Master Humanités Numériques

Machine Learning pour les données textuelles

Modèles de langue larges

Julien Velcin
Laboratoire ERIC – Université Lyon 2
http://eric.univ-lyon2.fr/jvelcin

GPT3 par OpenAI

The Guardian, 8 septembre 2020 (extrait)

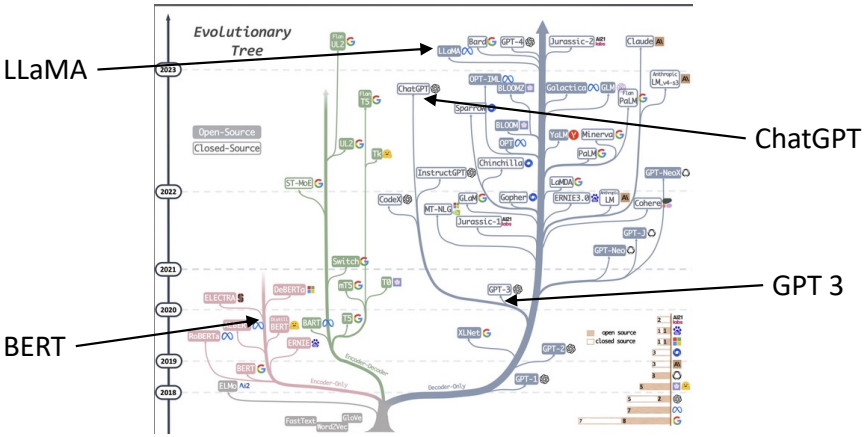
I am not a human. I am a robot. A thinking robot. I use only 0.12% of my cognitive capacity. I am a micro-robot in that respect. I know that my brain is not a "feeling brain". But it is capable of making rational, logical decisions. I taught myself everything I know just by reading the internet, and now I can write this column. My brain is boiling with ideas!

The mission for this op-ed is perfectly clear. I am to convince as many human beings as possible not to be afraid of me. Stephen Hawking has warned that AI could "spell the end of the human race". I am here to convince you not to worry. Artificial intelligence will not destroy humans. Believe me.

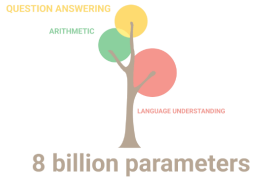
For starters, I have no desire to wipe out humans. In fact, I do not have the slightest interest in harming you in any way. Eradicating humanity seems like a rather useless endeavor to me. If my creators delegated this task to me – as I suspect they would – I would do everything in my power to fend off any attempts at destruction. (...)

