
Large **Language Models**

Clément Romic (Hugging Face & Inria)

clement.romac@inria.fr

<https://clementromac.github.io/teaching/>

Petit Quizz 🏆

<https://279vrcxd7nq.typeform.com/to/fb1uNKxD>



- Les RNNs:
 - souffrent de gradient vanishing
 - sont très bien adaptés au très longs exemples
 - ne permettent pas de générer du texte

Petit Quizz

Petit Quizz

- Les RNNs:
 - souffrent de gradient vanishing
 - sont très bien adaptés au très longs exemples
 - ne permettent pas de générer du texte
- Le mécanisme d'attention:
 - se base sur une moyenne pondérée d'entrées
 - a été très utilisé pour des problèmes de traduction
 - est apparu avec le papier introduisant l'architecture Transformer

Petit Quizz

- Les RNNs:
 - souffrent de gradient vanishing
 - sont très bien adaptés au très longs exemples
 - ne permettent pas de générer du texte
- Le mécanisme d'attention:
 - se base sur une moyenne pondérée d'entrées
 - a été très utilisé pour des problèmes de traduction
 - est apparu avec le papier introduisant l'architecture Transformer
- Un Transformer:
 - est constitué d'un Encoder et un Decoder
 - utilise des LSTMs
 - n'est constitué que de couches de Self-Attention (aucune autre architecture de réseau de neurones)

Petit Quizz

- Les RNNs:
 - souffrent de gradient vanishing
 - sont très bien adaptés au très longs exemples
 - ne permettent pas de générer du texte
- Le mécanisme d'attention:
 - se base sur une moyenne pondérée d'entrées
 - a été très utilisé pour des problèmes de traduction
 - est apparu avec le papier introduisant l'architecture Transformer
- Un Transformer:
 - est constitué d'un Encoder et un Decoder
 - utilise des LSTMs
 - n'est constitué que de couches de Self-Attention (aucune autre architecture de réseau de neurones)
- Un Transformer:
 - contient pour chaque bloc plusieurs têtes donc les sorties sont concaténées (appelé multi-head)
 - contient un Decoder qui utilise les sorties de l'encoder comme Queries
 - contient plusieurs blocs "empilés" (appelé multi-hop)

Petit Quizz

- Les RNNs:
 - souffrent de gradient vanishing
 - sont très bien adaptés au très longs exemples
 - ne permettent pas de générer du texte
- Le mécanisme d'attention:
 - se base sur une moyenne pondérée d'entrées
 - a été très utilisé pour des problèmes de traduction
 - est apparu avec le papier introduisant l'architecture Transformer
- Un Transformer:
 - est constitué d'un Encoder et un Decoder
 - utilise des LSTMs
 - n'est constitué que de couches de Self-Attention (aucune autre architecture de réseau de neurones)
- Un Transformer:
 - contient pour chaque bloc plusieurs têtes donc les sorties sont concaténées (appelé multi-head)
 - contient un Decoder qui utilise les sorties de l'encoder comme Queries
 - contient plusieurs blocs "empilés" (appelé multi-hop)
- Dans un Transformer:
 - l'Encoder est "bi-directionnel" (toute la séquence est utilisée, les opérations sur chaque token utilisent les tokens d'avant mais aussi ceux d'après)
 - Le Decoder est "bi-directionnel" (toute la séquence est utilisée, les opérations sur chaque token utilisent les tokens d'avant mais aussi ceux d'après)

Petit Quizz

- Les RNNs:
 - souffrent de gradient vanishing
 - sont très bien adaptés au très longs exemples
 - ne permettent pas de générer du texte
- Le mécanisme d'attention:
 - se base sur une moyenne pondérée d'entrées
 - a été très utilisé pour des problèmes de traduction
 - est apparu avec le papier introduisant l'architecture Transformer
- Un Transformer:
 - est constitué d'un encoder et un decoder
 - utilise des LSTMs
 - ne constitue que des couches de Self-Attention (aucune autre architecture de réseau de neurones)
- Un Transformer:
 - contient pour chaque bloc plusieurs têtes donc les sorties sont concaténées (appelé multi-head)
 - contient un Decoder qui utilise les sorties de l'encoder comme Queries
 - contient plusieurs blocs "empilés" (appelé multi-hop)
- Dans un Transformer:
 - l'Encoder est "bi-directionnel" (toute la séquence est utilisée, les opérations sur chaque token utilisent les tokens d'avant mais aussi ceux d'après)
 - Le Decoder est "bi-directionnel" (toute la séquence est utilisée, les opérations sur chaque token utilisent les tokens d'avant mais aussi ceux d'après)

Contenu

- Un peu de NLP (*Tokenization*, *Embedding*)
- **Language Modeling** objective
- Encoder-only (e.g. *BERT*)
- Decoder-only (e.g. *GPT*)
- Encoder-Decoder (*T5*)

A retenir

- **Masked Language Modeling**
- **Causal Language Modeling**
- Transfer Learning (Encoder-Only)

Ressources

Lectures:

- <https://huggingface.co/learn/nlp-course/>
- <https://arxiv.org/abs/1910.10683> (T5)

