

# INTRODUCTION AU MACHINE LEARNING

## MODÈLES DE LANGAGE

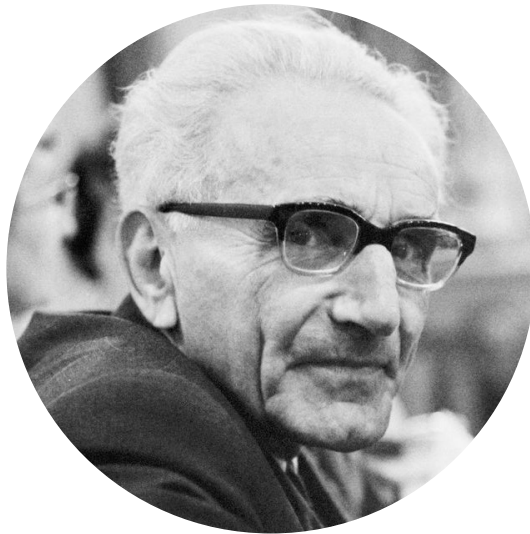
**Théo Lopès-Quintas**

BPCE Payment Services,  
Université Paris Dauphine

2022-2025

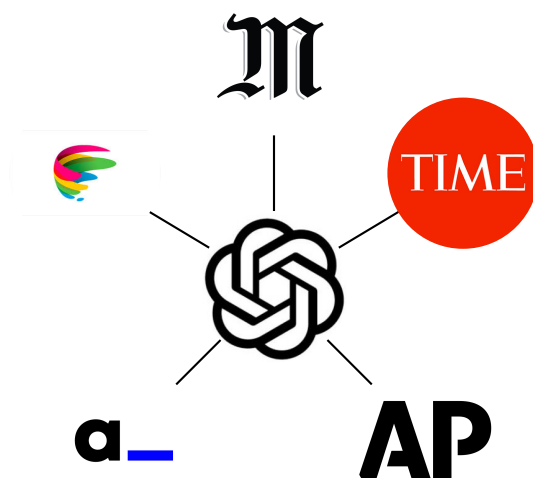
# INTRODUCTION

GÜNTHER ANDERS



# INTRODUCTION

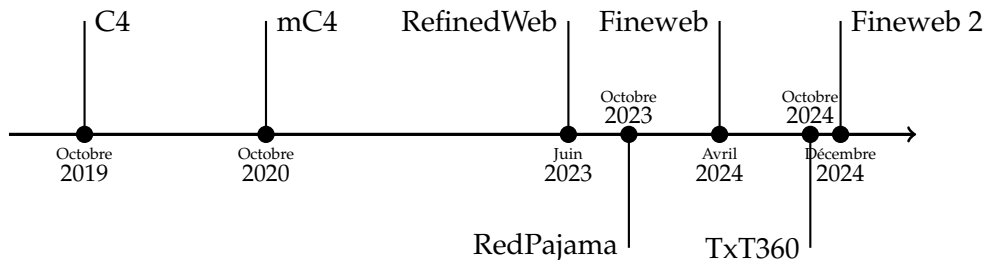
## QUELQUES PARTENARIATS D'OPENAI



# COLLECTER ET TRANSFORMER DU TEXTE

## COMMENT CONSTITUER UN CORPUS ?

Constituer un dataset d'entraînement est un challenge dans tous les problèmes de Machine Learning. Citons quelques exemples de *grands* datasets dans le domaine des modèles de langage :



**Figure** – Chronologie de quelques datasets

La source principale de l'ensemble de ces datasets est Internet, plus précisément des *images* d'internet prise mensuellement : les **dumps**.

# COLLECTER ET TRANSFORMER DU TEXTE

## COMMENT CONSTITUER UN CORPUS ?

Pour nettoyer les *dumps*, de nombreux filtres sont utilisés :

- ▶ **URL** : suppression de page selon une liste d'URL ( 4.6M de sites) et selon la présence de certains mots dans les URL
- ▶ **Langue** : ne sont conservé que les sites dont la langue identifié est l'anglais<sup>1</sup>.
- ▶ **Qualité et répétition** : ne sont conservé que les sites qui vérifie les conditions proposées pour construire MassiveText qui a entraîné Gopher [Rae et al., 2021] :
  - La page contient entre 50 et 100 000 mots, et la longueur moyenne des mots est comprise entre 3 et 10
  - La page a un ratio entre symboles et mots inférieur à 0.1 pour les symboles dièse ou points de suspension
  - La page a moins de 90% de ses lignes qui commence par des puces et qui ont moins de 30% des lignes qui termine par des points de suspension
  - 80% des mots doivent contenir au moins une lettre
  - Au moins deux mots parmi la liste : *the, be, to, of, and, that ; have, with* doivent être présent dans le document
  - La page ne dépasse aucun des seuils de répétitions décrit dans l'article [Rae et al., 2021] section A.1.1

