM always samples from its GMM or ADR never sets one value to one of its bounds) and do not append these monitoring tasks to the buffers used by some teachers to avoid perturbing the teacher's process.

In our experiments, we were able to run 8 seeds in parallel on a single Nvidia Tesla V100 GPU. In this setup, evaluating one ACL method requires approximately (based on ALP-GMM's wall-clocktime):

- 4608 GPU hours for all skill-specific experiments with 32 seeds.
- 168 GPU hours for the 48 seeded Parkour experiment.

Running both experiments would require 4776 GPU hours, or 48 hours on 100 Nvidia Tesla V100 GPUs. Users with smaller compute budgets could reduce the number of seeds (e.g., divide by 3) without strong statistical repercussions.

H.10.3 Stump Tracks variants

We used the Stump Tracks environment to create our challenge-specific comparison of the different ACL methods. We leveraged its two-dimensional task space (stumps' height

and spacing) to create experiments highlighting each of the 6 challenges listed in Section H.1. Each experiment used 32 seeds.

Test sets

We used the same test set in all our experiments on Stump Tracks in order to have a common test setup to compare and analyze the performance of our different ACL methods. This test set is the same as the one used in Portelas et al. (2020a) with 100 tasks evenly distributed over a task space with $\mu_s \in [0; 3]$ and $\Delta_s \in [0; 6]$.

Experiments

In the following paragraphs, we detail the setup of each of our experiments used in the challenge-specific comparison.

Expert knowledge setups – We allow three different amounts of prior knowledge about the task to our ACL teachers:

- *No expert knowledge*
- *Low expert knowledge*
- *High expert knowledge*

First, in the *No expert knowledge* setup, no prior knowledge concerning the task is accessible. Hence, no reward mastery range (ADR, GoalGAN, and Setter-Solver) is given. Additionally, no prior knowledge concerning the task space like regions containing trivial tasks for the agent (e.g., for ADR or SPDL's initial distribution) or subspace containing the test tasks (e.g., for SPDL's target distribution) are known. However, we still provide these two distributions using the following method:

- *Initial distribution*: we sample the mean $\mu_{INITIAL}$ of a Gaussian distribution uniformly random over the task space. We choose the variance of each dimension such that the standard deviation over this dimension equals 10% of the range of the dimension (as done when expert knowledge is accessible).
- *Target distribution*: we provide a Gaussian distribution whose mean is set to the center of each dimension and standard deviation to one fourth of the range of each dimension (leading to more than 95% of the samples that lie between the min and max of each dimension). This choice of target distribution was made to get closer to our true test distribution (uniform over the whole task space), while maintaining most of the sampled tasks inside our bounds. However, it is clear that this target distribution is not close enough to our test distribution to make SPDL proposing a good curriculum and lead to an agent learning an efficient policy to perform well in our test set. As mentioned in Section H.5 and Section H.8, using a Gaussian target distribution is not suited to our setup and would require modifications to make the target distribution match our true test distribution.

Hence in this setup, only ALP-GMM, RIAC, Covar-GMM and SPDL (even though its initial and target distribution do not give insightful prior knowledge) can run.

In the *Low expert knowledge* setup, we give access to the reward mastery range. Therefore, GoalGAN, Setter-Solver, and ADR can now enter the comparison. The initial distribution is still randomly sampled as explained above. It is used by GoalGAN to pre-train its GAN at the beginning of the training process, but also by ADR, which starts with a single example being $\mu_{INITIAL}$.

Finally, for the *High expert knowledge* setup, we give access to the information about regions of the task space. While the standard deviation of the initial distribution is still calculated in the same way (i.e., 10% of the range of each dimension), we set $\mu_{INITIAL}$ to $[0; 6]$, with the values being respectively μ_s and Δ_s. Hence, ADR now uses the task $[0; 6]$ as its initial task, and GoalGAN pre-trains its GAN with this distribution containing trivial tasks for the walking agent (as stumps are very small with a large spacing between them). SPDL also uses this new initial distribution, but keeps the same target distribution as we could not provide any distribution matching our real test distribution (i.e., uniform). Note that, as mentioned in Section H.8, ALP-GMM and Covar-GMM use this initial distribution in their bootstrap phase in this setup.

Mostly unfeasible task space – In this experiment, we use SAC with a classic bipedal walker. We consider stumps with height greater than 3 impossible to pass for a classic bipedal walker. Hence, in order to make most of the tasks in the task space unfeasible, we use in this experiment $\mu_s \in [0; 9]$ (and do not change $\Delta_s \in [0; 6]$) such that almost 80% of the tasks are unfeasible.

Mostly trivial task space – Similarly, we use in this experiment $\mu_s \in [-3; 3]$ (the Stump Tracks environments clips the negative values with $\mu_s = \max(0, \mu_s)$). Hence 50% of the tasks in the task space will result in a Gaussian distribution used to generate stumps' height with mean 0. We also use SAC with a classic bipedal walker.

Forgetting students – We simulate the catastrophic forgetting behaviour by resetting all the variables of the computational graph of our Deep RL student (SAC here) as well as its buffers every 7 million steps (hence twice in a training of 20 million steps). All variables (e.g., weights, optimizer's variables...) are reinitialized the same way they were before starting the training and the experience buffer used by SAC is emptied. Note that we also use the classic bipedal walker as embodiment and did not modify the initial task space ($\mu_s \in [0; 3]$ and $\Delta_s \in [0; 6]$).

Rugged difficulty landscape – In order to create a rugged difficulty landscape over our task space, we cut it into 4 regions of the same size and shuffle them (see algorithm 3). The teacher then samples tasks in the new task space using interpolation (see algorithm 4), which is now a discontinuous task space introducing peaks and cliffs in the difficulty landscape. While the cut of regions is always the same, the shuffling process is seeded at each experiment.

Algorithm 3 Cutting and shuffling of the task space.

Input: Number of dimensions $\mathcal{D}$, bounds $(min_i)_{i\in[\mathcal{D}]}$ and $(max_i)_{i\in[\mathcal{D}]}$, number of cuts k
for $d \in [\mathcal{D}]$ **do**
  Initialise arrays $\mathcal{O}_d, \mathcal{S}_d$
  $size \leftarrow |max_d - min_d|/k$
  for $j \in [k]$ **do**
    Store pair $(min_d + j * size, min_d + (j+1) * size)$ in $\mathcal{O}_d$ and $\mathcal{S}_d$
  Shuffle order of pairs in $\mathcal{S}_d$

Algorithm 4 Interpolate sampled task in the shuffled task space.

Input: Number of dimensions $\mathcal{D}$, task vector $\mathcal{T}$, number of cuts k
Initialise the vector $\mathcal{I}$ of size $\mathcal{D}$
for $d \in [\mathcal{D}]$ **do**
  for $j \in [k]$ **do**
    Get pair o_j in $\mathcal{O}_d$
    Initialize l with the first element of o_j
    Initialize h with the second element of o_j
    if $l \leq \mathcal{T}_d \leq h$ **then**
      Get pair s_j in $\mathcal{S}_d$
      Get β as the interpolation of $\mathcal{T}_d$ from the interval o_j to the interval s_j
      Set $\mathcal{I}_d = \beta$
      End the loop
Return $\mathcal{I}$

Robustness to diverse students – Finally, in order to highlight the robustness of an ACL teacher to diverse students, we perform 4 experiments (each with 32 seeds) and then aggregate results. We use the initial task space of Stump Tracks, but use both PPO and SAC, as well as two different embodiments: the short bipedal walker and the spider. Each embodiment is used both with PPO and SAC (hence 2 experiments per embodiment and thus a total of 4 experiments). We then aggregate the 128 seeds into a single experiment result.

H.10.4 Parkour experiments

We perform a single experiment in the Parkour environment using 48 seeds. Among these seeds, 16 use a classic bipedal walker, 16 a chimpanzee, and 16 a fish embodiment. We set the bounds of the task space to the following:

- CPPN's input vector $\theta \in [-0.35, 0.05] \times [0.6, 1.0] \times [-0.1, 0.3]$
- Creepers' height $\mu_c \in [0; 4]$
- Creepers' spacing $\Delta_c \in [0; 5]$

- Water level $\tau \in [0; 1]$

Note that the above CPPN's input space is considered as our medium one. We also provide the easy space ($\theta \in [-0.25, -0.05] \times [0.8, 1.0] \times [0.0, 0.2]$) as well as the hard one ($\theta \in [-1, 1] \times [-1, 1] \times [-1, 1]$). Both the easy and medium spaces were designed from our hard task space. Their boundaries were searched such that the task space contains feasible tasks while maintaining diverse terrains. They differ in their ratio between feasible and unfeasible tasks.

Test sets

Unlike in the Stump Track experiments, we needed in the Parkour environment different test sets as our three embodiments (i.e. bipedal walker, chimpanzee, fish) are not meant to act and live in the same milieu (e.g. swimmers do not survive in tasks not containing water). Therefore, creating a test set composed of tasks uniformly sampled would not allow for assessing the performance of the current embodiment. Hence, we made three different test sets for Parkour, each consisting of 100 tasks. As the task space previously defined is mainly composed of unfeasible tasks for any embodiment, we hand-designed each of the three test sets with the aim of showcasing the abilities of each morphology type, as well as showing the ability of the learned policy to generalize. Each test set has 60 tasks that belong to the training task space and 40 out-of-distribution tasks (using tasks outside the medium CPPN's input space as well as smoothing values different than 10 for the δ parameter). They also share the same distribution between easy (1/3), medium (1/3), and hard (1/3) tasks. We chose each task so that it seemed possible given the physical capacities of our embodiments. See figure H.7 for some examples of the test tasks.

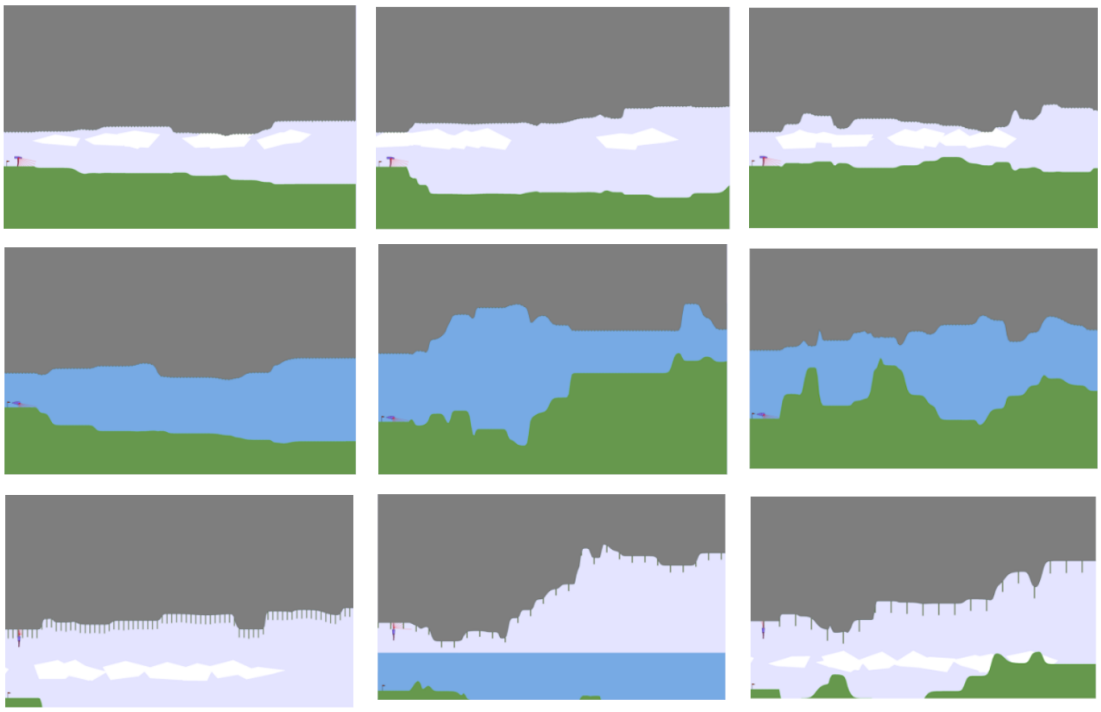


Figure H.7: We show some examples of the tasks belonging to our Parkour's test sets. The first line shows tasks from the walkers' test set, the second line from the swimmers' one, and finally, the last line for climbers.

H.11 Additional results

In this section, we provide additional results on experiments presented in Section H.5 as well as case studies. As mentioned in Section H.10, we monitor both the episodic reward on each of the test tasks and the average episodic reward on training tasks every 500000 steps for each seed. We use the episodic reward on test tasks to calculate our percentage of "mastered" tasks metric, which calculates the percentage of tasks on which the agent obtained an episodic reward greater than 230. Additionally, we compare two algorithms in an experiment using Welch's t-test between their population of seeds.

H.11.1 Original Stump Tracks

We trained our SAC student for 20 million steps on the original Stump Track task space (i.e., $\mu_s \in [0; 3]$ and $\Delta_s \in [0; 6]$) with each teacher and each expert knowledge setup. We used the best performance of each prior knowledge configuration as a baseline indication in figures H.1 and H.9. As in our challenge-specific experiments, we used 32 seeds as well as the same test set of 100 evenly distributed tasks. Results can be found in table H.3.

Table H.3: Percentage of mastered tasks after 20 million steps on the original Stump Tracks challenge (i.e., $\mu_s \in [0;3]$ and $\Delta_s \in [0;6]$). Results shown are averages over 32 seeds along with the standard deviation. We highlight the best results in bold, which then acted as an upper baseline indication in the challenge-specific comparisons.

Algorithm	No EK	Low EK	High EK
ADR	-	24.1 (± 20.8)	43.4 (± 7.2)
ALP-GMM	**52.1** (± 5.9)	**47.1** (± 13.9)	49.3 (± 5.9)
Covar-GMM	43.0 (± 9.1)	40.25 (± 16.5)	45.2(± 10.1)
GoalGAN	-	29.9 (± 26.2)	**51.9** (± 7.3)
RIAC	40.5 (± 8.4)	39.6 ((± 11.2)	42.2 (± 5.4)
SPDL	20.8 (± 19.4)	18.5 (± 20.8)	34.0 (± 10.6)
Setter-Solver	25.3 (± 10.7)	36.6(± 10.2)	37.4 (± 9.8)

H.11.2 Challenge-specific comparison

Overall results

We show the performance after 20 million steps of each ACL teacher on each challenge. Results are gathered in tables H.4, H.5, and H.6, as well as in figure H.8, where we show the results of Welch's t-test between all methods on every challenge.

Table H.4: Percentage of mastered tasks after 20 million steps with **no** prior knowledge in each challenge. Results shown are averages over all seeds along with the standard deviation. We highlight the best results in bold.

Algorithm	Mostly unf.	Mostly triv.	Forgetting stud.	Rugged dif.	Diverse stud.
Random	18.0 (± 10.5)	22.2 (± 15.2)	27.8 (± 14.6)	30.3 (± 7.7)	22.3 (± 11.5)
ALP-GMM	**42.8** (± 6.6)	**43.7** (± 6.0)	**42.1** (± 6.9)	**42.5** (± 4.8)	31.5 (± 9.2)
Covar-GMM	39.0 (± 9.9)	32.7 (± 16.0)	31.3 (± 16.2)	39.4 (± 7.4)	**32.3** (± 10.6)
RIAC	22.1 (± 14.5)	20.0 (± 10.9)	36.8 (± 6.9)	36.4 (± 7.9)	25.9 (± 11.3)
SPDL	6.4 (± 10.2)	15.3 (± 9.9)	10.4 (± 12.9)	19.3 (± 16.2)	8.9 (± 14.4)

Table H.5: Percentage of mastered tasks after 20 million steps with **low** prior knowledge in each challenge. Results shown are averages over all seeds along with the standard deviation. We highlight the best results in bold.

Algorithm	Mostly unf.	Mostly triv.	Forgetting stud.	Rugged dif.	Diverse stud.
Random	18.0 (± 10.1)	18.0 (± 7.1)	27.8 (± 14.6)	30.3 (± 7.7)	22.3 (± 11.5)
ADR	7.8 (± 17.9)	22.2 (± 15.2)	21.2 (± 21.2)	17.0 (± 19.6)	15.6 (± 19.1)
ALP-GMM	**43.5** (± 13.0)	**43.0** (± 9.0)	**41.6** (± 12.5)	**44.2** (± 7.1)	31.3 (± 9.4)
Covar-GMM	31.2 (± 16.8)	42.0 (± 8.4)	31.5 (± 18.4)	34.3 (± 10.7)	**32.1** (± 9.6)
GoalGAN	12.7 (± 16.2)	38.4 (± 16.1)	9.3 (± 15.8)	34.7 (± 19.1)	16.2 (± 17.5)
RIAC	20.5 (± 14.0)	21.3 (± 8.8)	34.3 (± 12.5)	38.3 (± 11.3)	26.0 (± 11.7)
SPDL	6.7 (± 10.2)	17.9 (± 12.2)	10.6 (± 12.2)	18.1 (± 15.8)	9.2 (± 14.2)
Setter-Solver	25.3 (± 10.7)	35.5 (± 8.9)	33.9 (± 12.5)	31.6 (± 11.3)	25.4 (± 9.0)

Table H.6: Percentage of mastered tasks after 20 million steps with **high** prior knowledge in each challenge. Results shown are averages over all seeds along with the standard deviation. We highlight the best results in bold.

Algorithm	Mostly unf.	Mostly triv.	Forgetting stud.	Rugged dif.	Diverse stud.
Random	18.0 (± 10.1)	18.0 (± 7.1)	27.8 (± 14.6)	30.3 (± 7.7)	22.3 (± 11.5)
ADR	45.3 (± 6.7)	32.5 (± 6.2)	39.8 (± 10.8)	17 (± 20.9)	32.3 (± 9.7)
ALP-GMM	**48.4** (± 11.2)	44.3 (± 14.2)	**43.0** (± 9.0)	**42.5** (± 7.3)	29.8 (± 8.8)
Covar-GMM	38.2 (± 11.9)	39.6 (± 10.3)	39.5 (± 12.5)	41.3 (± 7.0)	**32.6** (± 10.2)
GoalGAN	39.7 (± 10.1)	**45.6** (± 13.5)	23.4 (± 19.7)	41.2 (± 12.6)	27.5 (± 9.4)
RIAC	25.2 (± 12.3)	22.1 (± 11.1)	37.7 (± 12.5)	37.7 (± 8.8)	25.8 (± 11.7)
SPDL	19.1 (± 12.5)	22.9 (± 6.9)	12.9 (± 11.2)	31.0 (± 11.2)	15.4 (± 15.1)
Setter-Solver	28.2 (± 9.7)	33.7 (± 10.8)	37.4 (± 8.7)	34.7 (± 8.1)	24.0 (± 9.8)

Figure H.8: Performance of the different teachers at the end of training in every experiment of our challenge-specific comparison. We plot as bars the average percentage of mastered tasks for each ACL method. Additionally, we compare in every experiment all possible couples of teacher methods using Welch's t-test and annotate the significantly different ($p < 0.05$) ones.