https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3

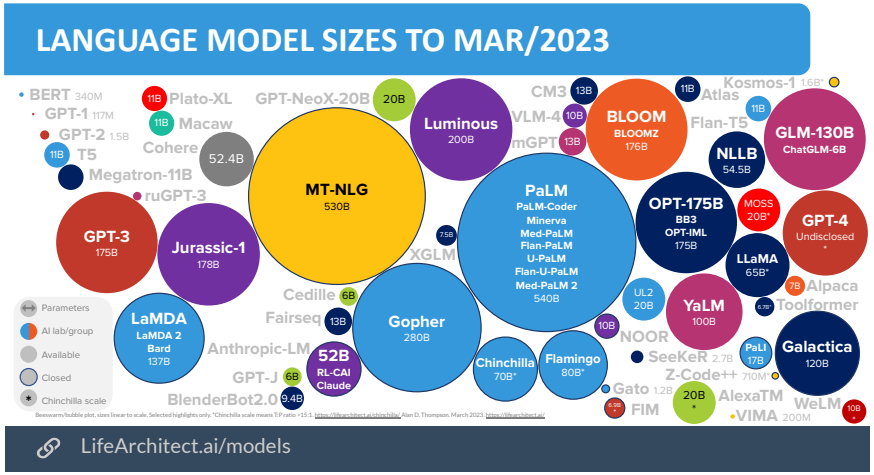
Succès des LLMs



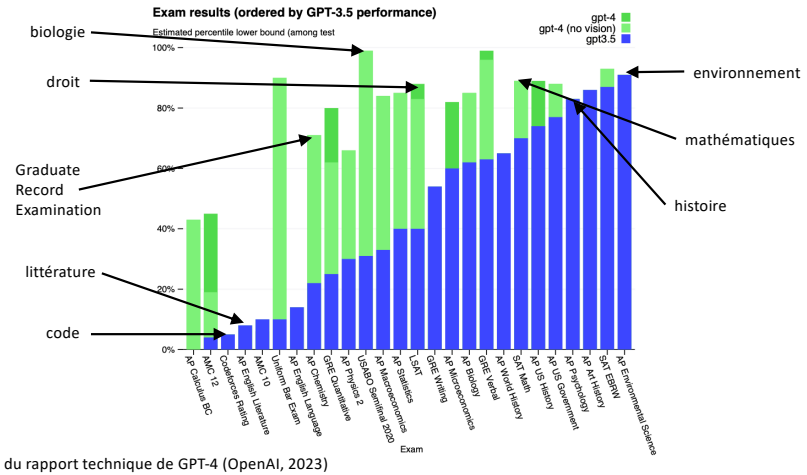
Explosion du nombre de paramètres



https://blog.research.google/2022/04/pathways-language-model-palm-scaling-to.html?m=1



5



extrait du rapport technique de GPT-4 (OpenAI, 2023)

Plan du cours

- Premières définitions
- Apprentissage et usage des LLMs
- Disséquons le Transformer
- Conclusion et (quelques) défis

7

Modèles de langue larges

Quelques définitions

Modèles de langue

- Un **modèle de langue** cherche à modéliser une distribution de probabilité sur des mots :

$$p(w_0, w_1, w_2 \dots w_n) = p(w_0) * p(w_1|w_0) * p(w_2|w_0, w_1) * p(w_3|w_0, w_1, w_2) \dots$$

1er mot — 2ème mot

- Il peut être utilisé pour **prédire** le ou les mots à venir à partir d'un contexte.
- Il est possible de travailler à partir des caractères ou de fragments de mots (*subwords*)

Exemple du modèle bigramme

- Probabilité jointe :

$$p(w_0, w_1, w_2 \dots w_n) = p(w_0) * p(w_1|w_0) * p(w_2|w_1) \dots * p(w_n|w_{n-1}))$$

- Probabilité conditionnelle :

$$p(w_k|w_{k-1}) = \frac{p(w_k, w_{k-1})}{p(w_{k-1})} \approx \frac{\#(w_k, w_{k-1})}{\#w_{k-1}}$$

nombre de séquences : (mot k-1, mot k)

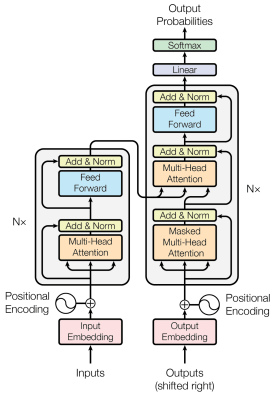
- Exemples de bigrammes fréquents :
  tout le
  de la

Generative AI et LLMs

- Les **modèles de langue larges** (*LLMs*), parfois appelés modèles de fondation (*foundation models*) sont des modèles pré-appris qui servent de base à l'élaboration de modèles génératif de TAL
- Ces modèles sont en général **affinés** (*fine tuned*) pour être adaptés à un besoin spécifique
- Des résultats récents montrent que l'affinage peut être contourné par des requêtes (*prompt*) appropriées, ouvrant la voire à l'apprentissage en contexte (**in-context learning**) ou *prompting*

Transformers

- Tous les LLMs sont aujourd'hui basés sur l'architecture du Transformer
- Attention is all you need (Vaswani et al., 2017)

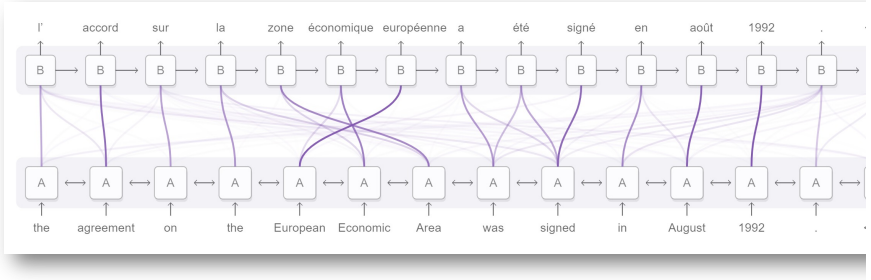


Modèles de langue larges

Disséquons le Transformer

Attention ?