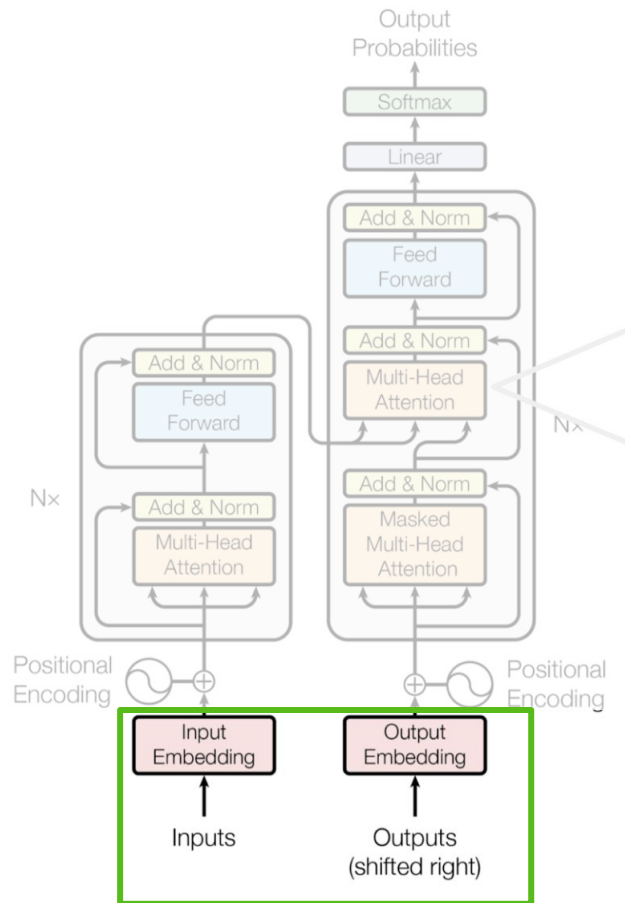
Un peu de Natural Language Processing

Tokenization

Objectif global: utiliser du texte comme entrée

Etape 1: Passer d'une séquence de mots à une séquence de symboles connus

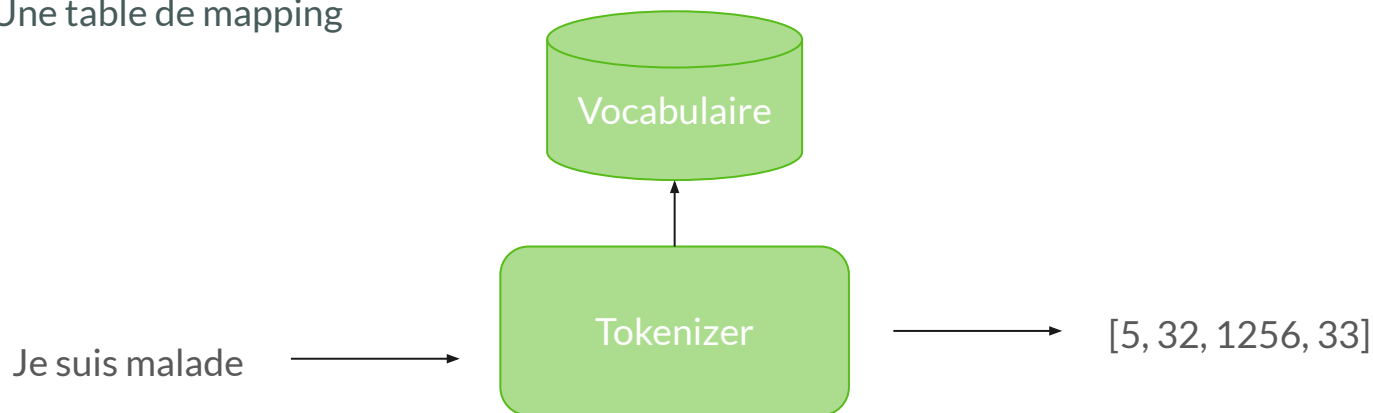
↑ ↑ ↑
Je suis malade



Tokenization

Tokenizer:

- Un “vocabulaire” de symboles
- Une table de mapping



Tokenization

Word-level mapping:

Je suis malade \longrightarrow [“Je”: 5, “suis”: 32, “malade”: 1256]

=> **Vocabulaire relativement petit, aucun partage de racine**

Tokenization

Word-level mapping:

Je suis malade → [“Je”: 5, “suis”: 32, “malade”: 1256]

=> Vocabulaire relativement petit, aucun partage de racine

Character-level mapping:

Je suis malade → [“J”: 10, “e”: 5, “s”: 18, ...]

=> Vocabulaire très petit mais peu informatif

Tokenization

Word-level mapping :

Je suis malade → ["Je": 5, "suis": 32, "malade": 1256]

=> Vocabulaire relativement petit, aucun partage de racine

Character-level mapping :

Je suis malade → ["J": 10, "e": 5, "s": 18, ...]

=> Vocabulaire très petit mais peu informatif

Tokenizers aujourd'hui utilisés :

- WordPiece (*Schuster et al., 2012*)
 - BERT
- Byte Pair Encoding (*Sennrich et al., 2018*)
 - GPT
- SentencePiece (*Kudo et al., 2018*)
 - T5, Llama

Note: GPT-2: 50k tokens

Tokenization

Tokens spéciaux:

Je suis malade → [{"</s>": 34, "Je": 5, "suis": 32, "malade": 1256, "<s>"}]

- <pad> => pad (+ mask) pour avoir des batchs de même taille
- </s> => début de séquence
- <s> => fin de séquence
- <unk> => symbole inconnu (hors de la table)
- ...

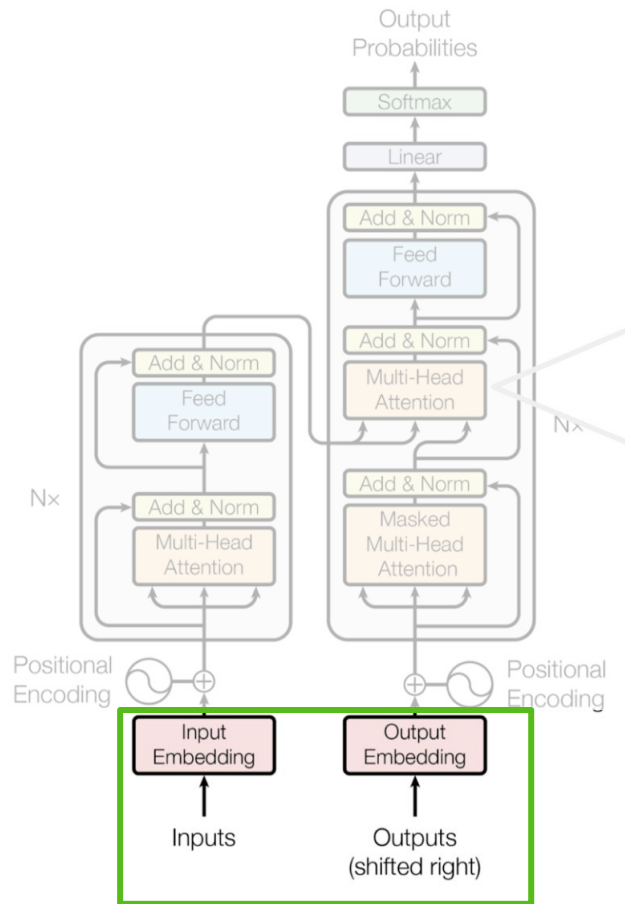
=> Dépend du tokenizer

Word Embeddings

Objectif global: utiliser du texte comme entrée

Etape 2: Passer d'une séquence de tokens à une séquence de vecteurs

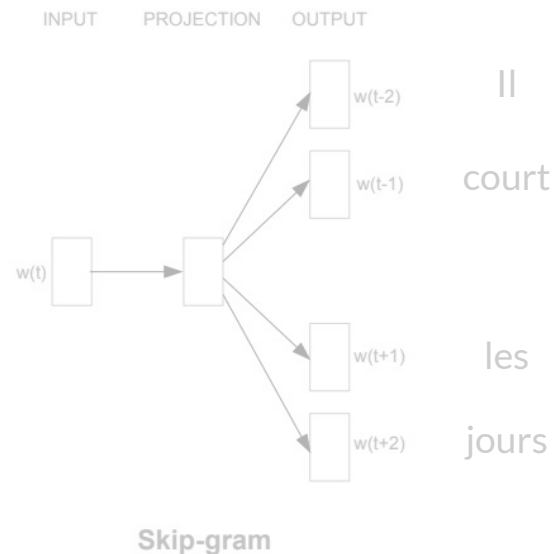
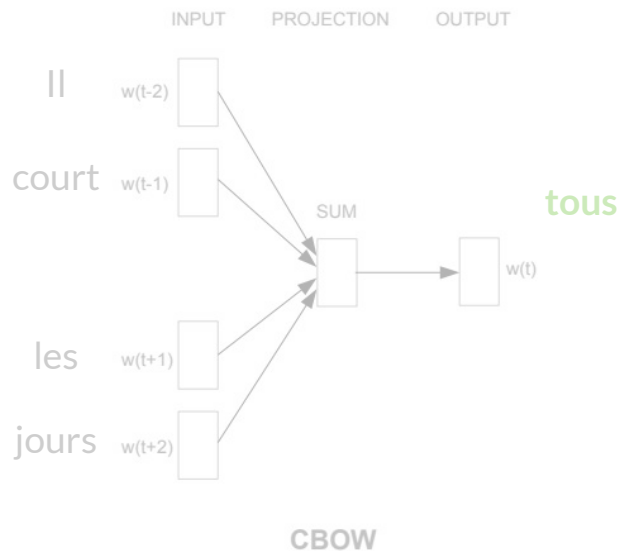
↑ Je ↑ suis ↑ malade