Case study: Sample efficiency

In this section, we take a look at the sample efficiency of the different ACL methods using their performance after only 5 million steps. We reuse the same radar chart as in Section H.5 in figure H.9.

Looking at results, one can see the impact of ACL in the mostly unfeasible challenge, as some methods (e.g., ALP-GMM or ADR with high expert knowledge) already reach twice the performance of random after only 5 million steps. This highlights how leveraging a curriculum adapted to the student's capabilities is key when most tasks are unfeasible. On the other hand, when the task space is easier (as in the mostly trivial challenge), Random samples more tasks suited for the current student's abilities, and the impact of Curriculum Learning is diminished.

Having a difficult landscape with ruggedness makes the search for learnable and adapted subspaces harder. Figure H.9 shows that only 5 million steps are not enough, even for teachers such as ALP-GMM or Covar-GMM, theoretically more suited for rugged difficulty landscapes, to explore and leverage regions with high learning progress.

Finally, one can see the strong impact of a well designed initial distribution of tasks in the beginning of learning. Indeed, both ADR and GoalGAN already almost reach their final performance (i.e., the one they reached after 20 million steps shown in figure H.1) after 5 million steps in the *High expert knowledge* setup, as they know where to focus and do not need exploration to find feasible subspaces. Similarly, adding expert knowledge to ALP-GMM increases its performance compared to the no and low expert knowledge setups, helping it focus the bootstrapping process on a feasible region. Leveraging this initial task distribution, GoalGAN obtains the best results in 3/5 challenges after 5 million steps with high expert knowledge. This shows, in addition of the results from Section H.5, that GoalGAN is a very competitive method, especially when it has access to high expert knowledge.

Figure H.9: **Performance of ACL methods measured after 5 million steps.** Results are presented as an order of magnitude of the performance of Random. Performance is defined as the average percentage of mastered test tasks over all 32 seeds. We also provide the same indications (✚) of the best performance (measured after 5 million steps) on the original Stump Tracks experiment as in figure H.1. Finally, we indicate on each axis which method performed significantly better than Random ($p < 0.05$) using colored stars matching each method's color (e.g., ★ for Covar-GMM, ★ for ADR). *EK: Expert Knowledge.*

Case study: On the difficulty of GoalGAN and SPDL to adapt the curriculum to forgetting students

As mentioned in Section H.5, both GoalGAN and SPDL struggled on the forgetting student challenge, no matter the amount of expert knowledge. In order to better understand their behaviour in this challenge, we plot in figure H.10 both the evolution of their percentage of mastered tasks and their average training return. We also add ALP-GMM and ADR (two students that performed well in this challenge) as baselines for comparison. While ADR and ALP-GMM make the student quickly recover from a reset (i.e., the percentage of mastered tasks quickly reaches the performance it had before the reset) and then carry on improving, both GoalGAN and SPDL suffer from resets and do not manage to recover, leading to a poor final performance.

Multiple factors could explain this phenomenon. First, in the case of SPDL, even though the algorithm tries to shift its sampling distribution such that it maximizes the student's performance, the optimization methods also have to minimize the distance to the target distribution, which is a Gaussian spanning over the entire task space. However, resetting the student's policy requires the ACL method to revert to the initial simple task distribution that it proposed at the beginning of training. Such a reverse process might not easily be achievable by SPDL, whose optimization procedure progressively shifts its sampling distribution towards the target one.

Concerning GoalGAN, the performance impact of student resets is most likely due to its use of a buffer of "Goals of Intermediate Difficulty" to train the goal generator. Upon student reset, this buffer becomes partially obsolete, as the student is reinitialized, making it lose all learned walking gaits, i.e., everything must be learned again. This means the goal generator will propose tasks that are too complex for a student who is just starting to learn. Because GoalGAN cannot reset its buffer of "Goals of Intermediate Difficulty" (which would require knowledge over the student's internal state), it impairs its ability to shift to simpler task subspaces quickly.

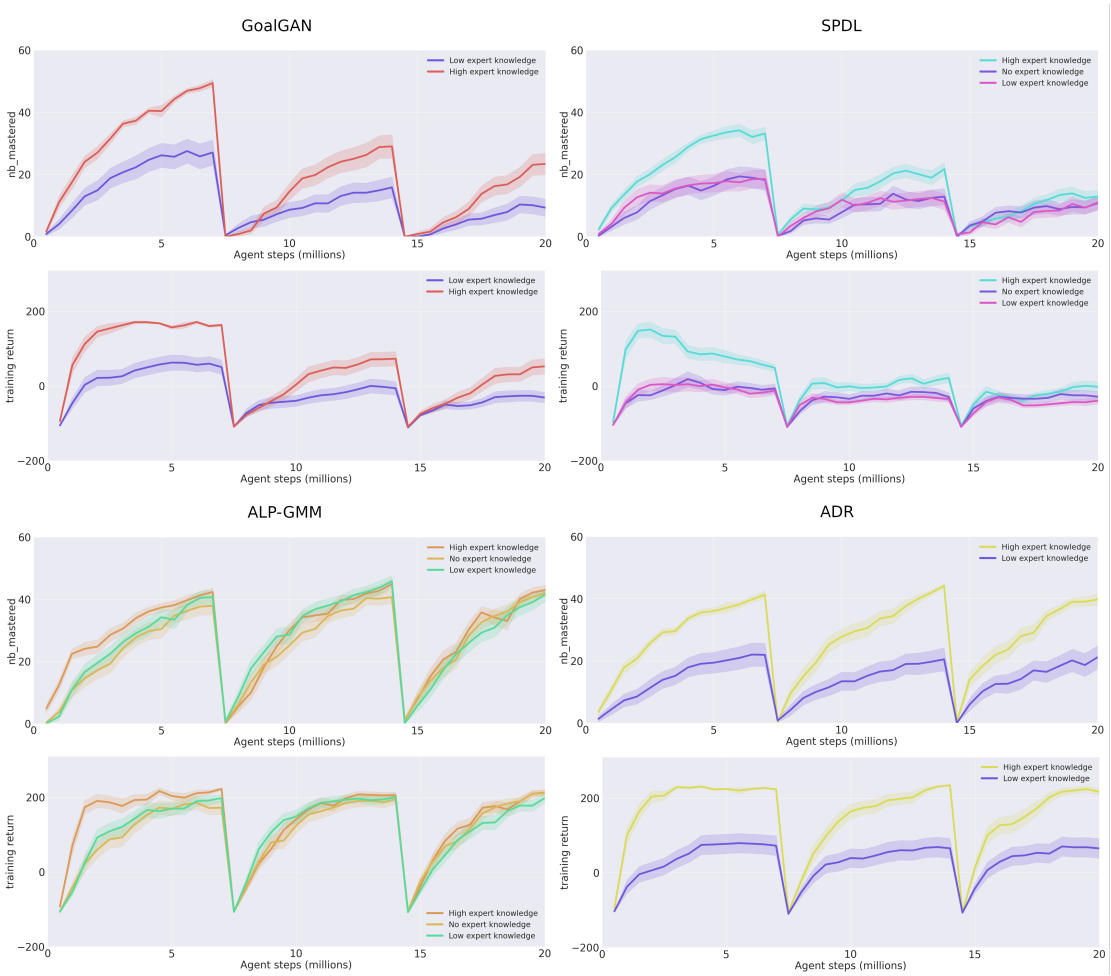


Figure H.10: Percentage of mastered test tasks and average training return of GoalGAN, SPDL, ALP-GMM and ADR on the forgetting student challenge. Curves are averages over 32 seeds along with the standard error of the mean.

Case study: Impact of expert knowledge on ALP-GMM

As mentioned, ALP-GMM is a method that initially did not require any expert knowledge. Moreover, it relies on an exploration (bootstrap) phase to fill its buffer, usually using uniform sampling over the task space. In TeachMyAgent, we provide an extended version of it where we added the possibility of using expert knowledge by

bootstrapping from an initial distribution instead of a uniform distribution. In this case study, we take a look at the impact such a theoretical improvement had on their performance. We focus on the mostly unfeasible and forgetting student challenges, as the first highlighted the most how prior knowledge can help an ACL method (helping it start in a feasible region) and the latter showed interesting results for this case study, in addition of being easy to analyze (as it only uses a bipedal walker on the original task space of the Stump Tracks). We gather these results in figure H.11. Note that both the no and low expert knowledge setups are the same for ALP-GMM, meaning that any difference between their results is only due to variance in both the student's learning and ACL process.

When looking at these results, one can see that the high expert knowledge setup is significantly better than the two other setups at the beginning of the training in both challenges. These results can also be completed by our sample efficiency case study (see figure H.9), showing that adding expert knowledge to ALP-GMM makes it more sample efficient. Then, as training advances, the difference becomes statistically insignificant ($p > 0.05$). Finally, while the final results given in tables H.4, H.5, and H.6 show an improved percentage of mastered tasks in almost all challenges, with a notable difference (at least +5) on the mostly unfeasible challenge, results on the original Stump Tracks experiments (table H.3) show better results with no expert knowledge. It is thus not clear whether adding this prior knowledge to ALP-GMM benefits the whole training instead of just the beginning. Note that similar behaviors were also obtained with Covar-GMM, even though they were not as significant as the ones of ALP-GMM.

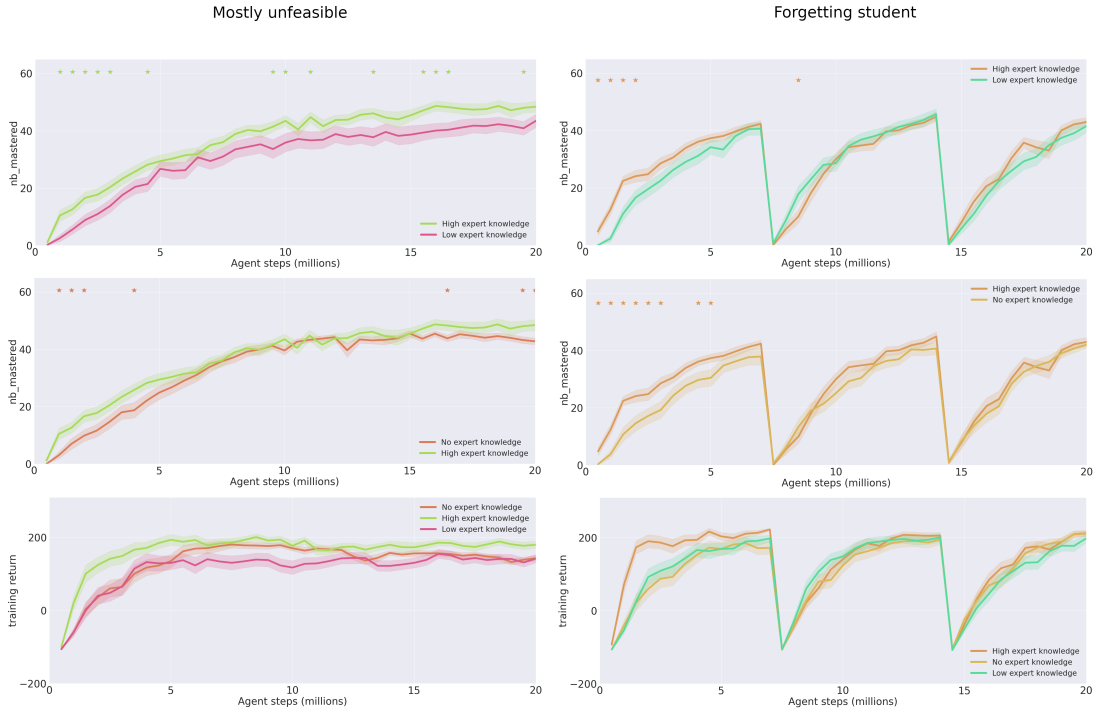


Figure H.11: Percentage of mastered test tasks and average training return of ALP-GMM on both the mostly unfeasible and the forgetting student challenges. Curves are averages over 32 seeds along with the standard error of the mean. We compare the impact of high expert knowledge compared to the two other setups using Welch's t-test and highlight significant ($p < 0.05$) differences with stars.)

Case study: What ADR needs

ADR is a very efficient and light method, that, when all its expert knowledge requirements are satisfied, competes with the best teachers. However, in order to obtain such an efficient behaviour, ADR needs certain conditions that are implied by its construction. First, as explained in Section H.8, ADR starts its process using a single task, and makes the assumption that this latter is easy enough for the freshly initialized student. It then progressively grows its sampling distribution around this task if the student manages to "master" the proposed tasks. While this behaviour seems close to SPDL's, ADR does not have any target distribution to help it shift the distribution even if the students' performances are not good enough. Hence, ADR can get completely stuck if it is initialized on a task lying in a very hard region, whereas SPDL would still try to converge to the target distribution (even though the performance would not be as good as if its initial distribution was set in an easy subspace). Similarly, GoalGAN also uses an initial distribution at the beginning of the training, which, as shown in the results, has a strong impact on the final performance. However, even without it, GoalGAN is still able to reach a decent performance in certain challenges (e.g., mostly trivial), unlike ADR. This observation can also be seen in the Parkour's experiments, where GoalGAN reaches the top 4 while ADR obtains the worst performance. In order to highlight this explanation,

we provide the results of ADR in the mostly unfeasible and mostly trivial challenges in figure H.12, in addition of the clear difference between expert knowledge setups showed by figure H.1 and tables H.4, H.5, and H.6. Using figure H.12, one can see the clear and significant ($p < 0.05$) difference between the two expert knowledge setups.

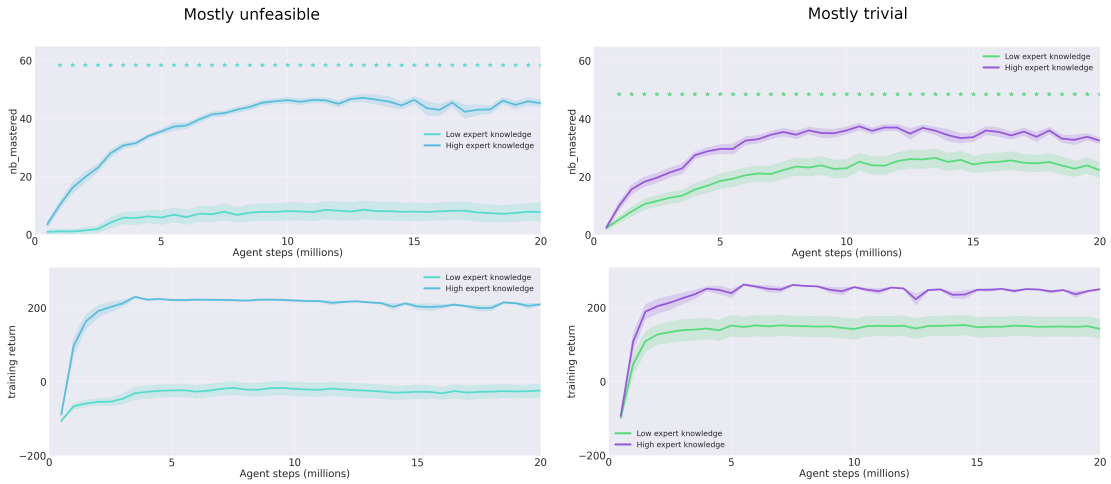


Figure H.12: Percentage of mastered test tasks and average training return of ADR on both the mostly unfeasible and mostly trivial challenges. Curves are averages over 32 seeds along with the standard error of the mean. We compare the impact of high to low expert knowledge using Welch's t-test ($p < 0.05$) and highlight significant differences with stars.)

In addition to an initial task well set using prior knowledge about the task space, ADR needs a difficulty landscape not too rugged to be able to expand and reach regions with high learning progress for the student. Indeed, when looking at its algorithm, one can see that the sampling distribution grows in a certain direction only if the student is able to master the tasks proposed at the edge of the distribution on this direction. If it is not the case (i.e., if this region of the task space is too hard for the current student's capabilities), the sampling distribution will shrink. This simple mechanism makes the strong assumption that if the difficulty is too hard at one edge of the distribution, there is no need to go further, implicitly saying that the difficulty further is at least as hard as the one at the edge. While this works well in the vanilla task space of our Stump Tracks environment (our difficulty is clearly smooth and even monotonically increasing), any task space with a rugged difficulty landscape would make the problem harder for ADR. Indeed, as it reaches a valley in the difficulty landscape surrounded by hills of unfeasible (or too hard for the current student's abilities) tasks, ADR can get stuck. In order to highlight this behaviour, we use our rugged difficulty landscape challenge, where we created a discontinuous difficulty landscape where unfeasible regions can lie in the middle of the task space. Figure H.13 shows how ADR is unable to solve this challenge, no matter the amount of expert knowledge it uses, leading to the worst performance of our benchmark (significantly worse than Random at $p < 0.05$). Note that this issue also happens in our Parkour experiments, as the difficulty of the task space is very rugged

(see Section H.5).