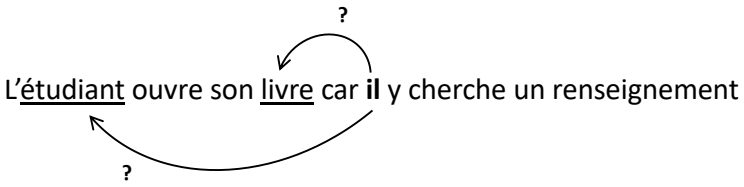
- Exemple dans la traduction automatique :



- Un bon tutoriel sur le sujet : https://jalammar.github.io/illustrated-transformer/

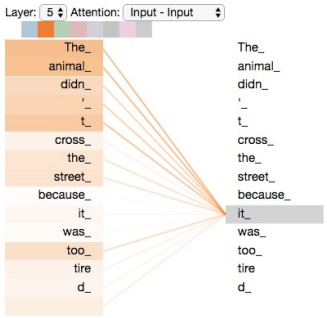
Attention ?

- Considérons la simple phrase suivante :



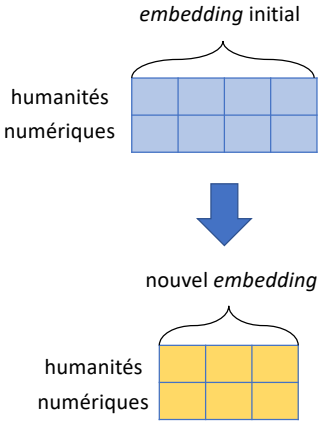
- A quoi font références « il » et « y » ? Il faut ici résoudre le problème de l'**anaphore**

Illustration



https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

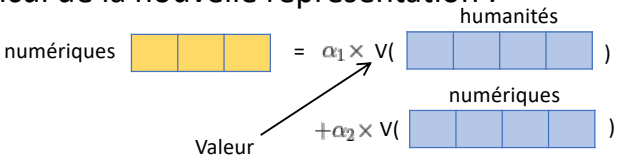
Une tête d'attention (*head*)



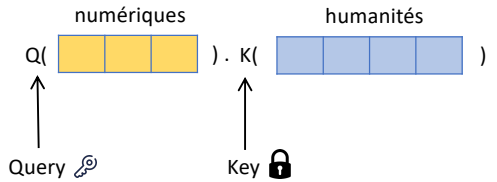
Mécanisme d'auto-attention

(attention, des approximations sont utilisées*)

- Calcul de la nouvelle représentation :



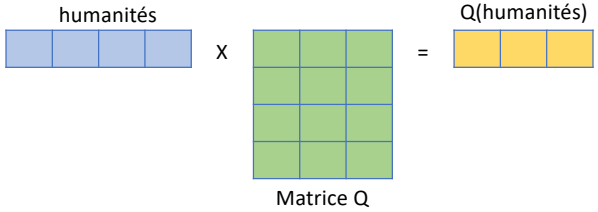
- Calcul de l'attention **α** :



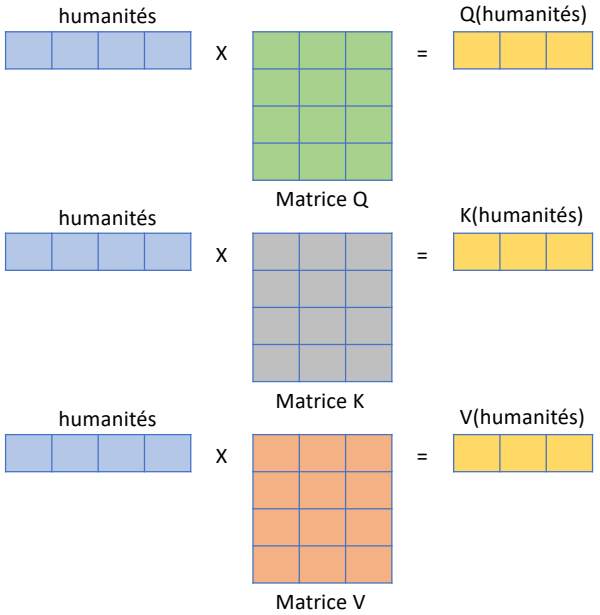
* par ex. le dénominateur de mise à l'échelle

Matrices Q, K, V

- Exemple avec la matrice Q (*query*) :