---

1. Le modèle fastText [Joulin et al., 2016] est utilisé avec un score de 0.65

# COLLECTER ET TRANSFORMER DU TEXTE

## COMMENT CONSTITUER UN CORPUS ?

[Lee et al., 2021] identifie quatre avantages d'entraîner un grand modèle de langage sur un dataset dédoublonné :

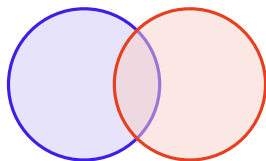
1. **Réduction du risque de mémorisation** de certaines séquences présentent trop souvent (démontré dans [Carlini et al., 2022])
2. **Réduction de l'overlap** entre train et test. L'article exhibe une séquence de 61 mots qui est répété 61 036 dans C4
3. **Gain en rapidité d'entraînement**, donc évitement de coût d'entraînement
4. **Gain en performance** jusqu'à 10% grâce à du texte de meilleure qualité

## COLLECTER ET TRANSFORMER DU TEXTE

### COMMENT DÉDUPLIQUER ?

Considérons deux ensembles  $A$  et  $B$ , on définit l'indice de Jaccard comme :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (\text{Indice de Jaccard})$$



Calculer ce coefficient avec l'ensemble de nos données est beaucoup trop coûteux. [Broder, 1997] introduit la méthode MinHash qui permet d'approcher cet indice à l'aide de fonctions de hachage. En appliquant une fonction de hachage à  $A$  et  $B$ , la probabilité que la valeur minimale du hash de  $A$  et la valeur minimale du hash de  $B$  soit égale est exactement  $J(A, B)$  !

# COLLECTER ET TRANSFORMER DU TEXTE

## ALGORITHME MINHASH

Texte 1	Texte 2	Texte 3	Texte 4
1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

**Table** – Vecteurs représentant des textes

On considère les permutations :

$$P1 = [1, 3, 7, 6, 2, 5, 4], \quad P2 = [4, 2, 1, 3, 6, 7, 5], \quad P3 = [1, 4, 7, 6, 1, 2, 5]$$



# COLLECTER ET TRANSFORMER DU TEXTE

## ALGORITHME MINHASH

On obtient alors :

T1	T2	T3	T4
1	0	1	0
0	1	0	1
1	0	1	0
1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1

T1	T2	T3	T4
0	1	0	1
1	0	0	1
1	0	1	0
0	1	0	1
1	0	1	0
1	0	1	0
0	1	0	1

T1	T2	T3	T4
1	0	1	0
0	1	0	1
1	0	1	0
1	0	1	0
1	0	1	0
1	0	0	1
0	1	0	1

## COLLECTER ET TRANSFORMER DU TEXTE

### COMMENT CONSTITUER UN CORPUS ?

On calcule la signature en prenant le premier indice où la valeur est 1 :

S1	S2	S3	S4
1	2	1	2
2	1	4	1
2	1	2	1

Avec cette table, on estime par exemple la similarité de T1 et T3 à 0.66 alors que la véritable similarité est 0.75 : c'est une bonne approximation.

FineWeb collecte l'ensemble des 5-grams de chaque document et calcule 112 fonctions de hash réparties dans 14 blocs de 8 hashes chacun. Ainsi, si deux documents présentent similarité  $s$ , la probabilité qu'ils soient effectivement identifiés comme similaires est :