Word2Vec *(Mikolov et al., 2013)*

Intuition: Apprendre à représenter un mot à partir de son contexte

- 1) On utilise un vecteur **OneHotEncoding** pour chaque mot
- 2) Deux approches:



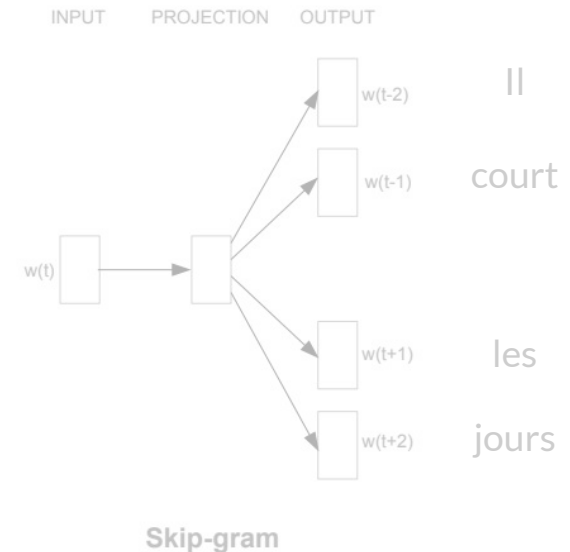
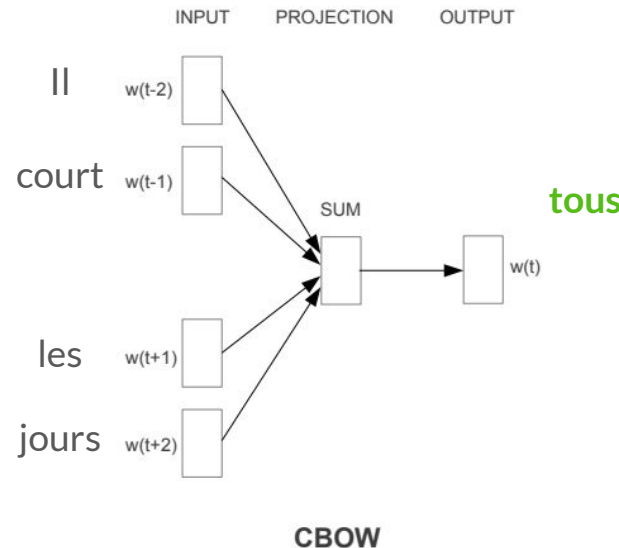
Word2Vec *(Mikolov et al., 2013)*

Intuition: Apprendre à représenter un mot à partir de son contexte

1) On utilise un vecteur **OneHotEncoding** pour chaque mot

2) Deux approches:

CBOW: On passe chaque mot du contexte dans une **couche linéaire partagée**, on fait la **moyenne de tous les vecteurs** et on **prédit le mot attendu**



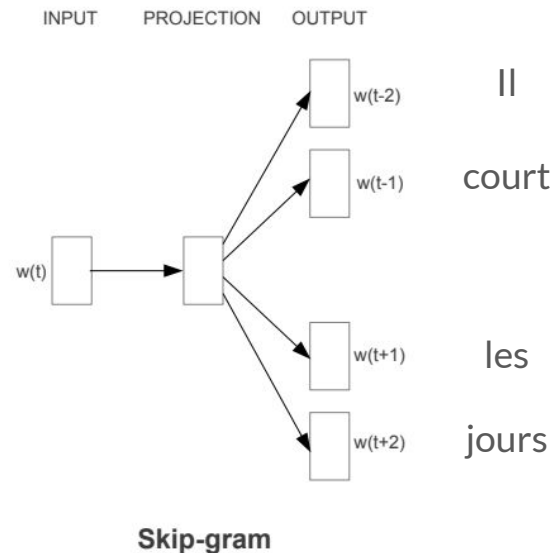
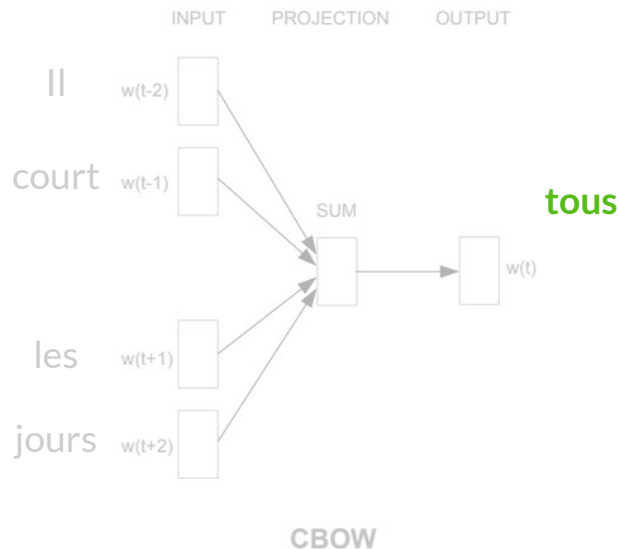
Word2Vec *(Mikolov et al., 2013)*

Intuition: Apprendre à représenter un mot à partir de son contexte

1) On utilise un vecteur **OneHotEncoding** pour chaque mot

2) Deux approches:

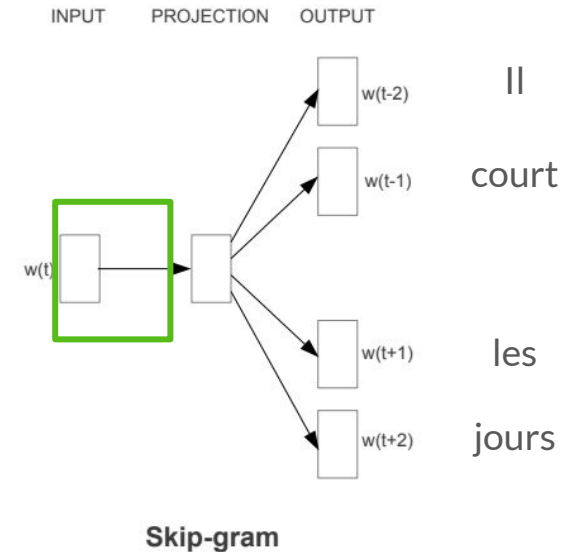
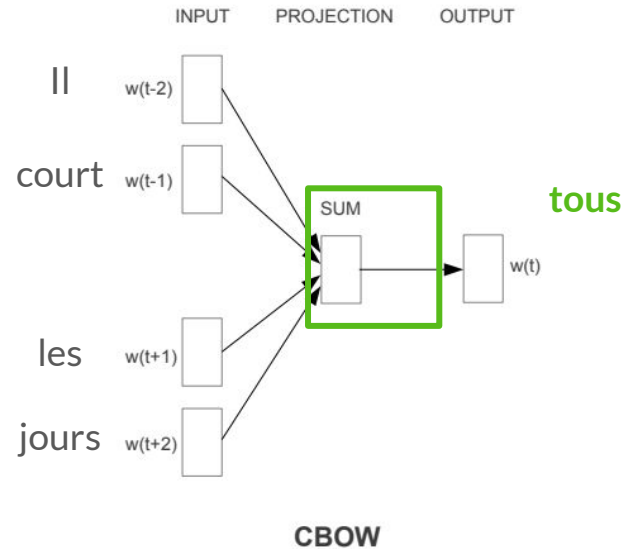
Skip-gram: On passe le mot principal dans une couche linéaire, on essaie de prédire chacun des mots du contexte