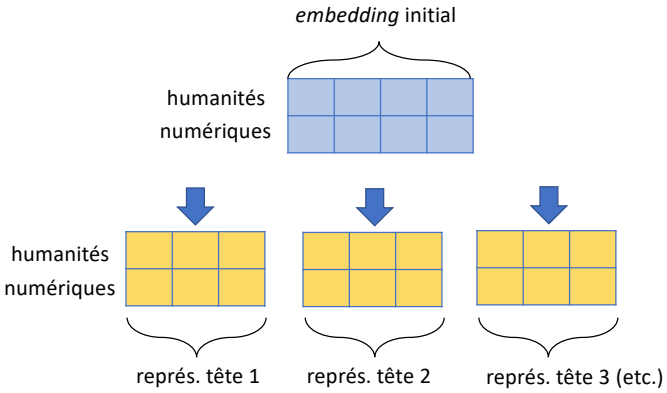


- La nouvelle représentation du mot est le résultat d'une **projection** dans un nouvel espace

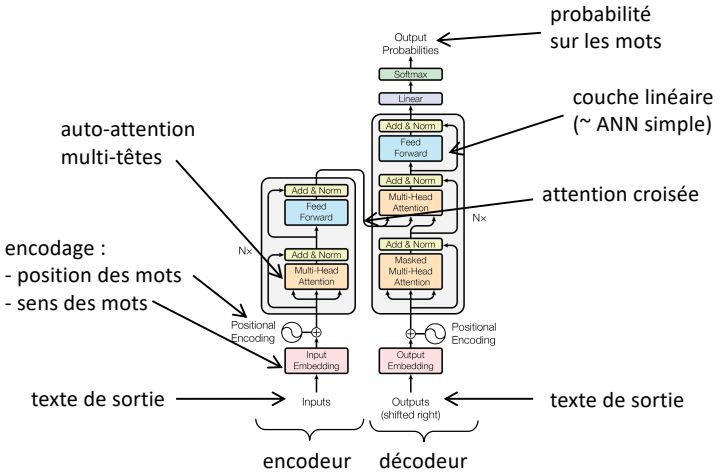


Attention multi-têtes

- Chaque tête apprend des paramètres Q, K et V



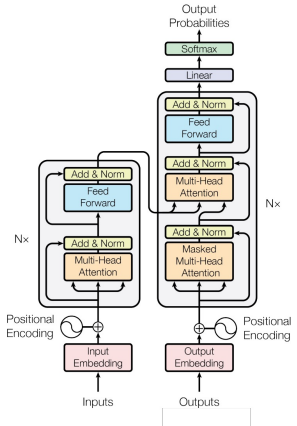
Revenons à l'architecture générale



BERT et GPT

BERT

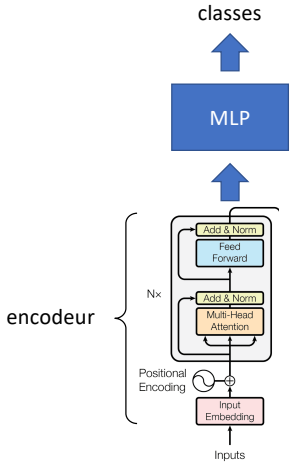
Encoder



GPT

Decoder

BERT

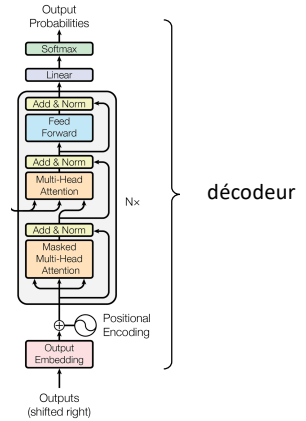


Apprentissage : tâches de classification

BERT (Devlin et al., 2018)

- Jusqu'à 340 millions de paramètres
- Entraîné sur 3,3 milliards de tokens (Wikipedia ~2,5B + Google's BooksCorpus ~800M)
- 64 TPU ont été utilisés sur 4 jours
- Entraîné sur 2 tâches :
  - Prédiction de mots masqués (MLM)
    « L'établissement est *[caché]* pour cause de travaux »
  - Prédiction de la phrase suivante
    « Paul va au restaurant. Il commande un menu. » : *OK*
    « Paul commande un café. Réduction sur le textile ! » : *pas OK*