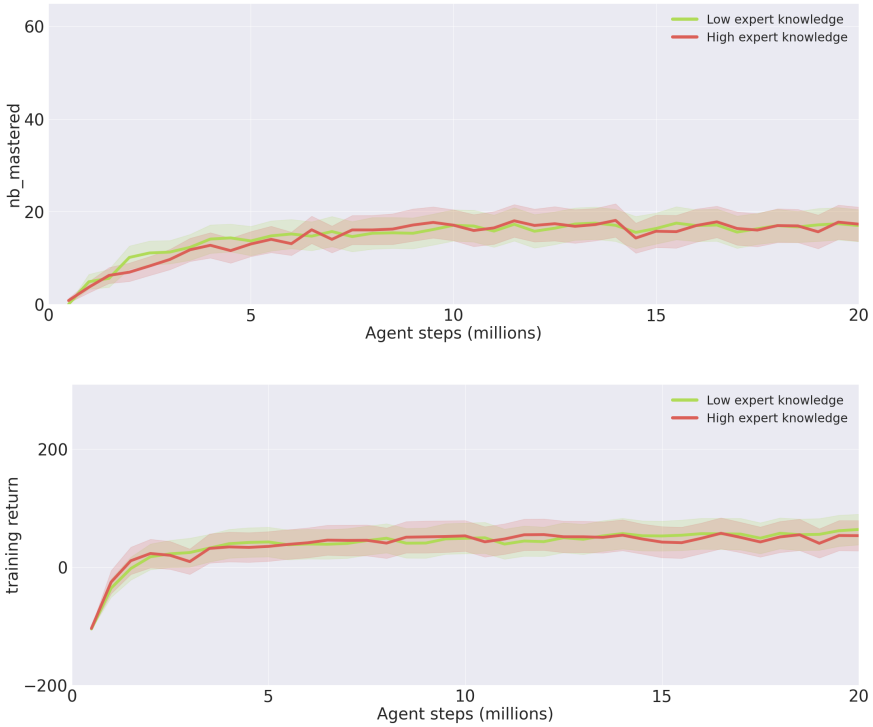


Figure H.13: Percentage of mastered test tasks and average training return of ADR on rugged difficulty landscape challenge. Curves are averages over 32 seeds along with the standard error of the mean.

H.11.3 Parkour

Overall results

In this section, we present the performance of our teacher algorithms on the Parkour track experiments. We present the final results in table H.7 as well as a comparison in figure H.14 using Welch's t-test. We also provide insights concerning the obtained policies at `http://developmentalsystems.org/TeachMyAgent/`. When looking at the overall results, one can see that ALP-GMM is the only teacher performing significantly better than Random throughout training. Covar-GMM's performance are very close to ALP-GMM, as well as RIAC, which obtains very similar results to GoalGAN. While being very close to Random, Setter-Solver's results are not significantly different from ALP-GMM's results by the end of the training. Finally, while SPDL's behavior is very similar to Random, ADR reaches a plateau very soon and eventually obtains significantly worse results than the random teacher.

Table H.7: Percentage of mastered tasks after 20 million steps on the Parkour track. Results shown are averages over 16 seeds along with the standard deviation for each morphology as well as the aggregation of the 48 seeds in the overall column. We highlight the best results in bold.

Algorithm	BipedalWalker	Fish	Climber	Overall
Random	27.25 (± 10.7)	23.6 (± 21.3)	0.0 (± 0.0)	16.9 (± 18.3)
ADR	14.7 (± 19.4)	5.3 (± 20.6)	0.0 (± 0.0)	6.7 (± 17.4)
ALP-GMM	**42.7** (± 11.2)	36.1 (± 28.5)	0.4 (± 1.2)	**26.4** (± 25.7)
Covar-GMM	35.7 (± 15.9)	29.9 (± 27.9)	0.5 (± 1.9)	22.1 (± 24.2)
GoalGAN	25.4 (± 24.7)	34.7 (± 37.0)	0.8 (± 2.7)	20.3 (± 29.5)
RIAC	31.2 (± 8.2)	**37.4** (± 25.4)	0.4 (± 1.4)	23.0 (± 22.4)
SPDL	30.6 (± 22.8)	9.0 (± 24.2)	**1.0** (± 3.4)	13.5 (± 23.0)
Setter-Solver	28.75 (± 20.7)	5.1 (± 7.6)	0.0 (± 0.0)	11.3 (± 17.9)

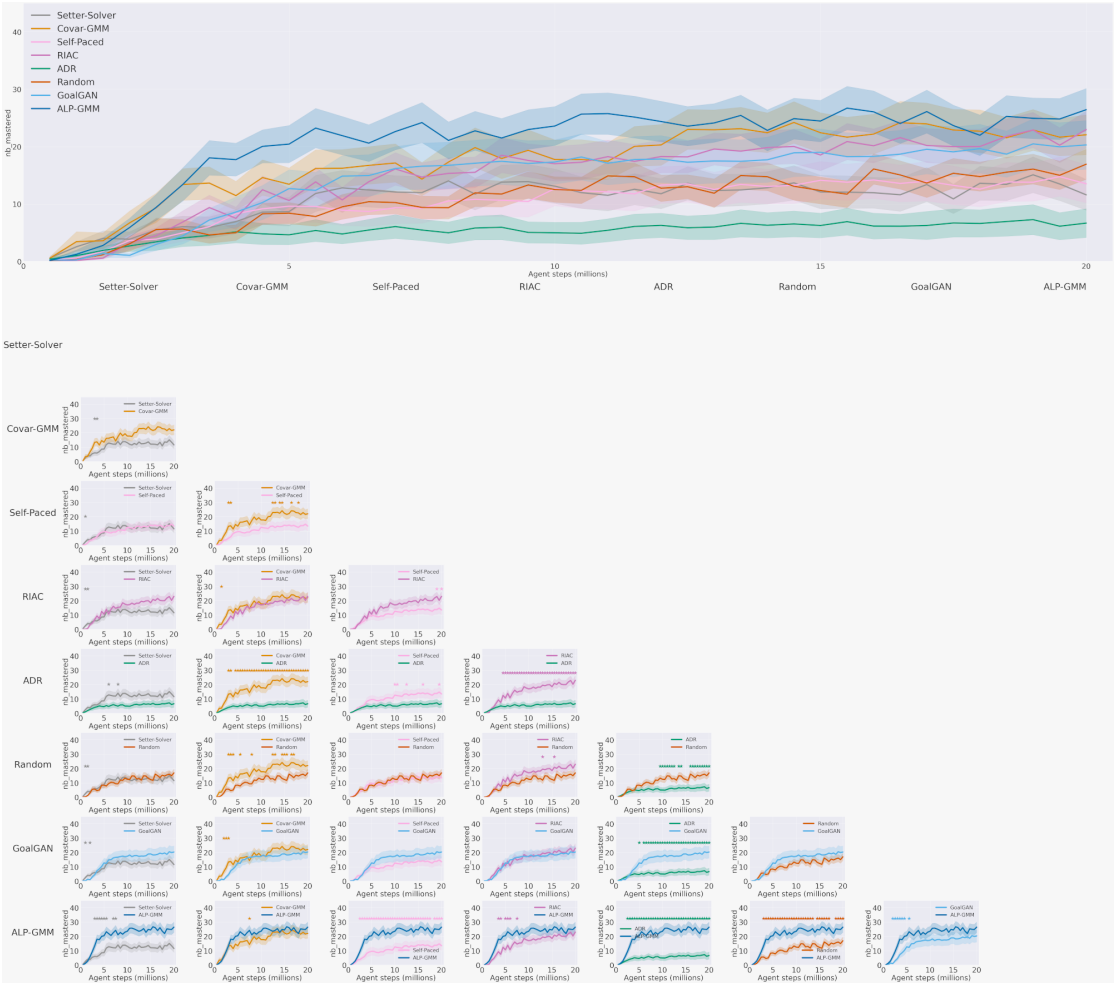


Figure H.14: **Comparison of the ACL methods on the Parkour experiments**. The upper figure shows the average percentage of mastered tasks over the 48 seeds with the standard error of the mean. We then extract all the possible couples of methods and compare their two curves. At each time step (i.e., every 500000 steps), we use Welch's t-test to compare the two distributions of seeds. If a significant difference exists ($p < 0.05$), we add a star above the curves at this time step.

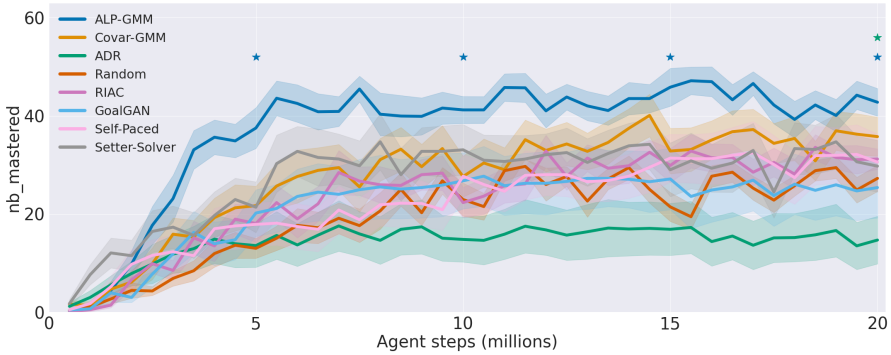


Figure H.15: Average percentage of mastered tasks over 16 seeds using the **bipedal walker** along with the standard error of the mean. We calculate, every 5 million steps, which method obtained statistically different ($p < 0.05$) results from Random and indicate it with a star.

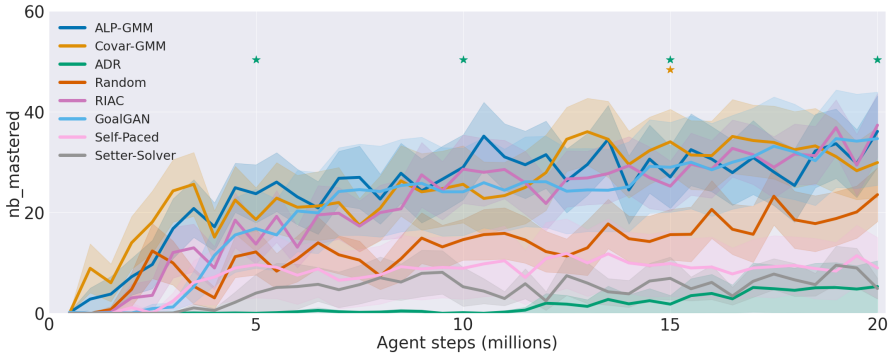


Figure H.16: Average percentage of mastered tasks over 16 seeds using the **fish** along with the standard error of the mean. We calculate, every 5 million steps, which method obtained statistically different ($p < 0.05$) results from Random and indicate it with a star.

Case study: Learning climbing locomotion

As shown in figures H.4 and H.17, none of the ACL methods implemented in TeachMyAgent managed to find a curriculum helping the student to learn an efficient climbing policy and master more than 1% of our test set. While learning climbing locomotion can arguably appear as a harder challenge compared to the swimming and walking locomotion, we present in this case study the results of an experiment using our easy CPPN's input space (see Section H.10.4), as well as no water (i.e., the maximum level is set to 0.2, leading to no tasks with water). Using this, we show that simplifying the task space allows our Random teacher to master more than 6% our test set with its best seed reaching 30% at the end of learning in only 10 million steps. In comparison, our results in the benchmark show a best performance of 1% of mastered tasks (SPDL) with its best seed

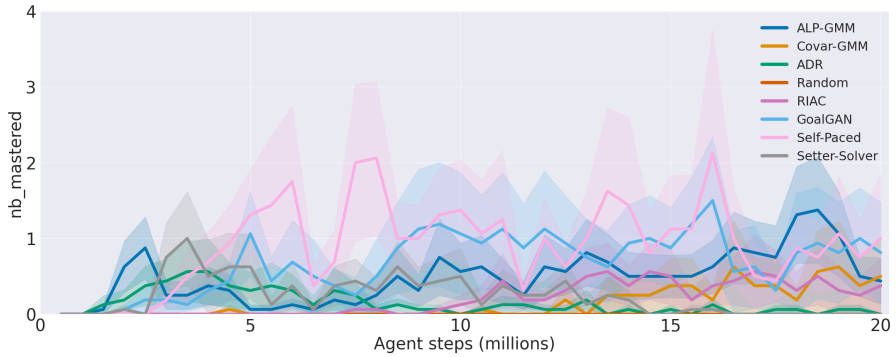


Figure H.17: Average percentage of mastered tasks over 16 seeds using the **chimpanzee** along with standard error of the mean. We calculate, every 5 million steps, which method obtained statistically different ($p < 0.05$) results from Random and indicate it with a star.

reaching only 14% by the end of learning. As this simpler task space contains more feasible tasks, these results show that the poor performance obtained with the chimpanzee embodiment are due to the inability of the implemented ACL algorithms to find feasible subspaces for their student. This also hints at possible better performance by future methods in this totally open challenge of TeachMyAgent. See figure H.18 for the evolution of the percentage of mastered tasks by the Random teacher in this simpler experiment.

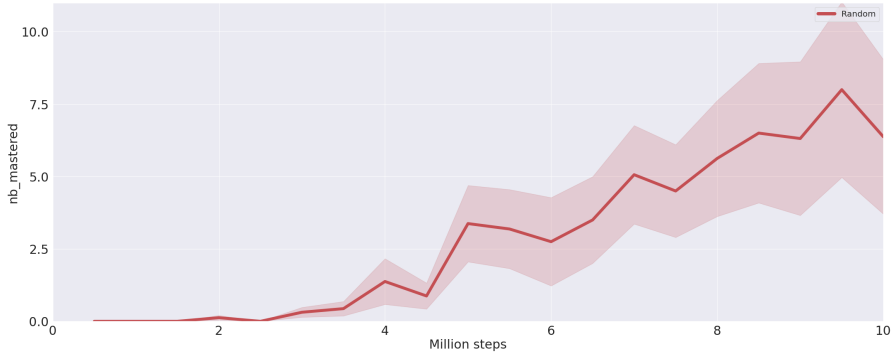


Figure H.18: **Random teacher in the easy CPPN's input space with no water**. Average percentage of mastered tasks over 16 seeds using our chimpanzee embodiment, along with the standard error of the mean.

Bibliography

Abdelghani, R., Murayama, K., Kidd, C., Sauzéon, H., and Oudeyer, P.-Y.
Investigating Middle School Students Question-Asking and Answer-Evaluation Skills When Using ChatGPT for Science Investigation, May 2025.
arXiv:2505.01106 [cs].

Abdou, M., Kulmizev, A., Hershcovich, D., Frank, S., Pavlick, E., and Søgaard, A.
Can Language Models Encode Perceptual Structure Without Grounding? A Case Study in Color.
In Bisazza, A. and Abend, O. (eds.), *Proceedings of the 25th Conference on Computational Natural Language Learning*, pp. 109–132, Online, November 2021. Association for Computational Linguistics.

Abdulhai, M., White, I., Snell, C., Sun, C., Hong, J., Zhai, Y., Xu, K., and Levine, S.
LMRL Gym: Benchmarks for Multi-Turn Reinforcement Learning with Language Models, November 2023.
arXiv:2311.18232 [cs].

Achiam, J. and Sastry, S.
Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning, March 2017.
arXiv:1703.01732 [cs].

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M.
Deep Reinforcement Learning at the Edge of the Statistical Precipice.
In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021.

Aggarwal, P. and Welleck, S.
L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning, March 2025.
arXiv:2503.04697 [cs].

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S.
Back to Basics: Revisiting REINFORCE-Style Optimization for Learning from Human Feedback in LLMs.
In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., and Yan, M.
Do As I Can, Not As I Say: Grounding Language in Robotic Affordances.
arXiv:2204.01691 [cs], April 2022.
arXiv: 2204.01691.

Aissi, M. S., Grislain, C., Chetouani, M., Sigaud, O., Soulier, L., and Thome, N.
VIPER: Visual Perception and Explainable Reasoning for Sequential Decision-Making, March 2025.
arXiv:2503.15108 [cs].

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O.
Grounding Language to Autonomously-Acquired Skills via Goal Generation.
In *International Conference on Learning Representations (ICLR 2021)*. International Conference on Learning Representations (ICLR), 2021.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millicah, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K.
Flamingo: a visual language model for few-shot learning.
In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 23716–23736, Red Hook, NY, USA, November 2022. Curran Associates Inc.

Ammanabrolu, P. and Riedl, M. O.
Situated language learning via interactive narratives.
Patterns, 2(9):100316, September 2021.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W.
Hindsight Experience Replay.
In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B.
A survey of robot learning from demonstration.
Robotics and Autonomous Systems, 57(5):469–483, May 2009.

Arjovsky, M., Chintala, S., and Bottou, L.
Wasserstein Generative Adversarial Networks.

In *Proceedings of the 34th International Conference on Machine Learning*, pp. 214–223. PMLR, July 2017.
ISSN: 2640-3498.

Arora, D. and Zanette, A.
Training Language Models to Reason Efficiently, May 2025.
arXiv:2502.04463 [cs].

Aubret, A., Matignon, L., and Hassas, S.
A survey on intrinsic motivation in reinforcement learning, November 2019.
arXiv:1908.06976 [cs].

Auer, P., Cesa-Bianchi, N., and Fischer, P.
Finite-time Analysis of the Multiarmed Bandit Problem.
Machine Learning, 47(2):235–256, May 2002a.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E.
The Nonstochastic Multiarmed Bandit Problem.
SIAM Journal on Computing, 32(1):48–77, January 2002b.
Publisher: Society for Industrial and Applied Mathematics.

Azar, M. G., Guo, Z. D., Piot, B., Munos, R., Rowland, M., Valko, M., and Calandriello, D.
A General Theoretical Paradigm to Understand Learning from Human Preferences.
In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, April 2024.
ISSN: 2640-3498.

Bahdanau, D., Cho, K., and Bengio, Y.
Neural Machine Translation by Jointly Learning to Align and Translate.
CoRR, September 2014.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y.
An Actor-Critic Algorithm for Sequence Prediction.
In *International Conference on Learning Representations (ICLR 2017)*. International Conference on Learning Representations (ICLR), February 2017.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, A., Kohli, P., and Grefenstette, E.
Learning to Understand Goal Specifications by Modelling Reward.
In *International Conference on Learning Representations (ICLR 2019)*. International Conference on Learning Representations (ICLR), September 2019a.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., Vries, H. d., and Courville, A.
Systematic Generalization: What Is Required and Can It Be Learned?
In *International Conference on Learning Representations (ICLR 2019)*. International Conference on Learning Representations (ICLR), September 2019b.

Bahlous-Boldi, R., Ding, L., Spector, L., and Niekum, S.
Pareto-Optimal Learning from Preferences with Hidden Context, February 2025.
arXiv:2406.15599 [cs].

Bai, H., Zhou, Y., Cemri, M., Pan, J., Suhr, A., Levine, S., and Kumar, A.
DigiRL: training in-the-wild device-control agents with autonomous reinforcement learning.
In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, volume 37 of *NIPS '24*, pp. 12461–12495, Red Hook, NY, USA, 2025. Curran Associates Inc.