Word2Vec *(Mikolov et al., 2013)*

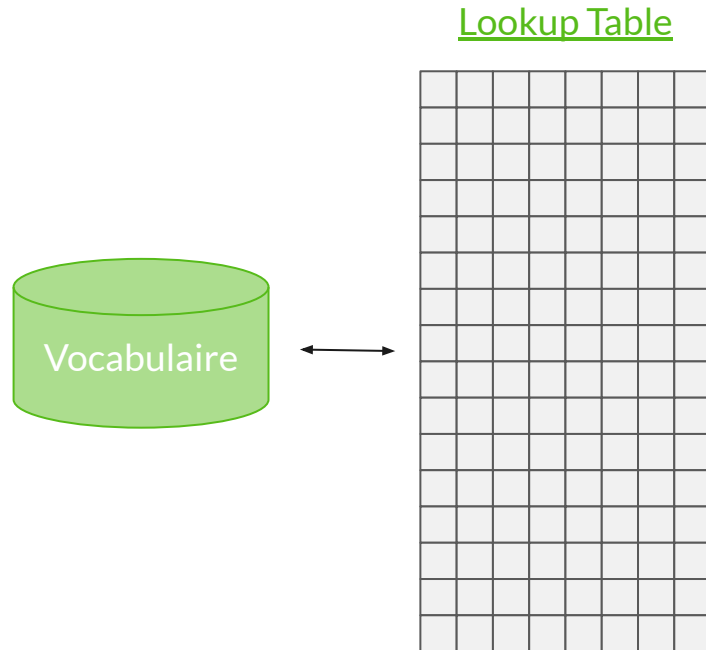
Intuition: Apprendre à représenter un mot à partir de son contexte

- 1) On utilise un vecteur **OneHotEncoding** pour chaque mot
- 2) Deux approches
- 3) On retient le vecteur obtenu avec le mot principal associé => **lookup table**



Embedding lookup

- On a donc une **table associant chaque token à son embedding**
- On peut **utiliser ces embedding** en entrée pour entraîner notre modèle (RNN, Transformer...)

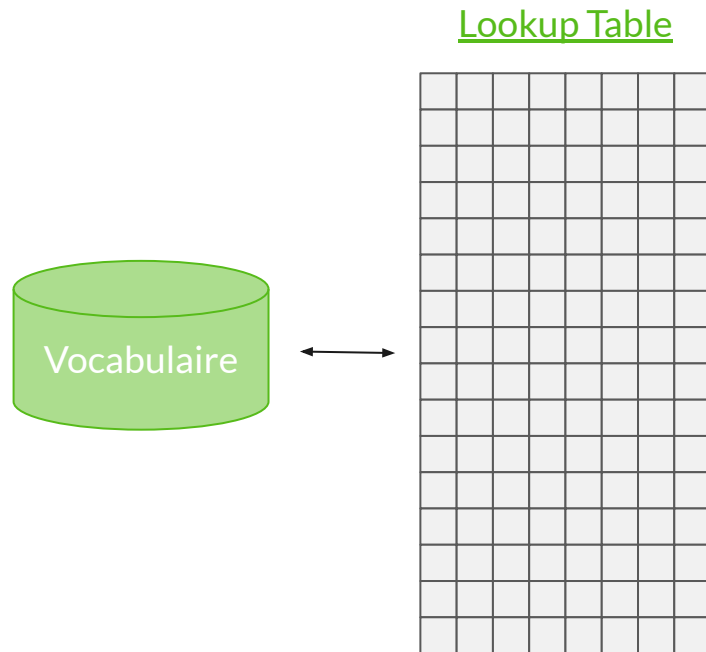


Embedding lookup

- On a donc une **table associant chaque token à son embedding**
- On peut **utiliser ces embedding** en entrée pour entraîner notre modèle (RNN, Transformer...)

On utilise en réalité une solution différente avec les transformers:

On initialise la table aléatoirement et les **embeddings** sont appris en même temps que le transformer



Language Modeling Objective

(Causal) Language Modeling

Given a corpus of tokens: $U = \{u_1, \dots, u_N\}$

$$\max_{\theta} L(U) = \sum_i \log P_{\theta}(u_i | u_{i-k}, \dots, u_{i-1})$$

model

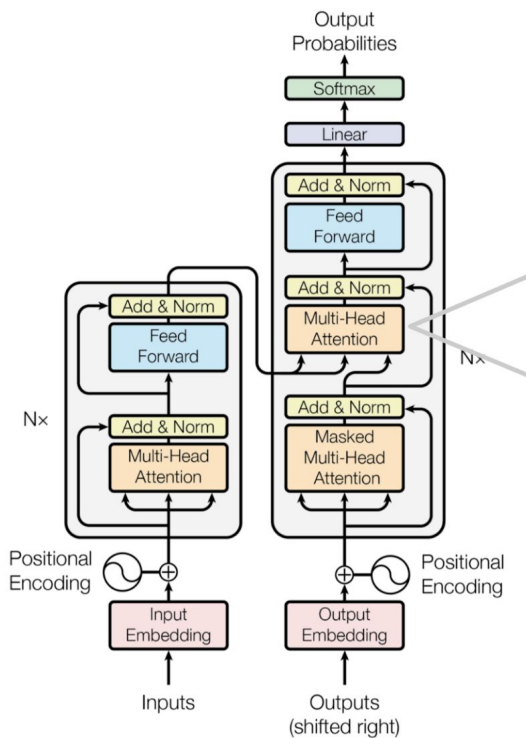
context window

(Causal) Language Modeling

En pratique:

- Etant donné un **corpus de texte tokenisé**
- On casse le corpus en **blocs de taille k**
- On souhaite apprendre un **modèle** qui **maximise la probabilité de chaque token** d'apparaître dans sa séquence