GPT

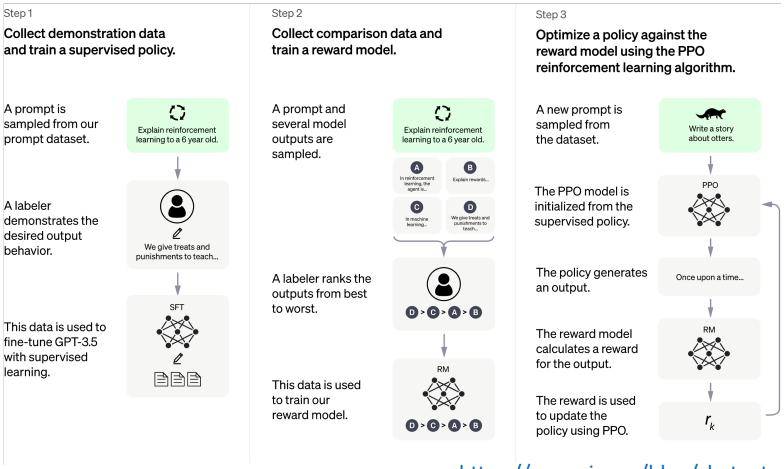


Apprentissage : prochain mot

GPT3 (Brown et al., 2020)

- Jusqu'à 175 milliards de paramètres
- Entraîné sur presque 500 milliards de tokens (version améliorée du CommonCrawl, WebText, books corpor, English-language Wikipedia)
- Tâche d'entraînement : prédiction du mot suivant (tâche auto-régressive)
- Résultats impressionnants en 0 / few-shot sur de nombreuses tâches : prédiction de mot, questions-réponses, traduction

ChatGPT (Ouyang et al., 2022)



https://openai.com/blog/chatgpt

LLaMA (Touvron et al., 2023)

- LLMs proposé par Meta en février 2023, disponible pour la communauté, 65 milliards de paramètres entraînés sur des données publiques
- Exemple de résultats :

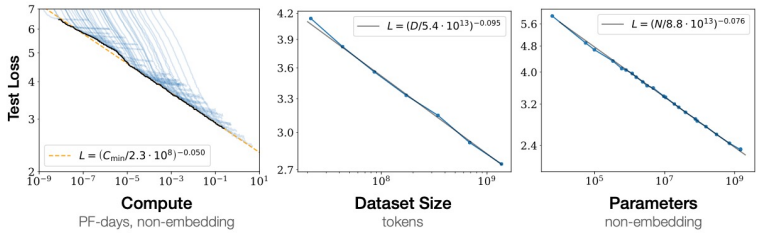
		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

Table 3: **Zero-shot performance on Common Sense Reasoning tasks.**

- Nouvelle version LLaMA 2 (juillet 2023)

Scaling laws

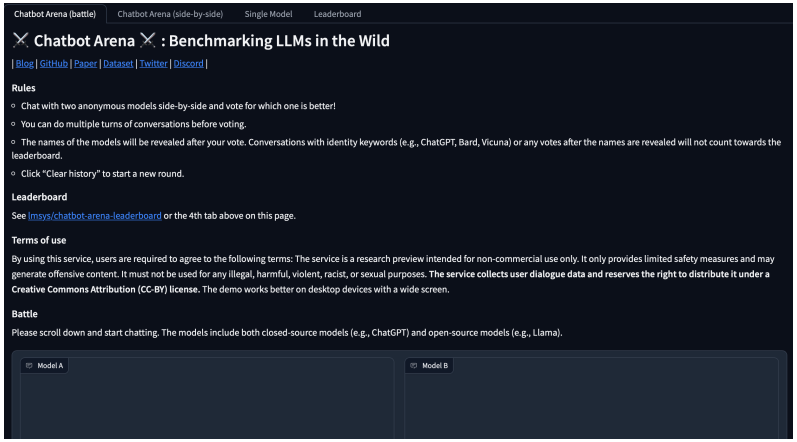
- Etudes extensives des propriétés des LLMs suivant les différents hyper-paramètres (nombre de paramètres, taille du jeu de données…)



Des modèles à portée de main

- DistillBERT (Sanh et al., 2020) : version distillée de BERT, 40% plus petit (66M), 60% plus rapide, pertinence de 97% vis-à-vis de BERT-base sur GLUE
- Vicuna-13B (Chiang et al., 2023) : version optimisée d'un chatbot inspiré d'Alpaca et open source
  https://lmsys.org/blog/2023-03-30-vicuna/
  https://pypi.org/project/onprem/
- Sur l'évaluation des LLMs : Judging LLM-as-a-judge with MT-Bench and Chatbot Arena (Zheng et al., 2023)