Baldassarre, G. and Mirolli, M. (eds.).
Intrinsically Motivated Learning in Natural and Artificial Systems.
Springer, Berlin, Heidelberg, 2013.

Ban, Y., Yan, Y., Banerjee, A., and He, J.
Neural Exploitation and Exploration of Contextual Bandits, May 2023.
arXiv:2305.03784 [cs].

Baranes, A. and Oudeyer, P.-Y.
R-IAC: Robust Intrinsically Motivated Exploration and Active Learning.
IEEE Transactions on Autonomous Mental Development, 1(3):155–169, October 2009.
Conference Name: IEEE Transactions on Autonomous Mental Development.

Baranes, A. and Oudeyer, P.-Y.
Active learning of inverse models with intrinsically motivated goal exploration in robots.
Robotics and Autonomous Systems, 61(1):49–73, January 2013.

Barsalou, L. W.
Perceptual symbol systems.
Behavioral and Brain Sciences, 22(4):577–660, 1999.
Place: United Kingdom Publisher: Cambridge University Press.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S.
DeepMind Lab, December 2016.
arXiv:1612.03801 [cs].

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M.
The Arcade Learning Environment: An Evaluation Platform for General Agents.
Journal of Artificial Intelligence Research, 47:253–279, June 2013.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R.
Unifying count-based exploration and intrinsic motivation.
In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 1479–1487, Red Hook, NY, USA, 2016. Curran Associates Inc.

Bellman, R.
Dynamic Programming.
Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

Beltagy, I., Peters, M. E., and Cohan, A.
Longformer: The Long-Document Transformer, December 2020.
arXiv:2004.05150 [cs].

Bender, E. M. and Koller, A.
Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data.
In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5185–5198, Online, July 2020. Association for Computational Linguistics.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S.
On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?
In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pp. 610–623, New York, NY, USA, March 2021. Association for Computing Machinery.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C.
A neural probabilistic language model.
J. Mach. Learn. Res., 3(null):1137–1155, March 2003.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J.
Curriculum learning.
In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, June 2009. Association for Computing Machinery.

Berlyne, D. E.
A theory of human curiosity.
British Journal of Psychology, 45:180–191, 1954.
Place: United Kingdom Publisher: British Psychological Society.

Berseth, G., Geng, D., Devin, C. M., Rhinehart, N., Finn, C., Jayaraman, D., and Levine, S.
SMiRL: Surprise Minimizing Reinforcement Learning in Unstable Environments.
In *International Conference on Learning Representations (ICLR 2021)*. International Conference on Learning Representations (ICLR), October 2021.

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A., Pinto, N., and Turian, J.
Experience Grounds Language, November 2020.
arXiv:2004.10151 [cs].

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S.
GPT-NeoX-20B: An Open-Source Autoregressive Language Model.
In Fan, A., Ilic, S., Wolf, T., and Gallé, M. (eds.), *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics.

Blaes, S., Vlastelica Pogančić, M., Zhu, J., and Martius, G.
Control What You Can: Intrinsically Motivated Task-Planning Agent.
In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Blei, D. M., Ng, A. Y., and Jordan, M. I.
Latent dirichlet allocation.
J. Mach. Learn. Res., 3(null):993–1022, March 2003.

Block, N.
Psychologism and Behaviorism.
The Philosophical Review, 90(1):5–43, 1981.
Publisher: [Duke University Press, Philosophical Review].

Boleda, G.
Distributional Semantics and Linguistic Theory.
Annual Review of Linguistics, 6(Volume 6, 2020):213–234, January 2020.
Publisher: Annual Reviews.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W.
OpenAI Gym, June 2016.
arXiv:1606.01540 [cs].

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M. S., Salazar, G., Sanketi, P. R., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q. H., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B.
RT-1: Robotics Transformer for Real-World Control at Scale.
In *Robotics: Science and System XIX*, volume 19, July 2023.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D.
Language models are few-shot learners.
In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pp. 1877–1901, Red Hook, NY, USA, 2020. Curran Associates Inc.

Bruner, J.
Child's Talk: Learning to Use Language.
Child Language Teaching and Therapy, 1(1):111–114, January 1985.
Publisher: SAGE Publications Ltd.

Bruner, J. S.
Acts of meaning.
Cambridge, MA: Harvard University Press, 1990.
Pages: 181 pages.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A.
Large-scale study of curiosity-driven learning.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *7th International Conference on Learning Representations (ICLR 2019)*. Curran Associates, Inc., 2019a.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O.
Exploration by random network distillation.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *7th International Conference on Learning Representations (ICLR 2019)*. Curran Associates, Inc., 2019b.

Cangelosi, A. and Schlesinger, M.
Developmental Robotics: From Babies to Robots.
The MIT Press, January 2015.

Cangelosi, A. and Stramandinoli, F.
A review of abstract concept learning in embodied agents and robots.
Philosophical Transactions of the Royal Society B: Biological Sciences, 373(1752): 20170131, June 2018.
Publisher: Royal Society.

Cangelosi, A., Metta, G., Sagerer, G., Nolfi, S., Nehaniv, C., Fischer, K., Tani, J., Belpaeme, T., Sandini, G., Nori, F., Fadiga, L., Wrede, B., Rohlfing, K., Tuci, E., Dautenhahn, K., Saunders, J., and Zeschel, A.
Integration of Action and Language Knowledge: A Roadmap for Developmental Robotics.
IEEE Transactions on Autonomous Mental Development, 2(3):167–195, September 2010.

Carta, T., Romac, C., Gaven, L., Oudeyer, P.-Y., Sigaud, O., and Lamprier, S.
HERAKLES: Hierarchical Skill Compilation for Open-ended LLM Agents, August 2025.
arXiv:2508.14751 [cs].

Castanet, N., Sigaud, O., and Lamprier, S.
Stein variational goal generation for adaptive exploration in multi-goal reinforcement learning.
In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pp. 3714–3731, Honolulu, Hawaii, USA, 2023. JMLR.org.

Chebotar, Y., Vuong, Q., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., Gopalakrishnan, K., Ibarz, J., Nachum, O., Sontakke, S. A., Salazar, G., Tran, H. T., Peralta, J., Tan, C., Manjunath, D., Singh, J., Zitkovich, B., Jackson, T., Rao, K., Finn, C., and Levine, S.
Q-Transformer: Scalable Offline Reinforcement Learning via Autoregressive Q-Functions.
In *Proceedings of The 7th Conference on Robot Learning*, pp. 3909–3928. PMLR, December 2023.
ISSN: 2640-3498.

Chen, K., Cusumano-Towner, M., Huval, B., Petrenko, A., Hamburger, J., Koltun, V., and Krähenbühl, P.
Reinforcement Learning for Long-Horizon Interactive LLM Agents, March 2025.
arXiv:2502.01600 [cs].

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I.
Decision Transformer: Reinforcement Learning via Sequence Modeling.
In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021.

Chen, W., Chen, J., Tajwar, F., Zhu, H., Duan, X., Salakhutdinov, R., and Schneider, J.
Fine-tuning LLM Agents with Retrospective In-Context Online Learning.
In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, November 2024.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K.
BAIL: Best-Action Imitation Learning for Batch Deep Reinforcement Learning.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 18353–18363. Curran Associates, Inc., 2020a.

Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A. J., Padlewski, P., Salz, D., Goodman, S., Grycner, A., Mustafa, B., Beyer, L., Kolesnikov, A., Puigcerver, J., Ding, N., Rong, K., Akbari, H., Mishra, G., Xue, L., Thapliyal, A., Bradbury, J., Kuo, W., Seyedhosseini, M., Jia, C., Ayan, B. K., Riquelme, C., Steiner, A., Angelova, A., Zhai, X., Houlsby, N., and Soricut, R.
PaLI: A Jointly-Scaled Multilingual Language-Image Model, June 2023.
arXiv:2209.06794 [cs].

Chen, Y.-C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., and Liu, J.
UNITER: UNiversal Image-TExt Representation Learning.
In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), *Computer Vision – ECCV 2020*, pp. 104–120, Cham, 2020b. Springer International Publishing.

Chersi, F., Thill, S., Ziemke, T., and Borghi, A. M.
Sentence processing: linking language to motor chains.
Frontiers in Neurorobotics, 4:4, 2010.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y.
BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning.
In *International Conference on Learning Representations (ICLR 2019)*. International Conference on Learning Representations (ICLR), September 2019.

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J.
Minigrid & miniworld: modular & customizable reinforcement learning environments for goal-oriented tasks.
In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pp. 73383–73394, Red Hook, NY, USA, 2023. Curran Associates Inc.

Cho, J., Lei, J., Tan, H., and Bansal, M.
Unifying Vision-and-Language Tasks via Text Generation.
In *Proceedings of the 38th International Conference on Machine Learning*, pp. 1931–1942. PMLR, July 2021.
ISSN: 2640-3498.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y.
On the Properties of Neural Machine Translation: Encoder–Decoder Approaches.
In Wu, D., Carpuat, M., Carreras, X., and Vecchi, E. M. (eds.), *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics.

Chomsky, N.
Syntactic structures.
Syntactic structures. Mouton, Oxford, England, 1957.
Pages: 116.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N.
PaLM: Scaling Language Modeling with Pathways, October 2022.
arXiv:2204.02311 [cs].

Christianos, F., Papoudakis, G., Zimmer, M., Coste, T., Wu, Z., Chen, J., Khandelwal, K., Doran, J., Feng, X., Liu, J., Xiong, Z., Luo, Y., Hao, J., Shao, K., Bou-Ammar, H., and Wang, J.
Pangu-Agent: A Fine-Tunable Generalist Agent with Structured Reasoning, December 2023.
arXiv:2312.14878 [cs].

Christodoulou, P.
Soft Actor-Critic for Discrete Action Settings, October 2019.
arXiv:1910.07207 [cs].

Chuang, Y.-N., Sarma, P. K., Gopalan, P., Boccio, J., Bolouki, S., Hu, X., and Zhou, H.
Learning to Route LLMs with Confidence Tokens, June 2025.
arXiv:2410.13284 [cs].

Clement, B., Roy, D., Oudeyer, P.-Y., and Lopes, M.
Multi-Armed Bandits for Intelligent Tutoring Systems.
Journal of Educational Data Mining, 7(2):20–48, June 2015.
Number: 2.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J.
Quantifying Generalization in Reinforcement Learning.
In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1282–1289. PMLR, May 2019.
ISSN: 2640-3498.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J.
Leveraging Procedural Generation to Benchmark Reinforcement Learning.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 2048–2056. PMLR, November 2020.
ISSN: 2640-3498.

Cohen, R., Dobler, K., Biran, E., and de Melo, G.
I Don't Know: Explicit Modeling of Uncertainty with an [IDK] Token.
Advances in Neural Information Processing Systems, 37:10935–10958, December 2024.

Colas, C., Sigaud, O., and Oudeyer, P.-Y.
How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments, July 2018.
arXiv:1806.08295 [cs].

Colas, C., Fournier, P., Chetouani, M., Sigaud, O., and Oudeyer, P.-Y.
CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning.
In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1331–1340. PMLR, May 2019.
ISSN: 2640-3498.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P., and Oudeyer, P.-Y.
Language as a Cognitive Tool to Imagine Goals in Curiosity Driven Exploration.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3761–3774. Curran Associates, Inc., 2020.

Colas, C., Karch, T., Moulin-Frier, C., and Oudeyer, P.-Y.
Language and culture internalization for human-like autotelic AI.
Nature Machine Intelligence, 4(12):1068–1076, December 2022a.
Publisher: Nature Publishing Group.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y.
Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey.
Journal of Artificial Intelligence Research, 74:1159–1199, 2022b.

Colas, C., Teodorescu, L., Oudeyer, P.-Y., Yuan, X., and Côté, M.-A.
Augmenting autotelic agents with large language models.
In *Conference on Lifelong Learning Agents*, pp. 205–226. PMLR, 2023.

Craik, K. J. W.
The Nature of Explanation.

Cambridge University Press, January 1943.
Google-Books-ID: EN0TrgEACAAJ.

Côté, M.-A., Kádár, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., El Asri, L., Adada, M., Tay, W., and Trischler, A.
TextWorld: A Learning Environment for Text-Based Games.
In Cazenave, T., Saffidine, A., and Sturtevant, N. (eds.), *Computer Games*, pp. 41–75, Cham, 2019. Springer International Publishing.

Dainese, N., Marttinen, P., and Ilin, A.
Reader: Model-based language-instructed reinforcement learning.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 16583–16599, Singapore, December 2023. Association for Computational Linguistics.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D.
Embodied Question Answering.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Dasgupta, I., Lampinen, A. K., Chan, S. C. Y., Creswell, A., Kumaran, D., McClelland, J. L., and Hill, F.
Language models show human-like content effects on reasoning, July 2022.
arXiv:2207.07051 [cs].

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z.
DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, January 2025.

arXiv:2501.12948 [cs].

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J.-M., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., and Riedmiller, M.
Magnetic control of tokamak plasmas through deep reinforcement learning.
Nature, 602(7897):414–419, February 2022.
Publisher: Nature Publishing Group.

Deng, L.
The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web].
IEEE Signal Processing Magazine, 29(6):141–142, November 2012.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S.
Emergent complexity and zero-shot transfer via unsupervised environment design.
In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pp. 13049–13061, Red Hook, NY, USA, 2020. Curran Associates Inc.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K.
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L. V. S., and Awadallah, A. H.
Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing, April 2024.
arXiv:2404.14618 [cs].

Ding, J., Zhang, Y., Shang, Y., Zhang, Y., Zong, Z., Feng, J., Yuan, Y., Su, H., Li, N., Sukiennik, N., Xu, F., and Li, Y.
Understanding World or Predicting Future? A Comprehensive Survey of World Models.
ACM Computing Surveys, pp. 3746449, June 2025.

Dinh, L., Sohl-Dickstein, J., and Bengio, S.
Density estimation using Real NVP, February 2017.
arXiv:1605.08803 [cs].

Dominey, P. F., Mallet, A., and Yoshida, E.
Real-time spoken-language programming for cooperative interaction with a humanoid apprentice.
International Journal of Humanoid Robotics, 06(02):147–171, June 2009.
Publisher: World Scientific Publishing Co.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T.
DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition.
In *Proceedings of the 31st International Conference on Machine Learning*, pp. 647–655. PMLR, January 2014.
ISSN: 1938-7228.

Doroudi, S., Aleven, V., and Brunskill, E.
Where's the Reward?
International Journal of Artificial Intelligence in Education, 29(4):568–620, December 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N.
An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.
In *International Conference on Learning Representations (ICLR 2021)*. International Conference on Learning Representations (ICLR), October 2020.

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P.
PaLM-E: An Embodied Multimodal Language Model.
In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 23–29 Jul 2023.

Drugan, M. M. and Nowe, A.
Designing multi-objective multi-armed bandits algorithms: A study.
In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, August 2013.
ISSN: 2161-4407.

Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., and Lou, Y.
Evaluating Large Language Models in Class-Level Code Generation.
In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, pp. 1–13, New York, NY, USA, 2024. Association for Computing Machinery.

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J.
Guiding Pretraining in Reinforcement Learning with Large Language Models.
In *Proceedings of the 40th International Conference on Machine Learning*, pp. 8657–8677. PMLR, July 2023.
ISSN: 2640-3498.

Dumais, S. T.
Latent Semantic Analysis.
Annual Review of Information Science and Technology (ARIST), 38:189–230, 2004.
ERIC Number: EJ678116.

Ebert, D., Sun, C., and Pavlick, E.
Do Trajectories Encode Verb Meaning?
In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V. (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2860–2871, Seattle, United States, July 2022. Association for Computational Linguistics.

Elliot, A. J. and Fryer, J. W.
The goal construct in psychology.
In *Handbook of motivation science*, pp. 235–250. The Guilford Press, New York, NY, US, 2008.

Elman, J. L.
Learning and development in neural networks: the importance of starting small.
Cognition, 48(1):71–99, July 1993.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K.
IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures.
In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1407–1416. PMLR, July 2018.
ISSN: 2640-3498.

Etcheverry, M., Moulin-Frier, C., and Oudeyer, P.-Y.
Hierarchically Organized Latent Modules for Exploratory Search in Morphogenetic Systems.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 4846–4859. Curran Associates, Inc., 2020.

Etcheverry, M., Moulin-Frier, C., Oudeyer, P.-Y., and Levin, M.
AI-driven Automated Discovery Tools Reveal Diverse Behavioral Competencies of Biological Networks.
eLife, 13, August 2024.
Publisher: eLife Sciences Publications Limited.

Faldor, M., Zhang, J., Cully, A., and Clune, J.
OMNI-EPIC: Open-endedness via Models of human Notions of Interestingness with Environments Programmed in Code.
In *International Conference on Learning Representations (ICLR 2025)*. International Conference on Learning Representations (ICLR), 2025.

Falk, M. J.
Curiosity-driven search for novel nonequilibrium behaviors.
Physical Review Research, 6(3), 2024.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A.
MINEDOJO: building open-ended embodied agents with internet-scale knowledge.

In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 18343–18362, Red Hook, NY, USA, November 2022. Curran Associates Inc.

Feng, J., Huang, S., Qu, X., Zhang, G., Qin, Y., Zhong, B., Jiang, C., Chi, J., and Zhong, W.
ReTool: Reinforcement Learning for Strategic Tool Use in LLMs, April 2025.
arXiv:2504.11536 [cs].

Firth, J. R.
A Synopsis of Linguistic Theory, 1930-1955.
1957.
Google-Books-ID: T8LDtgAACAAJ.

Florensa, C., Held, D., Geng, X., and Abbeel, P.
Automatic Goal Generation for Reinforcement Learning Agents.
In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1515–1528. PMLR, July 2018.
ISSN: 2640-3498.

Forbes, M., Holtzman, A., and Choi, Y.
Do Neural Language Representations Learn Physical Commonsense?
Proceedings of the Annual Meeting of the Cognitive Science Society, 41(0), 2019.

Forestier, S. and Oudeyer, P.-Y.
Modular Active Curiosity-Driven Discovery of Tool Use.
In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, South Korea, 2016.

Forestier, S., Portelas, R., Mollard, Y., and Oudeyer, P.-Y.
Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning, May 2022.
arXiv:1708.02190 [cs].