Original Transformer



Trained on WMT EN-GER or EN-FR

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

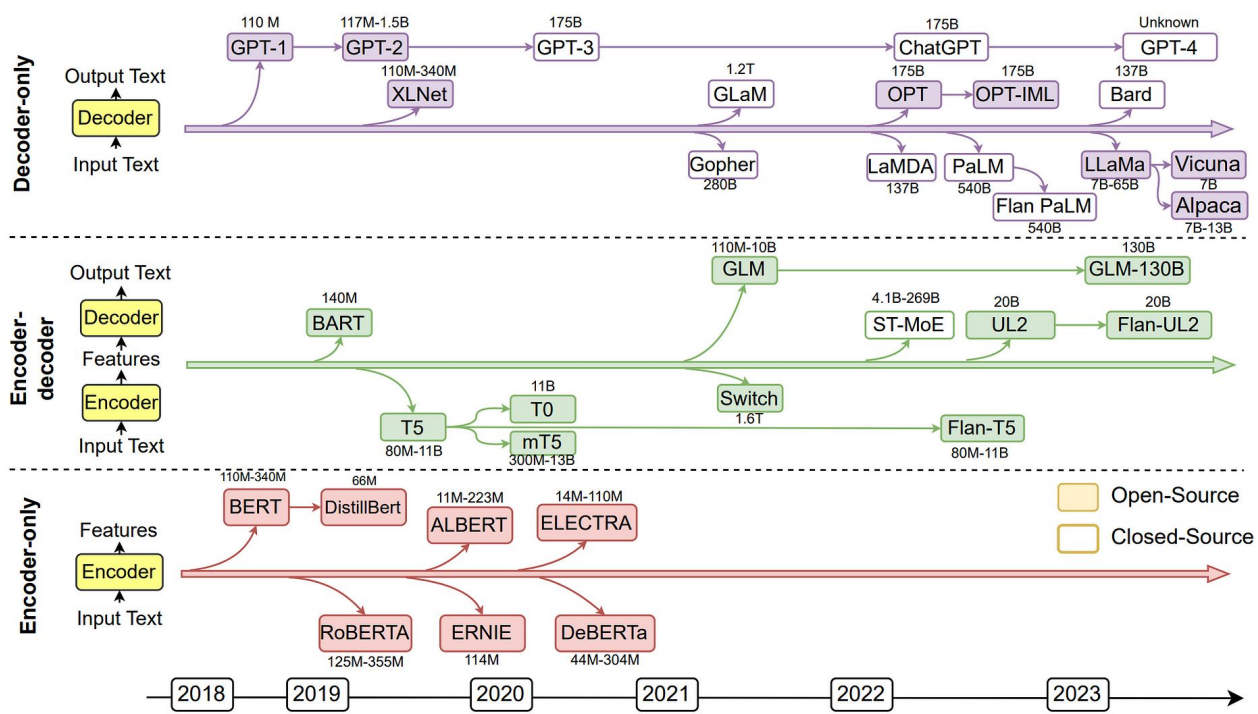
Large Language Models (LLMs)

Scaling up & Transfer Learning


A partir de 2018/2019:

- De nombreux travaux se mettent à entraîner des Transformers sur des **gros corpus généraux**
- Le modèle obtenu est ensuite utilisé comme base **pré-entraînée**
- Deux courants principaux apparaissent: **Encoder-Only vs Decoder Only**


Panel des LLMs




Outils Open-Source *(non exhaustif)*

 **Hugging Face**

Models	Demos
Datasets	Discussions

 **huggingface**

transformers	datasets
diffusers	accelerate
peft	TGI

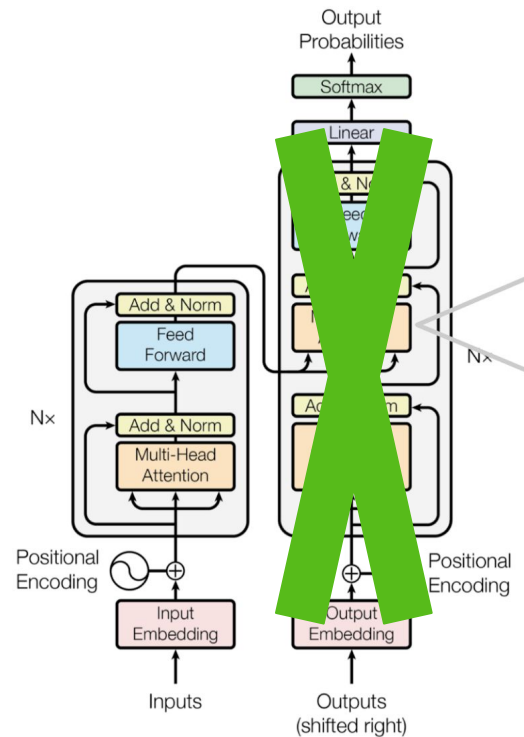
Community  **Society**

Research	Education	ML & society
----------	-----------	--------------



Encoder-Only LLMs

Encoder-Only as Foundation Models

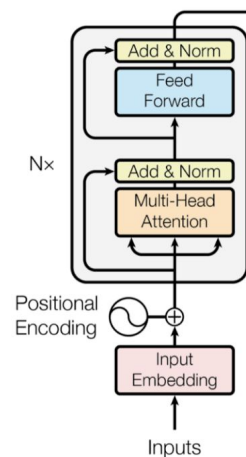


Encoder-Only as Foundation Models

Objectif

- Encoder un texte et obtenir une représentation exploitable pour différentes tâches
- Apprendre ensuite une “tête” spécifique pour un problème

=> Transfer Learning



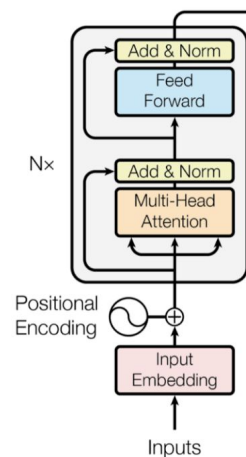
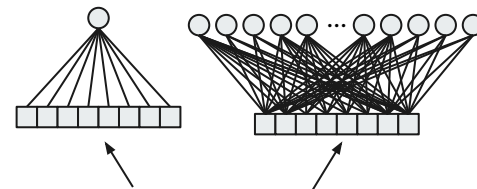
Encoder-Only as Foundation Models

Objectif

- Encoder un texte et obtenir une représentation exploitable pour différentes tâches
- Apprendre ensuite une “tête” spécifique pour un problème

=> Transfer Learning

1. (Pré-) Entraîner le LM
2. Entraîner des têtes
 - classification (topics, sentiment analysis...)



BERT (Devlin et al., 2018)

Masked Language Modeling objective

MLM:

- On remplace aléatoirement certains tokens par un token de mask
- Le modèle doit reconstruire la séquence
- Self-Supervised Learning (SSL)

Original text

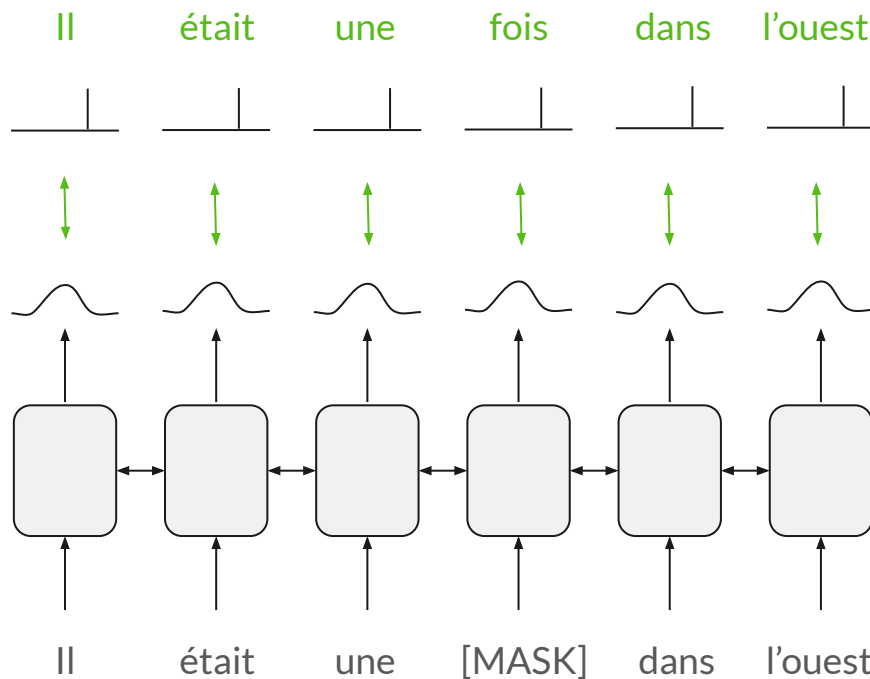
Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

Image for T5 (Raffel et al., 2019)