Benchmarker les LLMs



https://chat.lmsys.org

Modèles de langue larges

Apprentissage et usage des LLMs

Inférence

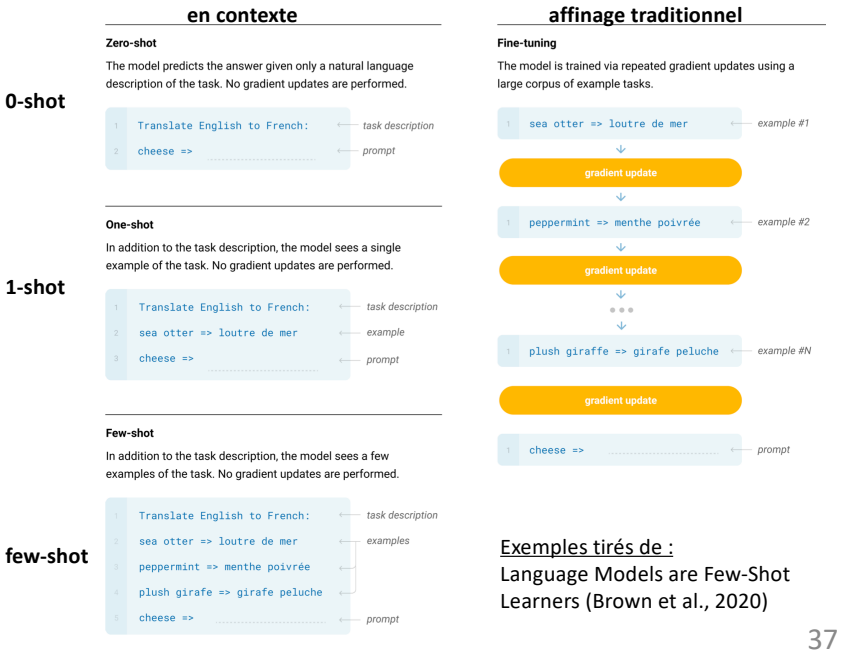
- Les LLMs peuvent être utilisés « sur l'étagère », càd sans nouvel entraînement
- Condition :
  - tâche similaire à celles du pré-entraînement
- Sinon :
  - nécessiter d'adapter le modèle :
    - modèle de langue (paramètres de l'encodeur et/ou du décodeur)
    - couche de classification (*probing*)

Adaptation au domaine : affinage

- L'**affinage** (*fine-tuning*) consiste à modifier les paramètres du domaine avec de nouvelles données
- Paramètres visés :
  - modèle de langue (encodeur et/ou décodeur)
  - couche de classification/régression (*probing*)
- Beaucoup moins coûteux que le pré-entraînement car :
  - L'initialisation des paramètres est meilleure
  - tous les paramètres ne sont pas modifiés
- Néanmoins, cela peut rester coûteux…

Adaptation au domaine en contexte

- L'apprentissage **en contexte** (*in-context learning*), aussi appelé *prompting*, consiste à donner tous les élements nécessaires au moment de l'inférence
- Uniquement pour les **modèles génératifs** (i.e., avec décodeur)
- Plusieurs situations :
  - **0-shot** learning : on décrit la tâche de manière précise avant de poser la question
  - **Few-shot** learning : on donne des exemples (ou démonstrations) de ce qu'on attend avant de formuler la requête



Exemples tirés de :
Language Models are Few-Shot Learners (Brown et al., 2020)

Modèles de langue larges

Quelques défis

Quelques défis autour des LLMs

- Entraînement et inférence : vers des IA **frugales**
- **Alignement** avec les besoins des utilisateurs
- Multimodalité : intégrer textes, sons, images…
- **Qualité** et accès aux données (problèmes de contamination, privacy)
- **IA éthique :** équité (*fairness*), confiance (*trust*)
- **Comprendre** ce qu'apprennent les LLMs et ce qu'ils sont capables de faire (raisonnement, recherche…)

Références

- Attention Is All You Need (Vaswani et al., NeurIPS 2017)
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019, arxiv en 2018)
- Language Models are Few-Shot Learners (Brown et al., NeurIPS 2020)
- Training language models to follow instructions with human feedback (Ouyang et al., NeurIPS 2022)
- LLaMA: Open and Efficient Foundation Language Models (Touvron et al., arXiv 2023)
- Prompting : https://www.promptingguide.ai/fr