Fournier, P., Sigaud, O., Chetouani, M., and Oudeyer, P.-Y.
Accuracy-based Curriculum Learning in Deep Reinforcement Learning, September 2018.
arXiv:1806.09614 [cs].

François, C., Péran, L., Bdeir, A., Dziri, N., Hawkins, W., Jernite, Y., Kapoor, S., Shen, J., Khlaaf, H., Klyman, K., Marda, N., Pellat, M., Raji, D., Siddarth, D., Skowron, A., Spisak, J., Srikumar, M., Storchan, V., Tang, A., and Weedon, J.
A Different Approach to AI Safety: Proceedings from the Columbia Convening on Openness in Artificial Intelligence and AI Safety, June 2025.
arXiv:2506.22183 [cs].

Gage, P.
A new algorithm for data compression.
C Users J., 12(2):23–38, 1994.

Gaier, A. and Ha, D.
Weight Agnostic Neural Networks.
In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Gallegos, I. O., Rossi, R. A., Barrow, J., Tanjim, M. M., Kim, S., Dernoncourt, F., Yu, T., Zhang, R., and Ahmed, N. K.
Bias and Fairness in Large Language Models: A Survey.
Computational Linguistics, 50(3):1097–1179, September 2024.
Place: Cambridge, MA Publisher: MIT Press.

Gallouédec, Q., Beeching, E., Romac, C., and Dellandréa, E.
Jack of All Trades, Master of Some, a Multi-Purpose Transformer Agent, July 2024.
arXiv:2402.09844 [cs].

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G.
PAL: Program-aided Language Models, January 2023.
arXiv:2211.10435 [cs].

Gasse, M., Grasset, D., Gaudron, G., and Oudeyer, P.-Y.
Using Confounded Data in Latent Model-Based Reinforcement Learning.
Transactions on Machine Learning Research, March 2023.

Ge, Y., Hua, W., Mei, K., Ji, J., Tan, J., Xu, S., Li, Z., and Zhang, Y.
OpenAGI: When LLM Meets Domain Experts.
Advances in Neural Information Processing Systems, 36:5539–5568, December 2023.

Gehring, J., Zheng, K., Copet, J., Mella, V., Cohen, T., and Synnaeve, G.
RLEF: Grounding Code LLMs in Execution Feedback with Reinforcement Learning.
In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, June 2025.

Gelman, S. A. and Legare, C. H.
Concepts and Folk Theories*.
Annual Review of Anthropology, 40(Volume 40, 2011):379–398, October 2011.
Publisher: Annual Reviews.

Girshick, R., Donahue, J., Darrell, T., and Malik, J.
Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.
2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587, 2013.

Glenberg, A. M. and Kaschak, M. P.
Grounding language in action.
Psychonomic Bulletin & Review, 9(3):558–565, 2002.
Place: US Publisher: Psychonomic Society.

Glenberg, A. M. and Robertson, D. A.
Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories of Meaning.
Journal of Memory and Language, 43(3):379–401, October 2000.

Goldberg, Y.
A Primer on Neural Network Models for Natural Language Processing.
Journal of Artificial Intelligence Research, 57:345–420, November 2016.

Golinkoff, R. M., Can, D. D., Soderstrom, M., and Hirsh-Pasek, K.
(Baby)Talk to Me: The Social Context of Infant-Directed Speech and Its Effects on Early Language Acquisition.
Current Directions in Psychological Science, 24(5):339–344, October 2015.
Publisher: SAGE Publications Inc.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.
Generative Adversarial Nets.
In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Goodman, N. D., Tenenbaum, J. B., Feldman, J., and Griffiths, T. L.
A Rational Analysis of Rule-Based Concept Learning.
Cognitive Science, 32(1):108–154, 2008.
_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1080/03640210701802071.

Gopnik, A.
The Scientist as Child.
Philosophy of Science, 63(4):485–514, 1996.
Publisher: [Cambridge University Press, The University of Chicago Press, Philosophy of Science Association].

Gopnik, A. and Meltzoff, A. N.
Words, Thoughts, and Theories.
The MIT Press, January 1997.

Gopnik, A. and Wellman, H. M.
Why the Child's Theory of Mind Really Is a Theory.
Mind & Language, 7(1-2):145–171, 1992.
_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0017.1992.tb00202.x.

Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., and Farhadi, A.
IQA: Visual Question Answering in Interactive Environments.
In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4089–4098, 2018.

Gottlieb, J. and Oudeyer, P.-Y.
Towards a neuroscience of active sampling and curiosity.
Nature Reviews Neuroscience, 19(12):758–770, December 2018.
Publisher: Nature Publishing Group.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A.,

Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., Linde, J. v. d., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., Maaten, L. v. d., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., Oliveira, L. d., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I.,

Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z.
The Llama 3 Herd of Models, November 2024.
arXiv:2407.21783 [cs].

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K.
Automated Curriculum Learning for Neural Networks.
In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1311–1320. PMLR, July 2017.
ISSN: 2640-3498.

Gregor, K., Rezende, D. J., and Wierstra, D.
Variational Intrinsic Control.
In *International Conference on Learning Representations (ICLR 2017)*. International Conference on Learning Representations (ICLR), February 2017.

Grizou, J., Points, L. J., Sharma, A., and Cronin, L.
A curious formulation robot enables the discovery of a novel protocell behavior.
Science Advances, January 2020.
Publisher: American Association for the Advancement of Science.

Gubelmann, R.

Pragmatic Norms Are All You Need – Why The Symbol Grounding Problem Does Not Apply to LLMs.
In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11663–11678, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., Macherey, W., Doucet, A., Firat, O., and Freitas, N. d.
Reinforced Self-Training (ReST) for Language Modeling, August 2023.
arXiv:2308.08998 [cs].

Ha, D.
Generating abstract patterns with tensorflow.
blog.otoro.net, 2016.

Ha, D.
Reinforcement Learning for Improving Agent Design.
Artificial Life, 25(4):352–365, November 2019.

Ha, D. and Schmidhuber, J.
World Models, March 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S.
Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.
In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, July 2018.
ISSN: 2640-3498.

Hafner, D.
Benchmarking the Spectrum of Agent Capabilities, February 2022.
arXiv:2109.06780 [cs].

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M.
Dream to Control: Learning Behaviors by Latent Imagination.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), April 2020.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J.
Mastering Atari with Discrete World Models.
In *International Conference on Learning Representations (ICLR 2021)*. International Conference on Learning Representations (ICLR), October 2021.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T.
Mastering diverse control tasks through world models.
Nature, 640(8059):647–653, April 2025.
Publisher: Nature Publishing Group.

Hansen, N. A., Su, H., and Wang, X.
Temporal Difference Learning for Model Predictive Control.
In *Proceedings of the 39th International Conference on Machine Learning*, pp. 8387–8406. PMLR, June 2022.
ISSN: 2640-3498.

Hansen, S., Pritzel, A., Sprechmann, P., Barreto, A., and Blundell, C.
Fast deep reinforcement learning using online adjustments from the past.
In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z.
Reasoning with Language Model is Planning with World Model.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023a. Association for Computational Linguistics.

Hao, S., Liu, T., Wang, Z., and Hu, Z.
ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings.
Advances in Neural Information Processing Systems, 36:45870–45894, December 2023b.

Harman, G.
Conceptual role semantics.
Notre Dame Journal of Formal Logic, 23(2), April 1982.

Harnad, S.
The symbol grounding problem.
Physica D: Nonlinear Phenomena, 42(1):335–346, June 1990.

Harris, Z. S.
Distributional Structure.
WORD, 10(2-3):146–162, August 1954.

Hausknecht, M., Ammanabrolu, P., Côté, M.-A., and Yuan, X.
Interactive Fiction Games: A Colossal Adventure.
Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):7903–7910, April 2020.
Number: 05.

Havrilla, A., Du, Y., Raparthy, S. C., Nalmpantis, C., Dwivedi-Yu, J., Hambro, E., Sukhbaatar, S., and Raileanu, R.
Teaching Large Language Models to Reason with Reinforcement Learning, June 2024.

He, K., Zhang, X., Ren, S., and Sun, J.
Deep Residual Learning for Image Recognition.
In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.
ISSN: 1063-6919.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., Wainwright, M., Apps, C., Hassabis, D., and Blunsom, P.
Grounded Language Learning in a Simulated 3D World, June 2017.
arXiv:1706.06551 [cs].

Hernandez, D., Denamganaï, K., Gao, Y., York, P., Devlin, S., Samothrakis, S., and Walker, J. A.
A Generalized Framework for Self-Play Training.
In *2019 IEEE Conference on Games (CoG)*, pp. 1–8, August 2019.
ISSN: 2325-4289.

Hernandez-Leal, P., Kartal, B., and Taylor, M. E.
A Survey and Critique of Multiagent Deep Reinforcement Learning.
Autonomous Agents and Multi-Agent Systems, 33:750 – 797, 2018.

Hessel, M., Modayil, J., Hasselt, H. v., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D.
Rainbow: Combining Improvements in Deep Reinforcement Learning.
Proceedings of the AAAI Conference on Artificial Intelligence, 32(1), April 2018.
Number: 1.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H.
Multi-task deep reinforcement learning with PopArt.
In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, volume 33 of *AAAI'19/IAAI'19/EAAI'19*, pp. 3796–3803, Honolulu, Hawaii, USA, January 2019. AAAI Press.

Hill, B.
Taking the help or going alone: ChatGPT and class assignments, March 2023.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A.
Environmental drivers of systematicity and generalization in a situated agent.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), September 2020a.

Hill, F., Mokra, S., Wong, N., and Harley, T.
Human Instruction-Following with Deep Reinforcement Learning via Transfer-Learning from Text.
arXiv:2005.09382 [cs], May 2020b.
arXiv: 2005.09382.

Hill, F., Tieleman, O., Glehn, T. v., Wong, N., Merzic, H., and Clark, S.
Grounded Language Learning Fast and Slow.
In *International Conference on Learning Representations (ICLR 2021)*. International Conference on Learning Representations (ICLR), October 2021.

Hochreiter, S. and Schmidhuber, J.
Long Short-Term Memory.
Neural Computation, 9(8):1735–1780, November 1997.

Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P.
VIME: Variational Information Maximizing Exploration.
In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Howell, S. R., Jankowicz, D., and Becker, S.
A model of grounded language acquisition: Sensorimotor features improve lexical and grammatical learning.
Journal of Memory and Language, 53(2):258–276, August 2005.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W.
LoRA: Low-Rank Adaptation of Large Language Models.
In *International Conference on Learning Representations (ICLR 2022)*. International Conference on Learning Representations (ICLR), October 2022.

Hu, H., Ye, J., Zhu, G., Ren, Z., and Zhang, C.
Generalizable Episodic Memory for Deep Reinforcement Learning.
In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4380–4390. PMLR, July 2021.
ISSN: 2640-3498.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I.
Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents.
In *Proceedings of the 39th International Conference on Machine Learning*, pp. 9118–9147. PMLR, June 2022.
ISSN: 2640-3498.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Jackson, T., Brown, N., Luu, L., Levine, S., Hausman, K., and Ichter, B.
Inner Monologue: Embodied Reasoning through Planning with Language Models.
In *Proceedings of The 6th Conference on Robot Learning*, pp. 1769–1782. PMLR, March 2023.
ISSN: 2640-3498.

Hugging Face.
Open r1: A fully open reproduction of deepseek-r1, January 2025.

Hutchins, W. J.
The Georgetown-IBM experiment demonstrated in January 1954.
In Frederking, R. E. and Taylor, K. B. (eds.), *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pp. 102–114, Washington, USA, September 2004. Springer.

Jabri, A., Hsu, K., Eysenbach, B., Gupta, A., Efros, A. A., Levine, S., and Finn, C.
Unsupervised Curricula for Visual Meta-Reinforcement Learning.
In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, September 2019.

Janner, M., Li, Q., and Levine, S.
Offline Reinforcement Learning as One Big Sequence Modeling Problem.
In *Advances in Neural Information Processing Systems*, volume 34, pp. 1273–1286. Curran Associates, Inc., 2021.

Jansen, P.
A Systematic Survey of Text Worlds as Embodied Natural Language Environments.
In Côté, M.-A., Yuan, X., and Ammanabrolu, P. (eds.), *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, pp. 1–15, Seattle, United States, July 2022. Association for Computational Linguistics.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D.
Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control.
In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1645–1654. PMLR, July 2017.
ISSN: 2640-3498.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E.
Mistral 7B, October 2023a.
arXiv:2310.06825 [cs].

Jiang, M., Luketina, J., Nardelli, N., Minervini, P., Torr, P. H. S., Whiteson, S., and Rocktäschel, T.
WordCraft: An Environment for Benchmarking Commonsense Agents, July 2020.
arXiv:2007.09185 [cs].

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., and Rocktäschel, T.
Replay-guided adversarial environment design.
In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pp. 1884–1897, Red Hook, NY, USA, 2021a. Curran Associates Inc.

Jiang, M., Grefenstette, E., and Rocktäschel, T.
Prioritized Level Replay.
In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4940–4950. PMLR, July 2021b.
ISSN: 2640-3498.

Jiang, Y., Gu, S., Murphy, K., and Finn, C.
Language as an abstraction for hierarchical deep reinforcement learning.
In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 845, pp. 9419–9431. Curran Associates Inc., Red Hook, NY, USA, 2019.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L.
VIMA: robot manipulation with multimodal prompts.
In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pp. 14975–15022, Honolulu, Hawaii, USA, 2023b. JMLR.org.

Johnson, B.
Metacognition for artificial intelligence system safety – An approach to safe and desired behavior.
Safety Science, 151:105743, July 2022.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D.
The Malmo platform for artificial intelligence experimentation.
In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 4246–4247, New York, New York, USA, 2016. AAAI Press.

Johnson, S. G. B., Karimi, A.-H., Bengio, Y., Chater, N., Gerstenberg, T., Larson, K., Levine, S., Mitchell, M., Rahwan, I., Schölkopf, B., and Grossmann, I.
Imagining and building wise machines: The centrality of AI metacognition, May 2025.
arXiv:2411.02478 [cs].

Johnson-Laird, P. N.
Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness.
Harvard University Press, 1983.
Google-Books-ID: FS3zSKAfLGMC.

Jones, K. S.
Natural Language Processing: A Historical Review.
In Zampolli, A., Calzolari, N., and Palmer, M. (eds.), *Current Issues in Computational Linguistics: In Honour of Don Walker*, pp. 3–16. Springer Netherlands, Dordrecht, 1994.

Justesen, N., Torrado, R. R., Bontrager, P., Khalifa, A., Togelius, J., and Risi, S.
Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation, November 2018.
arXiv:1806.10729 [cs].

Kadlčík, M., Štefánik, M., Sotolar, O., and Martinek, V.
Calc-X and Calcformers: Empowering Arithmetical Chain-of-Thought through Interaction with Symbolic Systems.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12101–12108, Singapore, December 2023. Association for Computational Linguistics.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S.
Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation.
In *Proceedings of The 2nd Conference on Robot Learning*, pp. 651–673. PMLR, October 2018.
ISSN: 2640-3498.

Kalchbrenner, N. and Blunsom, P.
Recurrent continuous translation models.
In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1700–1709, 2013.

Kanitscheider, I., Huizinga, J., Farhi, D., Guss, W. H., Houghton, B., Sampedro, R., Zhokhov, P., Baker, B., Ecoffet, A., Tang, J., Klimov, O., and Clune, J.
Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft, June 2021.
arXiv:2106.14876 [cs].

Kaplan, F. and Oudeyer, P.-Y.
In Search of the Neural Circuits of Intrinsic Motivation.
Frontiers in Neuroscience, 1(1):225–236, October 2007.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W.
Recurrent Experience Replay in Distributed Reinforcement Learning.
In *International Conference on Learning Representations (ICLR 2019)*. International Conference on Learning Representations (ICLR), September 2019.

Khajehabdollahi, S., Hamon, G., Cvjetko, M., Oudeyer, P.-Y., Moulin-Frier, C., and Colas, C.
Expedition & Expansion: Leveraging Semantic Representations for Goal-Directed Exploration in Continuous Cellular Automata, September 2025.
arXiv:2509.03863 [cs].

Kidd, C. and Hayden, B. Y.
The Psychology and Neuroscience of Curiosity.
Neuron, 88(3):449–460, November 2015.
Publisher: Elsevier.

Kiela, D., Bulat, L., Vero, A. L., and Clark, S.
Virtual Embodiment: A Scalable Long-Term Strategy for Artificial Intelligence Research, October 2016.
arXiv:1610.07432 [cs].

Kim, K., Sano, M., Freitas, J. D., Haber, N., and Yamins, D.
Active World Model Learning with Progress Curiosity.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5306–5315. PMLR, November 2020.
ISSN: 2640-3498.

Kingma, D. P. and Ba, J.
Adam: A Method for Stochastic Optimization.
In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Kirby, S., Griffiths, T., and Smith, K.
Iterated learning and the evolution of language.
Current Opinion in Neurobiology, 28:108–114, October 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R.
Overcoming catastrophic forgetting in neural networks.
Proceedings of the National Academy of Sciences, 114(13):3521–3526, March 2017.
Publisher: Proceedings of the National Academy of Sciences.

Klink, P., D' Eramo, C., Peters, J. R., and Pajarinen, J.
Self-Paced Deep Reinforcement Learning.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 9216–9227. Curran Associates, Inc., 2020.

Klink, P., D'Eramo, C., Peters, J., and Pajarinen, J.
On the Benefit of Optimal Transport for Curriculum Reinforcement Learning.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(11):7191–7204, November 2024.

Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., and Henaff, M.
Motif: Intrinsic Motivation from Artificial Intelligence Feedback, September 2023.
arXiv:2310.00166 [cs].

Klyubin, A. S., Polani, D., and Nehaniv, C. L.
All Else Being Equal Be Empowered.
In Capcarrère, M. S., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J. (eds.), *Advances in Artificial Life*, pp. 744–753, Berlin, Heidelberg, 2005. Springer.

Kovač, G., Laversanne-Finot, A., and Oudeyer, P.-Y.
GRIMGEP: Learning Progress for Robust Goal Sampling in Visual Deep Reinforcement Learning.
IEEE Transactions on Cognitive and Developmental Systems, 15(3):1396–1407, September 2023.
arXiv:2008.04388 [cs].