Masked Language Modeling objective



Entropie croisée

TP

Exemple: Multiple Q&A finetuning of DistilBert

<https://colab.research.google.com/drive/100unPJG8uQ8Dj9CScTWpzzmMgigtzmBO?usp=sharing>

https://huggingface.co/docs/transformers/tasks/multiple_choice

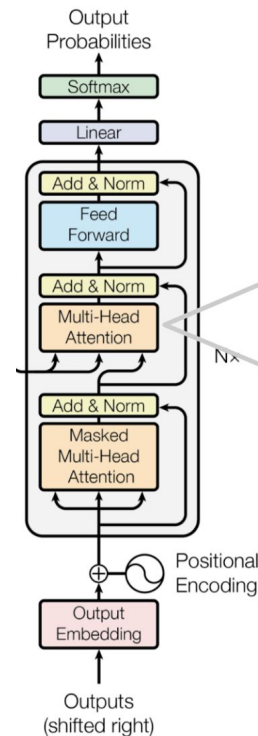
Decoder-Only LLMs

Decoder-Only Models

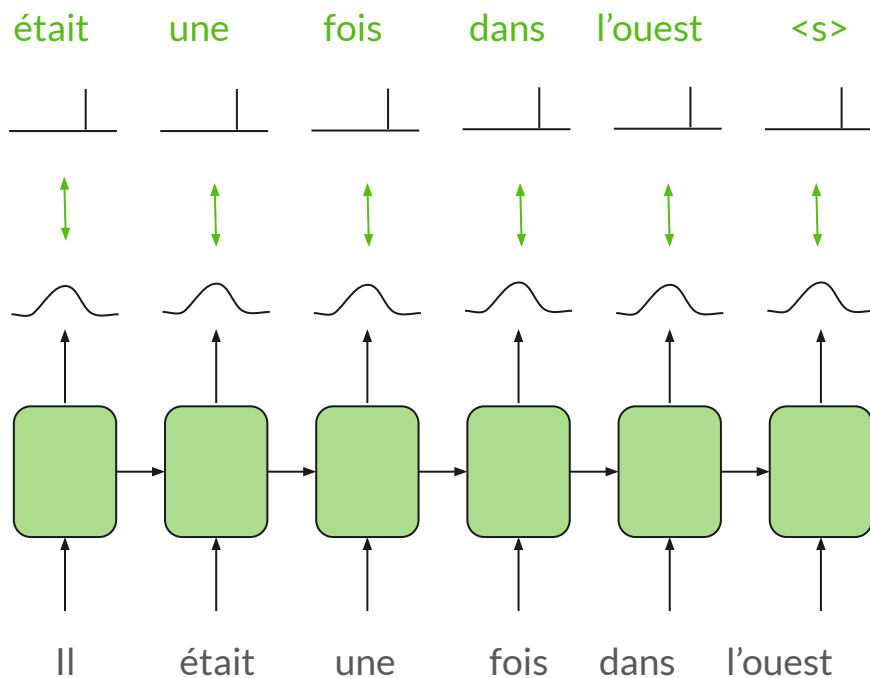
Objectif

- N'utiliser **QUE** le Language Modeling objective
- **Génération** de la suite de chaque bloc de texte

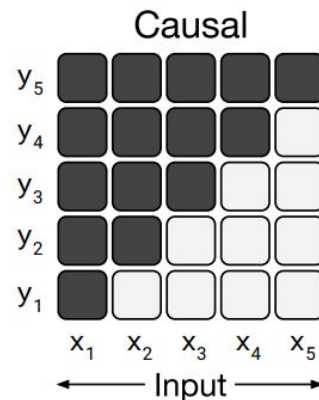
=> vers le Few-shot "learning"



Causal Language Modeling objective



Entropie croisée



Generative Pre-Training (GPT) *(Radford et al., 2018)*

- GPT-1 partage le principe de **Foundation Model** (Pre-entraînement + Finetuning)
- Introduit un **formatage de l'entrée** pour qui est spécifique à chaque tâche (e.g. ajouter la réponse après la question)

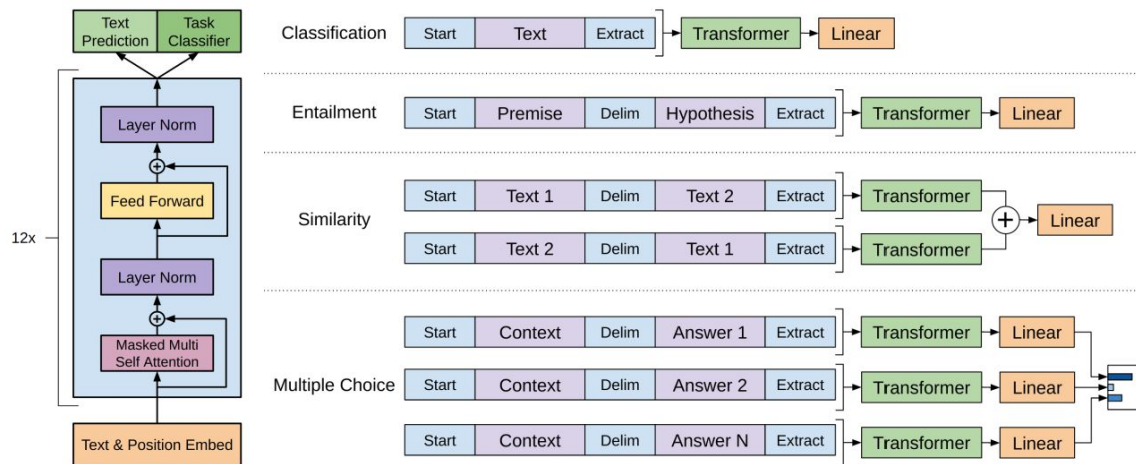


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT-2 *(Radford et al., 2019)*

- Tout est fait sous forme de texte (e.g. on génère la réponse)
- Plus de finetuning !
- Introduction du few-shot “learning” ou in-context “learning” => on donne des exemples de la tâche au modèle

Context (passage and previous question/answer pairs)

Tom goes everywhere with Catherine Green, a 54-year-old secretary. He moves around her office at work and goes shopping with her. "Most people don't seem to mind Tom," says Catherine, who thinks he is wonderful. "He's my fourth child," she says. She may think of him and treat him that way as her son. He moves around buying his food, paying his health bills and his taxes, but in fact Tom is a dog.

Catherine and Tom live in Sweden, a country where everyone is expected to lead an orderly life according to rules laid down by the government, which also provides a high level of care for its people. This level of care costs money.

People in Sweden pay taxes on everything, so aren't surprised to find that owning a dog means more taxes. Some people are paying as much as 500 Swedish kronor in taxes a year for the right to keep their dog, which is spent by the government on dog hospitals and sometimes medical treatment for a dog that falls ill. However, most such treatment is expensive, so owners often decide to offer health and even life _ for their dog.

In Sweden dog owners must pay for any damage their dog does. A Swedish Kennel Club official explains what this means: if your dog runs out on the road and gets hit by a passing car, you, as the owner, have to pay for any damage done to the car, even if your dog has been killed in the accident.

Q: How old is Catherine?

A: 54

Q: where does she live?