Kovač, G., Sawayama, M., Portelas, R., Colas, C., Dominey, P. F., and Oudeyer, P.-Y.
Large Language Models as superpositions of cultural perspectives.
In *12th International Conference on Learning Representations (ICLR 2024)*, October 2024.

Kudo, T.
Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates.
In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Kumar, A., Hong, J., Singh, A., and Levine, S.
Should I Run Offline Reinforcement Learning or Behavioral Cloning?
In *International Conference on Learning Representations (ICLR 2022)*. International Conference on Learning Representations (ICLR), October 2022.

Kumar, N., Silver, T., McClinton, W., Zhao, L., Proulx, S., Lozano-Pérez, T., Kaelbling, L. P., and Barry, J.
Practice Makes Perfect: Planning to Learn Skill Parameter Policies, May 2024.
arXiv:2402.15025 [cs].

Küttler, H., Nardelli, N., Miller, A. H., Raileanu, R., Selvatici, M., Grefenstette, E., and Rocktäschel, T.
The NetHack learning environment.
In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pp. 7671–7684, Red Hook, NY, USA, 2020. Curran Associates Inc.

Lai, T. and Robbins, H.
Asymptotically efficient adaptive allocation rules.
Adv. Appl. Math., 6(1):4–22, March 1985.

Lair, N., Colas, C., Portelas, R., Dussoux, J.-M., Dominey, P. F., and Oudeyer, P.-Y.
Language Grounding through Social Interactions and Curiosity-Driven Multi-Goal Learning, November 2019.
arXiv:1911.03219 [cs].

Lampinen, A. K., Roy, N. A., Dasgupta, I., Chan, S. C. Y., Tam, A., Yan, C., Santoro, A., Rabinowitz, N. C., Wang, J. X., and Hill, F.
Tell me why!—Explanations support learning relational and causal structure.
In *International Conference on Learning Representations (ICLR 2022)*. International Conference on Learning Representations (ICLR), October 2022.

Lampinen, A. K., Chan, S. C. Y., Dasgupta, I., Nam, A. J., and Wang, J. X.
Passive learning of active causal strategies in agents and language models, October 2023.
arXiv:2305.16183 [cs].

Lampinen, A. K., Dasgupta, I., Chan, S. C. Y., Sheahan, H. R., Creswell, A., Kumaran, D., McClelland, J. L., and Hill, F.
Language models, like humans, show content effects on reasoning tasks.
PNAS Nexus, 3(7):pgae233, July 2024.

Langford, J. and Zhang, T.
The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information.
In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

Laurençon, H., Saulnier, L., Wang, T., Akiki, C., Villanova del Moral, A., Le Scao, T., Von Werra, L., Mou, C., González Ponferrada, E., Nguyen, H., Frohberg, J., Šaško, M., Lhoest, Q., McMillan-Major, A., Dupont, G., Biderman, S., Rogers, A., Ben allal, L., De Toni, F., Pistilli, G., Nguyen, O., Nikpoor, S., Masoud, M., Colombo, P., de la Rosa, J., Villegas, P., Thrush, T., Longpre, S., Nagel, S., Weber, L., Muñoz, M., Zhu, J., Van Strien, D., Alyafeai, Z., Almubarak, K., Vu, M. C., Gonzalez-Dios, I., Soroa, A., Lo, K., Dey, M., Ortiz Suarez, P., Gokaslan, A., Bose, S., Adelani, D., Phan, L., Tran, H., Yu, I., Pai, S., Chim, J., Lepercq, V., Ilic, S., Mitchell, M., Luccioni, S. A., and Jernite, Y.

The BigScience ROOTS Corpus: A 1.6TB Composite Multilingual Dataset.
Advances in Neural Information Processing Systems, 35:31809–31826, December 2022.

Laurençon, H., Saulnier, L., Tronchon, L., Bekman, S., Singh, A., Lozhkov, A., Wang, T., Karamcheti, S., Rush, A., Kiela, D., Cord, M., and Sanh, V.
OBELICS: An Open Web-Scale Filtered Dataset of Interleaved Image-Text Documents.
Advances in Neural Information Processing Systems, 36:71683–71702, December 2023.

Laurençon, H., Tronchon, L., Cord, M., and Sanh, V.
What matters when building vision-language models?
Advances in Neural Information Processing Systems, 37:87874–87907, December 2024.

Laversanne-Finot, A., Péré, A., and Oudeyer, P.-Y.
Curiosity Driven Exploration of Learned Disentangled Goal Spaces, November 2018.
arXiv:1807.01521 [cs].

Lazaridou, A., Gribovskaya, E., Stokowiec, W., and Grigorev, N.
Internet-augmented language models through few-shot prompting for open-domain question answering, May 2022.
arXiv:2203.05115 [cs].

Le Scao, T. and Rush, A.
How many data points is a prompt worth?
In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2627–2636, Online, June 2021. Association for Computational Linguistics.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., and Mordatch, I.
Multi-Game Decision Transformers.
Advances in Neural Information Processing Systems, 35:27921–27936, December 2022.

Lee, S., Ekpo, D., Liu, H., Huang, F., Shrivastava, A., and Huang, J.-B.
Imagine, Verify, Execute: Memory-Guided Agentic Exploration with Vision-Language Models, June 2025.
arXiv:2505.07815 [cs].

Leonard, J. A., Cordrey, S. R., Liu, H. Z., and Mackey, A. P.
Young children calibrate effort based on the trajectory of their performance.
Developmental Psychology, 59(3):609–619, 2023.
Place: US Publisher: American Psychological Association.

Levine, S., Kumar, A., Tucker, G., and Fu, J.
Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, November 2020.
arXiv:2005.01643 [cs].

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M.
Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task.
ICLR, May 2023a.
Publisher: ICLR.

Li, L., Chu, W., Langford, J., and Schapire, R. E.
A Contextual-Bandit Approach to Personalized News Article Recommendation.
In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, April 2010.
arXiv:1003.0146 [cs].

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., and Chang, K.-W.
What Does BERT with Vision Look At?
In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5265–5275, Online, July 2020. Association for Computational Linguistics.

Li, M. Y., Fox, E. B., and Goodman, N. D.
Automated statistical model discovery with language models.
In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 27791–27807, Vienna, Austria, 2024. JMLR.org.

Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., Andreas, J., Mordatch, I., Torralba, A., and Zhu, Y.
Pre-trained language models for interactive decision-making.
In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.
event-place: New Orleans, LA, USA.

Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z., and Ye, D.
A Survey on Transformers in Reinforcement Learning.
Transactions on Machine Learning Research, July 2023b.

Li, X., Zou, H., and Liu, P.
ToRL: Scaling Tool-Integrated RL, March 2025.
arXiv:2503.23383 [cs].

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A.
Code as Policies: Language Model Programs for Embodied Control.
In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, May 2023.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D.
Continuous control with deep reinforcement learning.
In Bengio, Y. and LeCun, Y. (eds.), *ICLR*, 2016.

Lin, Y., Tan, Y. C., and Frank, R.
Open Sesame: Getting inside BERT's Linguistic Knowledge.

In Linzen, T., Chrupała, G., Belinkov, Y., and Hupkes, D. (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 241–253, Florence, Italy, August 2019. Association for Computational Linguistics.

Liu, H. and Abbeel, P.
Emergent Agentic Transformer from Chain of Hindsight Experience.
In *Proceedings of the 40th International Conference on Machine Learning*, pp. 21362–21374. PMLR, July 2023.
ISSN: 2640-3498.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G.
Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.
ACM Comput. Surv., 55(9):195:1–195:35, January 2023a.

Liu, R., Wei, J., Gu, S. S., Wu, T.-Y., Vosoughi, S., Cui, C., Zhou, D., and Dai, A. M.
Mind's Eye: Grounded Language Model Reasoning through Simulation.
In *International Conference on Learning Representations (ICLR 2023)*. International Conference on Learning Representations (ICLR), September 2023b.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., and Tang, J.
AgentBench: Evaluating LLMs as Agents.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024.

Lopes, M. and Oudeyer, P.-Y.
The strategic student approach for life-long exploration and learning.
In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–8, November 2012.
ISSN: 2161-9476.

Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P.-y.
Exploration in Model-based Reinforcement Learning by Empirically Estimating Learning Progress.
In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Loya, M., Sinha, D., and Futrell, R.
Exploring the Sensitivity of LLMs' Decision-Making Capabilities: Insights from Prompt Variations and Hyperparameters.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3711–3716, Singapore, December 2023. Association for Computational Linguistics.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D.
The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, September 2024.
arXiv:2408.06292 [cs].

Lu, T., Pal, D., and Pal, M.
Contextual Multi-Armed Bandits.
In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 485–492. JMLR Workshop and Conference Proceedings, March 2010. ISSN: 1938-7228.

Lucy, L. and Gauthier, J.
Are Distributional Representations Ready for the Real World? Evaluating Word Vectors for Grounded Perceptual Meaning.
In Bansal, M., Matuszek, C., Andreas, J., Artzi, Y., and Bisk, Y. (eds.), *Proceedings of the First Workshop on Language Grounding for Robotics*, pp. 76–85, Vancouver, Canada, August 2017. Association for Computational Linguistics.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T.
A Survey of Reinforcement Learning Informed by Natural Language.
In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. International Joint Conference on Artificial Intelligence, 2019.

Lungarella, M., , Giorgio, M., , Rolf, P., , and Sandini, G.
Developmental robotics: a survey.
Connection Science, 15(4):151–190, December 2003.
Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09540090310001655110.

Maaten, L. v. d. and Hinton, G.
Visualizing Data using t-SNE.
Journal of Machine Learning Research, 9(86):2579–2605, 2008.

Madsen, A., Reddy, S., and Chandar, S.
Post-hoc Interpretability for Neural NLP: A Survey.
ACM Comput. Surv., 55(8):155:1–155:42, 2022.

Mahaut, M., Aina, L., Czarnowska, P., Hardalov, M., Müller, T., and Marquez, L.
Factual Confidence of LLMs: on Reliability and Robustness of Current Estimators.
In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4554–4570, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Mahowald, K., Ivanova, A. A., Blank, I. A., Kanwisher, N., Tenenbaum, J. B., and Fedorenko, E.
Dissociating language and thought in large language models.
Trends in Cognitive Sciences, 28(6):517–540, June 2024.
Publisher: Elsevier.

Marino, K., Rastegari, M., Farhadi, A., and Mottaghi, R.
OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge.
In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3190–3199, 2019.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J.
Teacher-Student Curriculum Learning, November 2017.
arXiv:1707.00183 [cs].

Matthews, M., Beukman, M., Lu, C., and Foerster, J. N.
Kinetix: Investigating the Training of General Agents through Open-Ended Physics-Based Control Tasks.
In *International Conference on Learning Representations (ICLR 2025)*. International Conference on Learning Representations (ICLR), October 2025.

McClelland, J. L., Hill, F., Rudolph, M., Baldridge, J., and Schütze, H.
Extending Machine Language Models toward Human-Level Language Understanding.
CoRR, abs/1912.05877, 2019.
arXiv: 1912.05877.

McCoy, R. T., Pavlick, E., and Linzen, T.
Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference.
In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

McInnes, L., Healy, J., Saul, N., and Großberger, L.
UMAP: Uniform Manifold Approximation and Projection.
Journal of Open Source Software, 3(29):861, September 2018.

Mehta, B., Diaz, M., Golemo, F., Pal, C. J., and Paull, L.
Active Domain Randomization.
In *Proceedings of the Conference on Robot Learning*, pp. 1162–1176. PMLR, May 2020. ISSN: 2640-3498.

Merrill, W., Goldberg, Y., Schwartz, R., and Smith, N. A.
Provable Limitations of Acquiring Meaning from Ungrounded Form: What Will Future Language Models Understand?
Transactions of the Association for Computational Linguistics, 9:1047–1060, September 2021.

Mikolov, T., Chen, K., Corrado, G., and Dean, J.
Efficient Estimation of Word Representations in Vector Space.
In *International Conference on Learning Representations (ICLR 2013)*. International Conference on Learning Representations (ICLR), January 2013.

Min, S. Y., Wu, Y., Sun, J., Kaufmann, M., Tajwar, F., Bisk, Y., and Salakhutdinov, R.
Self-Regulation and Requesting Interventions, February 2025.
arXiv:2502.04576 [cs].

Mirchandani, S., Karamcheti, S., and Sadigh, D.
ELLA: exploration through learned language abstraction.
In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pp. 29529–29540, Red Hook, NY, USA, 2021. Curran Associates Inc.

Mirolli, M. and Parisi, D.
Towards a Vygotskyan cognitive robotics: The role of language as a cognitive tool.
New Ideas in Psychology, 29(3):298–311, December 2011.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533, February 2015.
Publisher: Nature Publishing Group.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K.
Asynchronous Methods for Deep Reinforcement Learning.
In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1928–1937. PMLR, June 2016.
ISSN: 1938-7228.

Moerland, T. M., Broekens, J., Plaat, A., and Jonker, C. M.
Model-based Reinforcement Learning: A Survey.
Foundations and Trends® in Machine Learning, 16(1):1–118, January 2023.
Publisher: Now Publishers, Inc.

Mollo, D. C. and Millière, R.
The Vector Grounding Problem, April 2023.
arXiv:2304.01481 [cs].

Moulin-Frier, C. and Oudeyer, P.-Y.
Exploration strategies in developmental robotics: A unified probabilistic framework.
In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–6, August 2013.
ISSN: 2161-9476.

Moulin-Frier, C., Nguyen, S. M., and Oudeyer, P.-Y.
Self-organization of early vocal development in infants and machines: the role of intrinsic motivation.
Frontiers in Psychology, 4, January 2014.
Publisher: Frontiers.

Murugesan, K., Atzeni, M., Kapanipathi, P., Shukla, P., Kumaravel, S., Tesauro, G., Talamadupula, K., Sachan, M., and Campbell, M.
Text-based RL Agents with Commonsense Knowledge: New Challenges, Environments and Baselines.
Proceedings of the AAAI Conference on Artificial Intelligence, 35(10):9018–9027, May 2021.
Number: 10.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S.
Visual Reinforcement Learning with Imagined Goals.
In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., Jiang, X., Cobbe, K., Eloundou, T., Krueger, G., Button, K., Knight, M., Chess, B., and Schulman, J.
WebGPT: Browser-assisted question-answering with human feedback, June 2022.
arXiv:2112.09332 [cs].

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P.
Curriculum learning for reinforcement learning domains: a framework and survey.
J. Mach. Learn. Res., 21(1):181:7382–181:7431, January 2020.

Ni, J., Hernandez Abrego, G., Constant, N., Ma, J., Hall, K., Cer, D., and Yang, Y.
Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models.
In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J.
Gotta Learn Fast: A New Benchmark for Generalization in RL, April 2018.
arXiv:1804.03720 [cs].

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L.
Solving Rubik's Cube with a Robot Hand.
arXiv:1910.07113 [cs, stat], October 2019.
arXiv: 1910.07113.

Ortiz Suárez, P. J., Romary, L., and Sagot, B.
A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages.
In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1703–1714, Online, July 2020. Association for Computational Linguistics.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Roy, B. V., Sutton, R., Silver, D., and Hasselt, H. V.
Behaviour Suite for Reinforcement Learning.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), April 2020.

Oudeyer, P.-Y. and Kaplan, F.
Discovering communication.
Connection Science, 18(2):189–206, June 2006.
Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09540090600768567.

Oudeyer, P.-Y. and Kaplan, F.
What is intrinsic motivation? A typology of computational approaches.
Frontiers in neurorobotics, 1:108, 2007.
Publisher: Frontiers.

Oudeyer, P.-Y. and Smith, L. B.
How Evolution May Work Through Curiosity-Driven Developmental Process.
Topics in Cognitive Science, 8(2):492–502, 2016.
_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/tops.12196.

Oudeyer, P.-Y., Kachergis, G., and Schueller, W.
Computational and robotic models of early language development: A review.
In *International handbook of language acquisition*, Routledge international handbooks, pp. 76–101. Routledge/Taylor & Francis Group, New York, NY, US, 2019.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R.
Training language models to follow instructions with human feedback.
In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 27730–27744, Red Hook, NY, USA, November 2022. Curran Associates Inc.

Parisi, A., Zhao, Y., and Fiedel, N.
TALM: Tool Augmented Language Models, May 2022.
arXiv:2205.12255 [cs].

Parisotto, E., Ba, J. L., and Salakhutdinov, R.
Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning, February 2016.
arXiv:1511.06342 [cs].

Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., and Rocktäschel, T.
Evolving Curricula with Regret-Based Environment Design.
In *Proceedings of the 39th International Conference on Machine Learning*, pp. 17473–17498. PMLR, June 2022.
ISSN: 2640-3498.

Patel, R., Pavlick, E., and Tellex, S.
Grounding Language to Non-Markovian Tasks with No Supervision of Task Specifications.
Robotics: Science and Systems XVI, July 2020.
Conference Name: Robotics: Science and Systems 2020 ISBN: 9780992374761 Publisher: Robotics: Science and Systems Foundation.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T.
Curiosity-driven Exploration by Self-supervised Prediction.
In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2778–2787. PMLR, July 2017.
ISSN: 2640-3498.

Pathak, D., Gandhi, D., and Gupta, A.
Self-Supervised Exploration via Disagreement.
In *Proceedings of the 36th International Conference on Machine Learning*, pp. 5062–5071. PMLR, May 2019.
ISSN: 2640-3498.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E.
Gorilla: Large Language Model Connected with Massive APIs, May 2023.
arXiv:2305.15334 [cs].

Paulus, R., Xiong, C., and Socher, R.
A Deep Reinforced Model for Abstractive Summarization.
In *International Conference on Learning Representations (ICLR 2018)*. International Conference on Learning Representations (ICLR), February 2018.

Pavlick, E.
Symbols and grounding in large language models.
Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 381(2251):20220041, June 2023.
Publisher: Royal Society.

Pecher, D. and Zwaan, R. A. (eds.).
Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thinking.
Cambridge University Press, Cambridge, 2005.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P.
Sim-to-Real Transfer of Robotic Control with Dynamics Randomization.
In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, May 2018.
ISSN: 2577-087X.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S.
Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning.
In *International Conference on Learning Representations (ICLR 2021)*. International Conference on Learning Representations (ICLR), October 2021.

Pere, A., Forestier, S., Sigaud, O., and Oudeyer, P.-Y.
Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration.
In *International Conference on Learning Representations (ICLR 2018)*. International Conference on Learning Representations (ICLR), February 2018.