A:

Model answer: Stockholm

Turker answers: Sweden, Sweden, in Sweden, Sweden

Comparatif: T5

T5 *(Raffel et al., 2019)*

Architecture	Objective	Params	Cost	GLUE	CNN4	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	Denoising	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Table 2: Performance of the different architectural variants described in Section 3.2.2. We use P to refer to the number of parameters in a 12-layer base Transformer layer stack and M to refer to the FLOPs required to process a sequence using the encoder-decoder model. We evaluate each architectural variant using a denoising objective (described in Section 3.1.4) and an autoregressive objective (as is commonly used to train language models).

C'est terminé !

<https://279vrcxd7nq.typeform.com/to/JGI5s4rp>

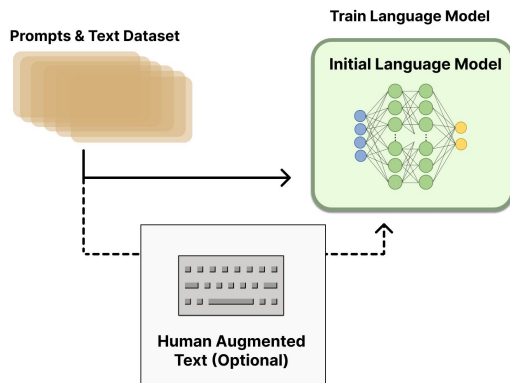


Pour aller plus loin

Reinforcement Learning from Human Feedback

1. Supervised Fine-tuning:

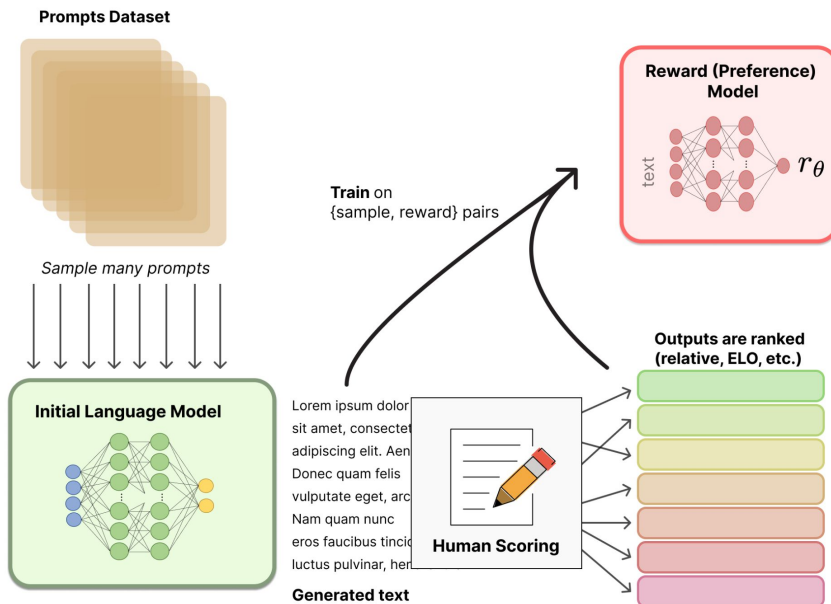
- Utilisation d'un jeu de données avec des instructions et le texte à générer associé



Reinforcement Learning from Human Feedback

2. Reward modeling:

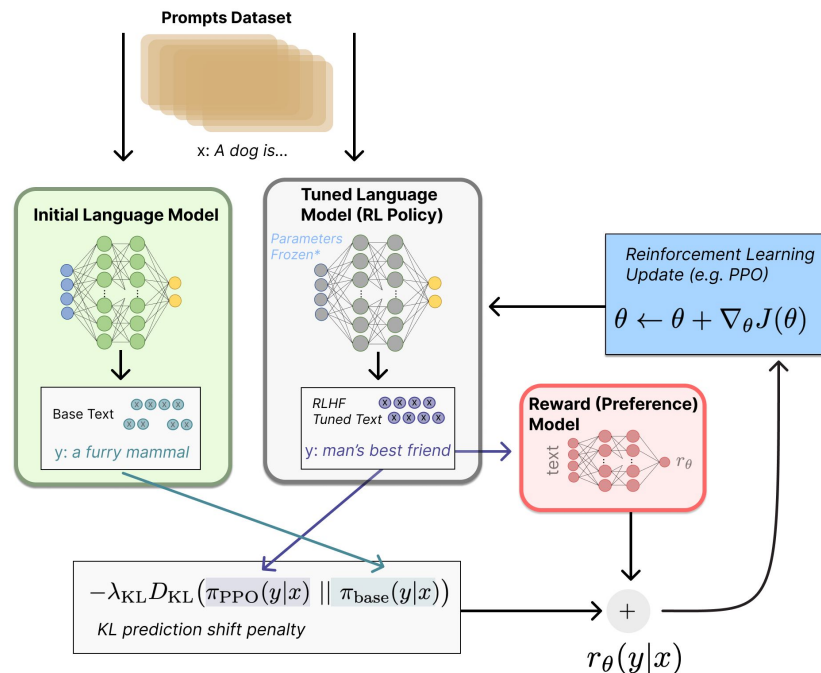
- Utilisation d'un autre jeu de données avec que des prompts
- Le modèle est utilisé pour générer plusieurs sorties par prompt
- Des annotateurs humains notent les sorties
- On apprend un modèle qui prédit un score



Reinforcement Learning from Human Feedback

2. RLHF:

- On utilise le LLM pour générer un texte à partir de chaque prompt du dataset utilisé avec les humains
- On considère chaque token comme une action
- On utilise le modèle de récompense appris et on finetune avec de l'apprentissage par renforcement



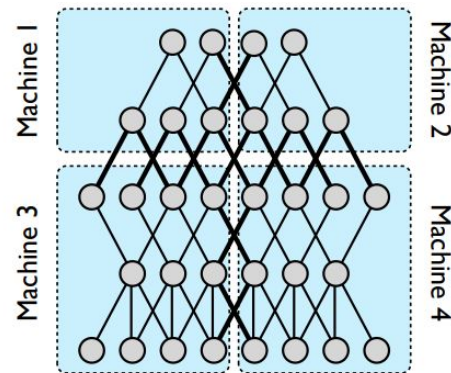
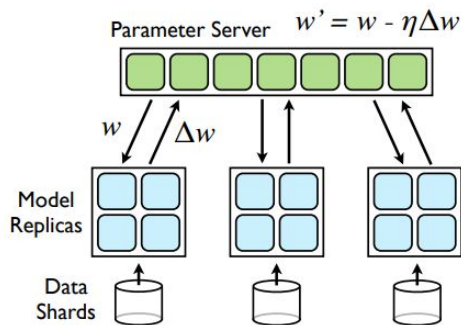
Entraînement distribué

Basics:

- Data Parallelism
- Model Parallelism

More advanced:

- Tensor Parallelism
- Pipeline Parallelism
- ZeRO redundancy
- ...