Peters, J. and Schaal, S.
Reinforcement learning by reward-weighted regression for operational space control.
In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pp. 745–750, New York, NY, USA, 2007. Association for Computing Machinery.

Petrenko, A., Huang, Z., Kumar, T., Sukhatme, G., and Koltun, V.
Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7652–7662. PMLR, November 2020.
ISSN: 2640-3498.

Piaget, J.
The construction of reality in the child.
The construction of reality in the child. Basic Books/Hachette Book Group, New York, NY, US, 1954.
Pages: xiii, 386.

Piantadosi, S. T. and Hill, F.
Meaning without reference in large language models, August 2022.
arXiv:2208.02957 [cs].

Piriyakulkij, W. T., Langenfeld, C., Le, T. A., and Ellis, K.
Doing Experiments and Revising Rules with Natural Language and Probabilistic Reasoning.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 53102–53137. Curran Associates, Inc., 2024.

Pitis, S., Chan, H., Zhao, S., Stadie, B., and Ba, J.
Maximum Entropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7750–7761. PMLR, November 2020.
ISSN: 2640-3498.

Poli, F., O'Reilly, J. X., Mars, R. B., and Hunnius, S.
Curiosity and the dynamics of optimal exploration.
Trends in Cognitive Sciences, 28(5):441–453, May 2024.
Publisher: Elsevier.

Pomerleau, D. A.
ALVINN: An Autonomous Land Vehicle in a Neural Network.
In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

Pong, V., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S.
Skew-Fit: State-Covering Self-Supervised Reinforcement Learning.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7783–7792. PMLR, November 2020.
ISSN: 2640-3498.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y.
Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments.
In *Proceedings of the Conference on Robot Learning*, pp. 835–853. PMLR, May 2020a.
ISSN: 2640-3498.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y.
Automatic Curriculum Learning For Deep RL: A Short Survey, May 2020b.
arXiv:2003.04664 [cs].

Portelas, R., Romac, C., Hofmann, K., and Oudeyer, P.-Y.
Meta Automatic Curriculum Learning.
arXiv:2011.08463 [cs], November 2020c.
arXiv: 2011.08463.

Pourcel, G., Carta, T., Kovač, G., and Oudeyer, P.-Y.
Autotelic LLM-based exploration for goal-conditioned RL, December 2024a.

Pourcel, J., Colas, C., Molinaro, G., Oudeyer, P.-Y., and Teodorescu, L.
ACES: Generating a Diversity of Challenging Programming Puzzles with Autotelic Generative Models.
Advances in Neural Information Processing Systems, 37:67627–67662, December 2024b.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C.
Neural Episodic Control.
In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2827–2836. PMLR, July 2017.
ISSN: 2640-3498.

Pulvermüller, F., Härle, M., and Hummel, F.
Walking or talking?: Behavioral and neurophysiological correlates of action verb processing.
Brain and Language, 78(2):143–168, 2001.
Place: Netherlands Publisher: Elsevier Science.

Putta, P., Mills, E., Garg, N., Motwani, S., Finn, C., Garg, D., and Rafailov, R.
Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents, August 2024.
arXiv:2408.07199 [cs].

Qassimi, S. and Rakrak, S.
Multi-objective contextual bandits in recommendation systems for smart tourism.
Scientific Reports, 15(1):13669, April 2025.
Publisher: Nature Publishing Group.

Qwen, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z.
Qwen2.5 Technical Report, January 2025.
arXiv:2412.15115 [cs].

Racaniere, S., Lampinen, A. K., Santoro, A., Reichert, D. P., Firoiu, V., and Lillicrap, T. P.
Automated curricula through setter-solver interactions, January 2020.
arXiv:1909.12892 [cs].

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I.
Improving language understanding by generative pre-training, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I.
Language Models are Unsupervised Multitask Learners, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I.
Learning Transferable Visual Models From Natural Language Supervision.
arXiv:2103.00020 [cs], February 2021.
arXiv: 2103.00020.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., Driessche, G. v. d., Hendricks, L. A., Rauh, M., Huang, P.-S., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X. L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J.-B., Tsimpoukelli, M., Grigorev, N., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., d'Autume, C. d. M., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., Casas, D. d. L., Guy, A., Jones, C., Bradbury, J., Johnson, M., Hechtman, B., Weidinger, L., Gabriel, I., Isaac, W., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K., Stanway, J., Bennett, L., Hassabis, D., Kavukcuoglu, K., and Irving, G.
Scaling Language Models: Methods, Analysis & Insights from Training Gopher.
arXiv:2112.11446 [cs], January 2022.
arXiv: 2112.11446.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C.
Direct preference optimization: your language model is secretly a reward model.
In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pp. 53728–53741, Red Hook, NY, USA, 2023. Curran Associates Inc.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J.
Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.
Journal of Machine Learning Research, 21(140):1–67, 2020.

Raileanu, R. and Rocktäschel, T.
RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), September 2020.

Rajaraman, N., Yang, L., Jiao, J., and Ramchandran, K.
Toward the Fundamental Limits of Imitation Learning.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 2914–2924. Curran Associates, Inc., 2020.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S.

EPOpt: Learning Robust Neural Network Policies Using Model Ensembles.
In *International Conference on Learning Representations (ICLR 2017)*. International Conference on Learning Representations (ICLR), February 2017.

Ramamurthy, R., Ammanabrolu, P., Brantley, K., Hessel, J., Sifa, R., Bauckhage, C., Hajishirzi, H., and Choi, Y.
Is Reinforcement Learning (Not) for Natural Language Processing: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization.
In *International Conference on Learning Representations (ICLR 2023)*. International Conference on Learning Representations (ICLR), January 2023.

Rame, A., Couairon, G., Dancette, C., Gaya, J.-B., Shukor, M., Soulier, L., and Cord, M.
Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards.
Advances in Neural Information Processing Systems, 36:71095–71134, December 2023.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W.
Sequence Level Training with Recurrent Neural Networks, May 2016.
arXiv:1511.06732 [cs].

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and Freitas, N. d.
A Generalist Agent.
Transactions on Machine Learning Research, August 2022.

Reid, M., Yamada, Y., and Gu, S. S.
Can Wikipedia Help Offline Reinforcement Learning?
arXiv:2201.12122 [cs], January 2022.
arXiv: 2201.12122.

Reinke, C., Etcheverry, M., and Oudeyer, P.-Y.
Intrinsically Motivated Discovery of Diverse Patterns in Self-Organizing Systems.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), April 2020.

Riedmiller, M.
Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method.
In Gama, J., Camacho, R., Brazdil, P. B., Jorge, A. M., and Torgo, L. (eds.), *Machine Learning: ECML 2005*, pp. 317–328, Berlin, Heidelberg, 2005. Springer.

Risi, S. and Togelius, J.
Increasing Generality in Machine Learning through Procedural Content Generation, March 2020.
arXiv:1911.13071 [cs].

Rogers, A., Kovaleva, O., and Rumshisky, A.
A Primer in BERTology: What We Know About How BERT Works.

Transactions of the Association for Computational Linguistics, 8:842–866, 2020. Place: Cambridge, MA Publisher: MIT Press.

Roy, D.
Grounding words in perception and action: computational insights.
Trends in Cognitive Sciences, 9(8):389–396, August 2005a.

Roy, D.
Semiotic schemas: A framework for grounding language in action and perception.
Artificial Intelligence, 167(1):170–205, September 2005b.

Russell, S. and Norvig, P.
Artificial Intelligence: A Modern Approach.
Prentice Hall Press, USA, 3rd edition, November 2009.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R.
Policy Distillation, January 2016.
arXiv:1511.06295 [cs].

Rutherford, A., Beukman, M., Willi, T., Lacerda, B., Hawes, N., and Foerster, J. N.
No Regrets: Investigating and Improving Regret Approximations for Curriculum Discovery.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), November 2024.

Salimans, T. and Chen, R.
Learning Montezuma's Revenge from a Single Demonstration, December 2018.
arXiv:1812.03381 [cs].

Salinas, A. and Morstatter, F.
The Butterfly Effect of Altering Prompts: How Small Changes and Jailbreaks Affect Large Language Model Performance.
In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4629–4651, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Sancaktar, C., Gumbsch, C., Zadaianchuk, A., Kolev, P., and Martius, G.
SENSEI: Semantic Exploration Guided by Foundation Models to Learn Versatile World Models.
In *International Conference on Learning Representations (ICLR 2025)*. International Conference on Learning Representations (ICLR), June 2025.

Sarti, G., Feldhus, N., Sickert, L., and van der Wal, O.
Inseq: An Interpretability Toolkit for Sequence Generation Models.
In Bollegala, D., Huang, R., and Ritter, A. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 421–435, Toronto, Canada, July 2023. Association for Computational Linguistics.

Sayalı, C., Heling, E., and Cools, R.
Learning progress mediates the link between cognitive effort and task engagement. *Cognition*, 236:105418, July 2023.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ili'c, S., Hesslow, D., Castagn'e, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurenccon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Etxabe, A. S., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C. C., Klamm, C., Leong, C., van Strien, D. A., Adelani, D. I., Radev, D. R., Ponferrada, E. G., Levkovizh, E., Kim, E., Natan, E., Toni, F. D., Dupont, G., Kruszewski, G., Pistilli, G., ElSahar, H., Benyamina, H., Tran, H. T., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., von Werra, L., Weber, L., Phan, L., Allal, L. B., Tanguy, L., Dey, M., Muñoz, M. R., Masoud, M., Grandury, M., vSavsko, M., Huang, M., Coavoux, M., Singh, M., Jiang, M. T.-J., Vu, M. C., mad A. Jauhar, M., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., L'opez, R., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S. H., Sharma, S. S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T. T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Salesky, E., Mielke, S. J., Lee, W. Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J. A., Rozen, J., Gao, L., Sutawika, L., Bari, M. S., Al-shaibani, M. S., Manica, M., Nayak, N. V., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S. H., Kim, T., Bers, T., Févry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H. W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavall'ee, P. F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., N'ev'eol, A., Lovering, C., Garrette, D., Tunuguntla, D. R., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G. I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J. Z., Tang, X., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Mirkin, S., Pais, S. O., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Kasner, Z., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A. S. R., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B. A., Saxena, B. K., Ferrandis, C. M., Contractor, D., Lansky, D. M., David, D., Kiela, D., Nguyen, D. A., Tan, E., Baylor, E., inwanne Ozoani, E., Mirza, F. T., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J. B., Fort, K., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick,

M. K., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R. P., Hao, R., Alizadeh, S., Shubber, S., Wang, S. L., Roy, S., Viguier, S., Le, T.-C., Oyebade, T., Le, T. N. H., Yang, Y., Nguyen, Z., Kashyap, A. R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A. K., Beilharz, B., Wang, B., de Brito, C. M. F., Zhou, C., Jain, C., Xu, C., Fourrier, C., Perin'an, D. L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H. U., Bello, I. I., Dash, I., Kang, J. S., Giorgi, J., Golde, J., Posada, J. D., Sivaraman, K., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M. H., Takeuchi, M., Pàmies, M., Castillo, M. A., Nezhurina, M., Sanger, M., Samwald, M., Cullan, M., Weinberg, M., Wolf, M., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N. M., Muellner, N., Fung, P., Haller, P., Haller, P., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S. S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S. P., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z. X., Xie, Z., Ye, Z., Bras, M., Belkada, Y., and Wolf, T.
BLOOM: A 176B-Parameter Open-Access Multilingual Language Model, November 2022.

Schaul, T., Horgan, D., Gregor, K., and Silver, D.
Universal Value Function Approximators.
In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1312–1320. PMLR, June 2015.
ISSN: 1938-7228.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T.
Toolformer: Language Models Can Teach Themselves to Use Tools, February 2023.
arXiv:2302.04761 [cs].

Schmidhuber, J.
Curious model-building control systems.
In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pp. 1458–1463 vol.2, Singapore, 1991a. IEEE.

Schmidhuber, J.
A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers.
Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats, pp. 222–227, 1991b.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D.
Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model.
Nature, 588(7839):604–609, December 2020.
arXiv:1911.08265 [cs].

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P.
Trust Region Policy Optimization.

In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897. PMLR, June 2015.
ISSN: 1938-7228.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.
Proximal Policy Optimization Algorithms.
arXiv:1707.06347 [cs], August 2017.
arXiv: 1707.06347.

Schulz, L.
Chapter Ten - Finding New Facts; Thinking New Thoughts.
In Xu, F. and Kushnir, T. (eds.), *Advances in Child Development and Behavior*, volume 43 of *Rational Constructivism in Cognitive Development*, pp. 269–294. JAI, January 2012.

Schuster, M. and Nakajima, K.
Japanese and Korean voice search.
In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, March 2012.
ISSN: 2379-190X.

Schuster, M. and Paliwal, K.
Bidirectional recurrent neural networks.
IEEE Transactions on Signal Processing, 45(11):2673–2681, November 1997.

Scialom, T.
Natural Language Generation with Reinforcement Learning.
phdthesis, Sorbonne Universite, July 2022.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A.
Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024.

Scorolli, C. and Borghi, A. M.
Sentence comprehension and action: Effector specific modulation of the motor system.
Brain Research, 1130:119–124, January 2007.

Searle, J. R.
Minds, brains, and programs.
Behavioral and Brain Sciences, 3(3):417–424, September 1980.

Selfridge, O. G., Sutton, R. S., and Barto, A. G.
Training and tracking in robotics.
In *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1*, IJCAI'85, pp. 670–672, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc.

Sennrich, R., Haddow, B., and Birch, A.
Neural Machine Translation of Rare Words with Subword Units.
In Erk, K. and Smith, N. A. (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

Sharma, P., Ding, N., Goodman, S., and Soricut, R.
Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning.
In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S.
Reflexion: language agents with verbal reinforcement learning.
In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pp. 8634–8652, Red Hook, NY, USA, 2023. Curran Associates Inc.

Shojaee, P., Mirzadeh, I., Alizadeh, K., Horton, M., Bengio, S., and Farajtabar, M.
The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity, June 2025.
arXiv:2506.06941 [cs].

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M.
ALFWorld: Aligning Text and Embodied Environments for Interactive Learning.
In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., Zouitine, A., Aractingi, M., Pascal, C., Russi, M., Marafioti, A., Alibert, S., Cord, M., Wolf, T., and Cadene, R.
SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics, June 2025.
arXiv:2506.01844 [cs].

Sigaud, O., Baldassarre, G., Colas, C., Doncieux, S., Duro, R., Oudeyer, P.-Y., Perrin-Gilbert, N., and Santucci, V. G.
A Definition of Open-Ended Learning Problems for Goal-Conditioned Agents, June 2024.
arXiv:2311.00344 [cs].

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D.
Mastering the game of Go with deep neural networks and tree search.
Nature, 529(7587):484–489, January 2016.
Publisher: Nature Publishing Group.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A. T., Kumar, A., Alemi, A. A., Rizkowsky, A., Nova, A., Adlam, B., Bohnet, B., Elsayed, G. F., Sedghi, H., Mordatch, I., Simpson, I., Gur, I., Snoek, J., Pennington, J., Hron, J., Kenealy, K., Swersky, K., Mahajan, K., Culp, L. A., Xiao, L., Bileschi, M., Constant, N., Novak, R., Liu, R., Warkentin, T., Bansal, Y., Dyer, E., Neyshabur, B., Sohl-Dickstein, J., and Fiedel, N.
Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models.
Transactions on Machine Learning Research, January 2024.

Smith, L. and Gasser, M.
The Development of Embodied Cognition: Six Lessons from Babies.
Artificial Life, 11(1-2):13–29, January 2005.

Snell, C. V., Kostrikov, I., Su, Y., Yang, S., and Levine, S.
Offline RL for Natural Language Generation with Implicit Language Q Learning.
In *International Conference on Learning Representations (ICLR 2023)*. International Conference on Learning Representations (ICLR), September 2023.

Song, D. R., Yang, C., McGreavy, C., and Li, Z.
Recurrent Deterministic Policy Gradient Method for Bipedal Locomotion on Rough Terrain Challenge.
In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 311–318, November 2018.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M., and Botvinick, M. M.
V-MPO: On-Policy Maximum a Posteriori Policy Optimization for Discrete and Continuous Control.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), September 2020a.

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B.
Observational Overfitting in Reinforcement Learning.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), September 2020b.

Song, X., Salcianu, A., Song, Y., Dopson, D., and Zhou, D.
Fast WordPiece Tokenization.
In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2089–2103, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Stanley, K. O.
Compositional pattern producing networks: A novel abstraction of development.
Genetic Programming and Evolvable Machines, 8(2):131–162, June 2007.

Stanton, C. and Clune, J.

Deep Curiosity Search: Intra-Life Exploration Can Improve Performance on Challenging Deep Reinforcement Learning Problems, November 2018.
arXiv:1806.00553 [cs].

Steels, L.
Language games for autonomous robots.
IEEE Intelligent Systems, 16(5):16–22, September 2001.

Steels, L.
The Autotelic Principle.
In Iida, F., Pfeifer, R., Steels, L., and Kuniyoshi, Y. (eds.), *Embodied Artificial Intelligence: International Seminar, Dagstuhl Castle, Germany, July 7-11, 2003. Revised Papers*, pp. 231–242. Springer, Berlin, Heidelberg, 2004.

Steels, L. and Kaplan, F.
AIBO's first words: The social learning of language and meaning.
Evolution of Communication, 4(1):3–32, April 2002.
Publisher: John Benjamins Publishing Company.

Steyvers, M. and Peters, M. A. K.
Metacognition and Uncertainty Communication in Humans and Large Language Models, April 2025.
arXiv:2504.14045 [cs].

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F.
Learning to summarize with human feedback.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020.

Still, S. and Precup, D.
An information-theoretic approach to curiosity-driven reinforcement learning.
Theory in Biosciences, 131(3):139–148, September 2012.

Stout, A. and Barto, A. G.
Competence progress intrinsic motivation.
In *2010 IEEE 9th International Conference on Development and Learning*, pp. 257–262, August 2010.
ISSN: 2161-9476.

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., and Dai, J.
VL-BERT: Pre-training of Generic Visual-Linguistic Representations.
In *International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (ICLR), September 2020.