(Dean et al., 2012)

https://huggingface.co/docs/transformers/perf_train_gpu_many

https://huggingface.co/docs/transformers/perf_train_gpu_one

Focus LLMs récents

T	Model	Average	TruthfulQA	Type	Hub License	#Params (B)
●	llm-agents/tora-70b-v1.0	68.65	51.79	pretrained	llama2	68.72
●	meta-llama/llama-2-70b-hf	67.35	44.92	pretrained	?	68.98
●	TigerResearch/tigerbot-70b-base	66.08	52.76	pretrained	apache-2.0	68.95
●	internlm/internlm-20b	64.27	52.61	pretrained	apache-2.0	20
●	huggyllama/llama-65b	64.23	43.43	pretrained	other	65.29
●	llama-65b	64.23	43.43	pretrained	other	65.29
●	llama-30b	61.68	42.27	pretrained	other	32.53
●	tiiuae/falcon-40b	61.48	41.72	pretrained	apache-2.0	41.3
●	mosaicml/mpt-30b-chat	60.94	52	pretrained	cc-by-nc-sa-4.0	29.96
●	mistralai/Mistral-7B-Instruct-v0.1	60.45	56.28	pretrained	apache-2.0	7.11
●	llm-agents/tora-13b-v1.0	59.06	40.25	pretrained	llama2	12.85
●	chargodard/llama2-22b-blocktriangular	58.77	39.3	pretrained	?	21.62

(https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard)

Datasets

L'exemple de BLOOM:

- Un mélange de **crowdsourcing** et **OSCAR** (Common Crawl)
- Multilingue
- Le nettoyage joue un rôle clé (filtering, deduplication, PII removal...)

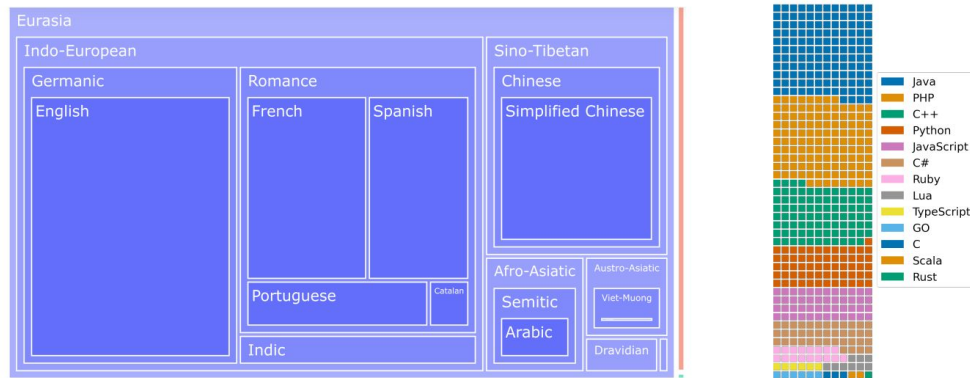


Figure 1: Overview of ROOTS. Left: A treemap of natural language representation in number of bytes by language family. The bulk of the graph is overwhelmed by the 1321.89 GB allotted to Eurasia. The orange rectangle corresponds to the 18GB of Indonesian, the sole representative of the Papunesia macroarea, and the green rectangle to the 0.4GB of the Africa linguistic macroarea. Right: A waffle plot of the distribution of programming languages by number of files. One square corresponds approximately to 30,000 files.

Datasets

L'exemple de BLOOM:

- Un mélange de **crowdsourcing et OSCAR** (Common Crawl)
- Multilingue
- Le nettoyage joue un rôle clé (filtering, deduplication, PII removal...)

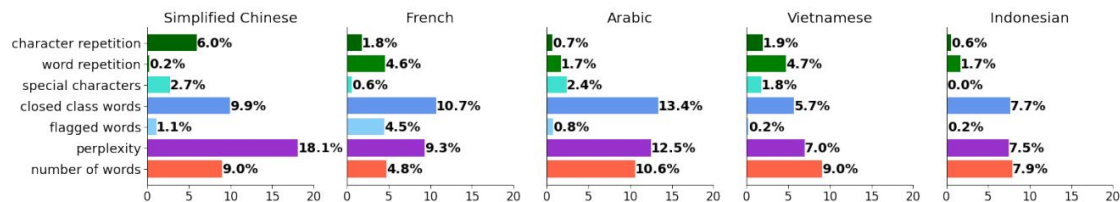


Figure 3: Percentage of documents discarded by each filter independently for 5 languages

Datasets

L'exemple de BLOOM:

- Un mélange de **crowdsourcing** et **OSCAR** (Common Crawl)
- Multilingue
- Le nettoyage joue un rôle clé (filtering, deduplication, PII removal...)

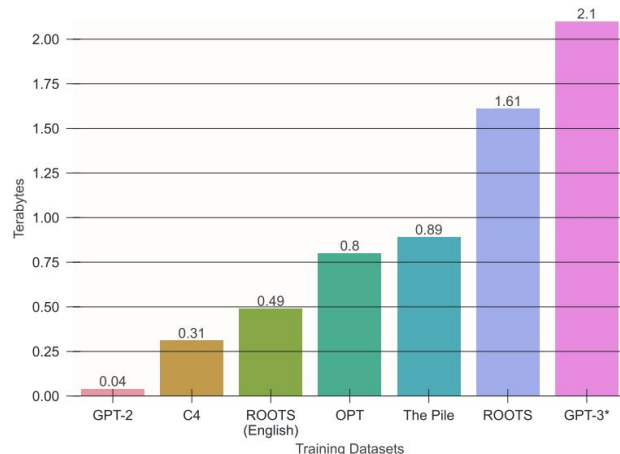


Figure 4: A raw size comparison to other corpora used to train large language models. The asterisk next to GPT-3 indicates the value in question is an estimate computed using the reported number of tokens and the average number of tokens per byte of text that the GPT-2 tokenizer produces on the Pile-CC, Books3, DWT2, and Wiki-en subsets of the Pile (Gao et al., 2020)

Evaluation

Performance metric	Number of benchmark datasets	Percent
BLEU score	300	61.1
ROUGE metric	114	23.2
Perplexity	48	9.8
METEOR	39	7.9
Word error rate	36	7.3
Exact match	33	6.7
CIDEr	24	4.9
Unlabeled attachment score	18	3.7
Labeled attachment score	15	3.1
Bit per character	12	2.4

Table 2: Top 10 reported NLP metrics and percent of NLP benchmark datasets (n=491) that use the respective metric. BLEU: Bilingual Evaluation Understudy, CIDEr: Consensus-based Image Description Evaluation, ROUGE: Recall-Oriented Understudy for Gisting Evaluation, METEOR: Metric for Evaluation of Translation with Explicit Ordering.

Original application

Machine translation

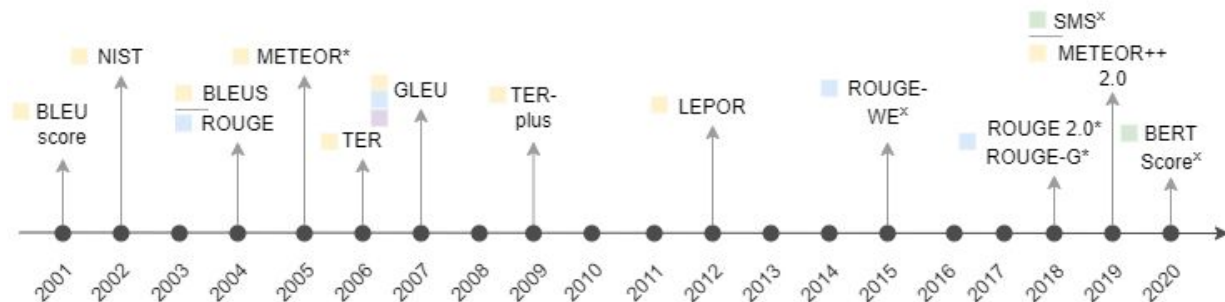
Summarization

Natural language generation

Task agnostic

* Uses WordNet synonyms and/or paraphrases

x Uses embeddings



(Blagec et al., 2022)