Sukhija, B., Treven, L., Sferrazza, C., Dorfler, F., Abbeel, P., and Krause, A.
Optimism via Intrinsic Rewards: Scalable and Principled Exploration for Model-based Reinforcement Learning, February 2025.

Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C.
VideoBERT: A Joint Model for Video and Language Representation Learning.
In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7463–7472, 2019.

Sutskever, I., Vinyals, O., and Le, Q. V.
Sequence to sequence learning with neural networks.
In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, volume 2 of *NIPS'14*, pp. 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

Sutton, R. S. and Barto, A. G.
Reinforcement Learning: An Introduction.
A Bradford Book, Cambridge, MA, USA, October 2018.

Szot, A., Schwarzer, M., Agrawal, H., Mazoure, B., Metcalf, R., Talbott, W., Mackraz, N., Hjelm, R. D., and Toshev, A. T.
Large Language Models as Generalizable Policies for Embodied Tasks.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024.

Takagi, S.
On the effect of pre-training for transformer in different modality on offline reinforcement learning.
In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 30678–30692, Red Hook, NY, USA, November 2022. Curran Associates Inc.

Tamari, R., Shani, C., Hope, T., Petruck, M. R. L., Abend, O., and Shahaf, D.
Language (Re)modelling: Towards Embodied Language Understanding.
In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6268–6281, Online, July 2020. Association for Computational Linguistics.

Tan, W., Zhang, W., Liu, S., Zheng, L., Wang, X., and An, B.
True Knowledge Comes from Practice: Aligning Large Language Models with Embodied Environments via Reinforcement Learning.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024.

Tang, H., Key, D., and Ellis, K.
WorldCoder, a Model-Based LLM Agent: Building World Models by Writing Code and Interacting with the Environment.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 70148–70212. Curran Associates, Inc., 2024.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M.
DeepMind Control Suite, January 2018.
arXiv:1801.00690 [cs].

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., Grill, J.-b., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C. J., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., Ji, J.-y., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L.
Gemma 3 Technical Report, March 2025.
arXiv:2503.19786 [cs].

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R.
Distral: Robust multitask reinforcement learning.
In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Ten, A., Kaushik, P., Oudeyer, P.-Y., and Gottlieb, J.
Humans monitor learning progress in curiosity-driven exploration.
Nature Communications, 12(1):5972, October 2021.
Publisher: Nature Publishing Group.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D.
How to Grow a Mind: Statistics, Structure, and Abstraction.

Science, 331(6022):1279–1285, March 2011.
Publisher: American Association for the Advancement of Science.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S. R., Das, D., and Pavlick, E.
What do you learn from context? Probing for sentence structure in contextualized word representations.
In *International Conference on Learning Representations (ICLR 2019)*. International Conference on Learning Representations (ICLR), September 2019.

Thill, S., Padó, S., and Ziemke, T.
On the Importance of a Rich Embodiment in the Grounding of Concepts: Perspectives From Embodied Cognitive Science and Computational Linguistics.
Topics in Cognitive Science, 6(3):545–558, 2014.
_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/tops.12093.

Tian, K., Mitchell, E., Zhou, A., Sharma, A., Rafailov, R., Yao, H., Finn, C., and Manning, C.
Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, Singapore, December 2023. Association for Computational Linguistics.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P.
Domain randomization for transferring deep neural networks from simulation to the real world.
In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, September 2017.
ISSN: 2153-0866.

Todorov, E., Erez, T., and Tassa, Y.
MuJoCo: A physics engine for model-based control.
In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, October 2012.
ISSN: 2153-0866.

Tomasello, M.
Constructing a Language: A Usage-Based Theory of Language Acquisition.
Harvard University Press, 2003.

Tomasello, M.
Becoming Human: A Theory of Ontogeny.
Harvard University Press, January 2019.
Google-Books-ID: ZnhyDwAAQBAJ.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V.,

Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T.
Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023.
arXiv:2307.09288 [cs].

Tsimpoukelli, M., Menick, J. L., Cabi, S., Eslami, S. M. A., Vinyals, O., and Hill, F.
Multimodal Few-Shot Learning with Frozen Language Models.
In *Advances in Neural Information Processing Systems*, volume 34, pp. 200–212. Curran Associates, Inc., 2021.

Tsividis, P. A., Loula, J., Burga, J., Foss, N., Campero, A., Pouncy, T., Gershman, S. J., and Tenenbaum, J. B.
Human-Level Reinforcement Learning through Theory-Based Modeling, Exploration, and Planning, July 2021.
arXiv:2107.12544 [cs].

Turgay, E., Oner, D., and Tekin, C.
Multi-objective Contextual Bandit Problem with Similarity Information.
In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pp. 1673–1681. PMLR, March 2018.
ISSN: 2640-3498.

Turian, J., Ratinov, L.-A., and Bengio, Y.
Word Representations: A Simple and General Method for Semi-Supervised Learning.
In Hajič, J., Carberry, S., Clark, S., and Nivre, J. (eds.), *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Turing, A. M.
I.—COMPUTING MACHINERY AND INTELLIGENCE.
Mind, LIX(236):433–460, October 1950.

Turney, P. D. and Pantel, P.
From Frequency to Meaning: Vector Space Models of Semantics.
Journal of Artificial Intelligence Research, 37:141–188, February 2010.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I.
Solving math word problems with process- and outcome-based feedback, November 2022.
arXiv:2211.14275 [cs].

Ullman, T., Goodman, N., and Tenenbaum, J.
Theory Acquisition as Stochastic Search.
Proceedings of the Annual Meeting of the Cognitive Science Society, 32(32), 2010.

Ullman, T. D. and Tenenbaum, J. B.
Bayesian Models of Conceptual Development: Learning as Building Models of the World.
Annual Review of Developmental Psychology, 2(Volume 2, 2020):533–558, December 2020.
Publisher: Annual Reviews.

Vafa, K., Chen, J. Y., Rambachan, A., Kleinberg, J., and Mullainathan, S.
Evaluating the World Model Implicit in a Generative Model.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 26941–26975. Curran Associates, Inc., 2024.

Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N.
Finite-Time Analysis of Kernelised Contextual Bandits, September 2013.
arXiv:1309.6869 [cs].

Valmeekam, K., Olmo, A., Sreedharan, S., and Kambhampati, S.
Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change), November 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I.
Attention is All you Need.
In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Venkattaramanujam, S., Crawford, E., Doan, T., and Precup, D.
Self-supervised Learning of Distance Functions for Goal-Conditioned Reinforcement Learning, June 2020.
arXiv:1907.02998 [cs].

Vygotsky, L. S.
Thought and Language.
MIT Press, Cambridge, MA, USA, January 1962.

Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S.
Universal Adversarial Triggers for Attacking and Analyzing NLP.
In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2153–2162, Hong Kong, China, November 2019. Association for Computational Linguistics.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A.
Voyager: An Open-Ended Embodied Agent with Large Language Models.
Transactions on Machine Learning Research, November 2023a.

Wang, J., Ming, Y., Shi, Z., Vineet, V., Wang, X., Li, Y., and Joshi, N.

Is A Picture Worth A Thousand Words? Delving Into Spatial Reasoning for Vision Language Models.
Advances in Neural Information Processing Systems, 37:75392–75421, December 2024a.

Wang, L., Yang, F., Zhang, C., Lu, J., Qian, J., He, S., Zhao, P., Qiao, B., Huang, H., Qin, S., Su, Q., Ye, J., Zhang, Y., Lou, J.-G., Lin, Q., Rajmohan, S., Zhang, D., and Zhang, Q.
Large Action Models: From Inception to Implementation.
Transactions on Machine Learning Research, January 2025.

Wang, R. and Ammanabrolu, P.
A Practitioner's Guide to Multi-turn Agentic Reinforcement Learning, December 2025. arXiv:2510.01132 [cs].

Wang, R., Lehman, J., Clune, J., and Stanley, K. O.
Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions, February 2019. arXiv:1901.01753 [cs].

Wang, R., Lehman, J., Rawal, A., Zhi, J., Li, Y., Clune, J., and Stanley, K.
Enhanced POET: Open-ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9940–9951. PMLR, November 2020.
ISSN: 2640-3498.

Wang, R., Jansen, P., Côté, M.-A., and Ammanabrolu, P.
ScienceWorld: Is your Agent Smarter than a 5th Grader?
In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

Wang, R., Zelikman, E., Poesia, G., Pu, Y., Haber, N., and Goodman, N.
Hypothesis Search: Inductive Reasoning with Language Models.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024b.

Wang, X., Chen, Y., Yuan, L., Zhang, Y., Li, Y., Peng, H., and Ji, H.
Executable code actions elicit better LLM agents.
In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 50208–50232, Vienna, Austria, 2024c. JMLR.org.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W.
MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark.
Advances in Neural Information Processing Systems, 37:95266–95290, December 2024d.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and Freitas, N. d.
Sample Efficient Actor-Critic with Experience Replay.

In *International Conference on Learning Representations (ICLR 2017)*. International Conference on Learning Representations (ICLR), February 2017.

Wang, Z., Cai, S., Chen, G., Liu, A., Ma, X., and Liang, Y.
Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents.
In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, November 2023b.

Wang, Z., Cai, S., Liu, A., Jin, Y., Hou, J., Zhang, B., Lin, H., He, Z., Zheng, Z., Yang, Y., Ma, X., and Liang, Y.
JARVIS-1: Open-World Multi-task Agents with Memory-Augmented Multimodal Language Models, November 2023c.
arXiv:2311.05997 [cs].

Warde-Farley, D., Wiele, T. V. d., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V.
Unsupervised Control Through Non-Parametric Discriminative Rewards.
In *International Conference on Learning Representations (ICLR 2019)*. International Conference on Learning Representations (ICLR), September 2019.

Watkins, C. J. C. H. and Dayan, P.
Q-learning.
Machine Learning, 8(3):279–292, May 1992.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W.
Emergent Abilities of Large Language Models.
Transactions on Machine Learning Research, June 2022.

Weizenbaum, J.
ELIZA—a computer program for the study of natural language communication between man and machine.
Communications of the ACM, 9(1):36–45, January 1966.

Wen, M., Deng, C., Wang, J., Zhang, W., and Wen, Y.
Entropy-Regularized Token-Level Policy Optimization for Large Language Models, February 2024a.
arXiv:2402.06700 [cs].

Wen, M., Wan, Z., Zhang, W., Wang, J., and Wen, Y.
Reinforcing Language Agents via Policy Optimization with Action Decomposition, May 2024b.
arXiv:2405.15821 [cs].

White, D. J.
A Survey of Applications of Markov Decision Processes.
Journal of the Operational Research Society, 44(11):1073–1096, November 1993.
Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/jors.1993.181.

White, R. W.
Motivation reconsidered: The concept of competence.
Psychological Review, 66(5):297–333, 1959.
Place: US Publisher: American Psychological Association.

Williams, R. J.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine Learning, 8(3):229–256, May 1992.

Wilson, M.
Six views of embodied cognition.
Psychonomic Bulletin & Review, 9(4):625–636, December 2002.

Wimmer, H. and Perner, J.
Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception.
Cognition, 13(1):103–128, January 1983.

Winograd, T.
Procedures as a Representation for Data in a Computer Program for Understanding Natural Language.
January 1971.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A.
Transformers: State-of-the-Art Natural Language Processing.
In Liu, Q. and Schlangen, D. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics.

Wood, D., Bruner, J. S., and Ross, G.
The Role of Tutoring in Problem Solving.
Journal of Child Psychology and Psychiatry, 17(2):89–100, 1976.
_eprint: https://acamh.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-7610.1976.tb00381.x.

Wu, Z., Qiu, L., Ross, A., Akyürek, E., Chen, B., Wang, B., Kim, N., Andreas, J., and Kim, Y.
Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks.
In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1819–1862, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

Xiang, J., Tao, T., Gu, Y., Shu, T., Wang, Z., Yang, Z., and Hu, Z.
Language Models Meet World Models: Embodied Experiences Enhance Language Models.
Advances in Neural Information Processing Systems, 36:75392–75412, December 2023.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T.
OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments.
Advances in Neural Information Processing Systems, 37:52040–52094, December 2024.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D.
The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search, April 2025.
arXiv:2504.08066 [cs].

Yan, X., Song, Y., Cui, X., Christianos, F., Zhang, H., Mguni, D. H., and Wang, J.
Ask more, know better: Reinforce-Learned Prompt Questions for Decision Making with Large Language Models, October 2023.
arXiv:2310.18127 [cs].

Yang, J., Prabhakar, A., Narasimhan, K., and Yao, S.
InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback, June 2023.
arXiv:2306.14898 [cs].

Yang, R., Xu, H., WU, Y., and Wang, X.
Multi-Task Reinforcement Learning with Soft Modularization.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 4767–4777. Curran Associates, Inc., 2020.

Yao, S., Rao, R., Hausknecht, M., and Narasimhan, K.
Keep CALM and Explore: Language Models for Action Generation in Text-based Games.
In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8736–8754, Online, November 2020. Association for Computational Linguistics.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y.
ReAct: Synergizing Reasoning and Acting in Language Models, October 2022.
arXiv:2210.03629 [cs].

Yao, W., Heinecke, S., Niebles, J. C., Liu, Z., Feng, Y., Xue, L., N, R. R., Chen, Z., Zhang, J., Arpit, D., Xu, R., Mui, P. L., Wang, H., Xiong, C., and Savarese, S.
Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024.

Ying, L., Collins, K. M., Sharma, P., Colas, C., Zhao, K. I., Weller, A., Tavares, Z., Isola, P., Gershman, S. J., Andreas, J. D., Griffiths, T. L., Chollet, F., Allen, K. R., and Tenenbaum, J. B.
Assessing Adaptive World Models in Machines with Novel Games, July 2025.
arXiv:2507.12821 [cs].

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S.
Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning.
In *Proceedings of the Conference on Robot Learning*, pp. 1094–1100. PMLR, May 2020. ISSN: 2640-3498.

Yue, M., Zhao, J., Zhang, M., Du, L., and Yao, Z.
Large Language Model Cascades with Mixture of Thought Representations for Cost-Efficient Reasoning.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024.

Yun, T., Sun, C., and Pavlick, E.
Does Vision-and-Language Pretraining Improve Lexical Grounding?, September 2021. arXiv:2109.10246 [cs].

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R.
Deconvolutional networks.
In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2528–2535, June 2010.
ISSN: 1063-6919.

Zeng, A., Attarian, M., Ichter, B., Choromanski, K. M., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M. S., Sindhwani, V., Lee, J., Vanhoucke, V., and Florence, P.
Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language.
In *International Conference on Learning Representations (ICLR 2023)*. International Conference on Learning Representations (ICLR), September 2023.

Zhai, Y., Bai, H., Lin, Z., Pan, J., Tong, S., Zhou, Y., Suhr, A., Xie, S., LeCun, Y., Ma, Y., and Levine, S.
Fine-Tuning Large Vision-Language Models as Decision-Making Agents via Reinforcement Learning.
In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, November 2025.

Zhang, A. L., Nguyen, K. X., Tuyls, J., Lin, A., and Narasimhan, K. R.
Language-guided World Models: A Model-based Approach to AI Control.
In *Proceedings of the 4th Workshop on Spatial Language Understanding and Grounded Communication for Robotics (SpLU-RoboNLP 2024)*, pp. 1–16. Association for Computational Linguistics, 2024a.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S.
A Study on Overfitting in Deep Reinforcement Learning, April 2018. arXiv:1804.06893 [cs].

Zhang, C., Yang, Z., Liu, J., Li, Y., Han, Y., Chen, X., Huang, Z., Fu, B., and Yu, G.
AppAgent: Multimodal Agents as Smartphone Users.
In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, pp. 1–20, New York, NY, USA, 2025. Association for Computing Machinery.

Zhang, J., Lehman, J., Stanley, K., and Clune, J.
OMNI: Open-endedness via Models of human Notions of Interestingness.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024b.

Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., Choi, Y., and Gao, J.
VinVL: Revisiting Visual Representations in Vision-Language Models.
In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5575–5584, 2021.

Zhang, Y., Abbeel, P., and Pinto, L.
Automatic Curriculum Learning through Value Disagreement.
In *Advances in Neural Information Processing Systems*, volume 33, pp. 7648–7659. Curran Associates, Inc., 2020.

Zhao, W., Queralta, J. P., and Westerlund, T.
Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey.
In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744, December 2020.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S.
Calibrate Before Use: Improving Few-shot Performance of Language Models.
In *Proceedings of the 38th International Conference on Machine Learning*, pp. 12697–12706. PMLR, July 2021.
ISSN: 2640-3498.

Zheng, Q., Zhang, A., and Grover, A.
Online Decision Transformer.
In *Proceedings of the 39th International Conference on Machine Learning*, pp. 27042–27059. PMLR, June 2022.
ISSN: 2640-3498.

Zhong, V., Hanjie, A. W., Wang, S. I., Narasimhan, K., and Zettlemoyer, L.
SILG: the multi-environment symbolic interactive language grounding benchmark.
In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pp. 21505–21519, Red Hook, NY, USA, 2021. Curran Associates Inc.

Zhou, D., Li, L., and Gu, Q.
Neural Contextual Bandits with UCB-based Exploration, July 2020.
arXiv:1911.04462 [cs].

Zhou, K., Jurafsky, D., and Hashimoto, T.
Navigating the Grey Area: How Expressions of Uncertainty and Overconfidence Affect Language Models.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5506–5524, Singapore, December 2023. Association for Computational Linguistics.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G.
WebArena: A Realistic Web Environment for Building Autonomous Agents.
In *International Conference on Learning Representations (ICLR 2024)*. International Conference on Learning Representations (ICLR), October 2024a.

Zhou, Y., Zanette, A., Pan, J., Levine, S., and Kumar, A.
ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL.
In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, February 2024b.

Zhou, Z., Atreya, P., Lee, A., Walke, H., Mees, O., and Levine, S.
Autonomous Improvement of Instruction Following Skills via Foundation Models, October 2024c.
arXiv:2407.20635 [cs].

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G.
Fine-Tuning Language Models from Human Preferences, January 2020.
arXiv:1909.08593 [cs].

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P. R., Salazar, G., Ryoo, M. S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T.-W. E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N. J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K. A., Driess, D., Ding, T., Choromanski, K. M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M. G., and Han, K.
RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control.
In *Proceedings of The 7th Conference on Robot Learning*, pp. 2165–2183. PMLR, December 2023.
ISSN: 2640-3498.