$$\mathbb{P}(\text{documents similaire identifié}) = (1 - (1 - s^8)^{14})$$

# COLLECTER ET TRANSFORMER DU TEXTE

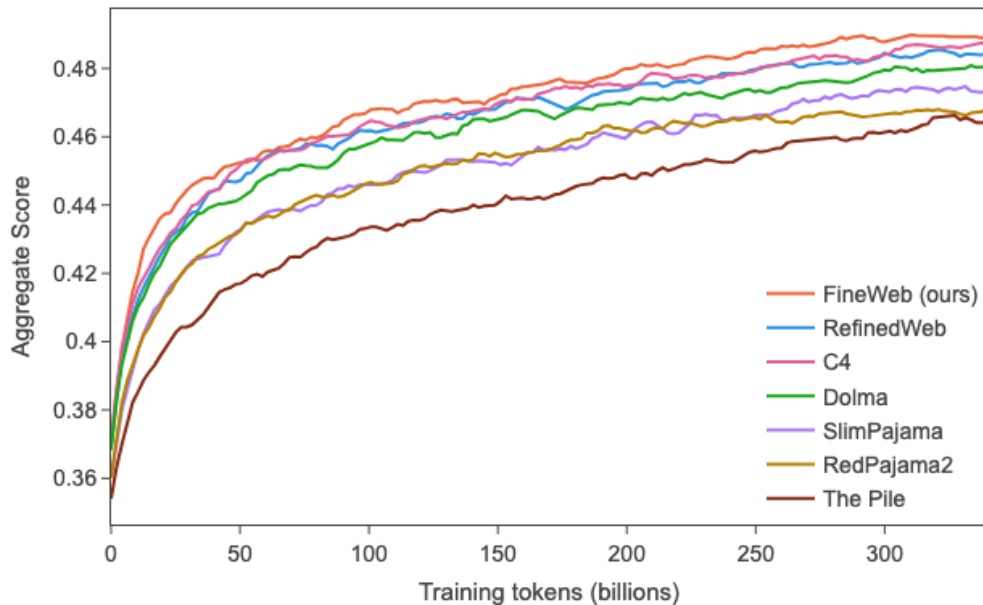
## COMMENT CONSTITUER UN CORPUS ?

Modèle	Année	Tokens d'entraînement
Chinchilla	Mars 2022	1.3T
PaLM	Avril 2022	768B
phi-1	Novembre 2022	6B
LLaMa	Février 2023	1.4T
PaLM 2	Mai 2023	3.6T
Falcon	Juin 2023	5T
LLaMa 2	Juillet 2023	2T
Gemma	Février 2024	6T
LLaMa 3	Avril 2024	15T

**Table** – Nombre de token d'entraînement pour quelques grands modèles de langage

# COLLECTER ET TRANSFORMER DU TEXTE

## COMMENT CONSTITUER UN CORPUS ?



**Figure** – Performances d’un modèle de langage en fonction du nombre de token d’entraînement et selon le dataset. Source : [Penedo et al., 2024]

# COLLECTER ET TRANSFORMER DU TEXTE

## COMMENT CONSTITUER UN CORPUS ?

Modèle	Page Web	Code	Encyclopédique	Livres	Académique	Réseaux sociaux	Langue
LLaMa	82	4.5	4.5	4.5	2.5	2.0	0
GPT-3	60	22	3	15	0	0	0
PaLM	27	50	4	13	0	5	1
Gopher	58	3	2	27	0	0	10
Chinchilla	55	4	1	30	0	0	10
Falcon	100	0	0	0	0	0	0
phi-1	0	100	0	0	0	0	0
LaMDA	12.5	50	12.5	0	0	12.5	12.5

**Table** – Composition du corpus d’entraînement (en %). Source : [Liu et al., 2024]

Être capable de constituer un dataset diversifié selon les sources tout en conservant la qualité est un véritable défi. TxT360 [Liping Tang, 2024] adresse spécifiquement ce sujet.

## COLLECTER ET TRANSFORMER DU TEXTE

### COMMENT CONSTITUER UN CORPUS ?

Langue	Tokens dans mC4		Locuteurs natifs	
	Nombre (en milliards)	Proportion (%)	Nombre (en millions)	Proportion (%)
Anglais	2000	45.5	379	5.1
Russe	250	5.7	154	2.1
Allemand	200	4.5	95	1.3
Français	170	3.9	76.8	1.0
Espagnol	160	3.6	460	6.2
Chinois simplifié	150	3.4	1300	17.6
Portugais	120	2.7	234	3.2
Italien	100	2.3	59.8	0.8
Polonais	90	2.0	46.6	0.6
Japonais	80	1.8	128	1.7

**Table** – 10 langues les plus représentées dans mC4, sources Ethnologue et [Raffel et al., 2019]

# COLLECTER ET TRANSFORMER DU TEXTE

COMMENT RENDRE INTELLIGIBLE DU TEXTE POUR UN MODÈLE DE LANGAGE ?

Exemple de tokenization pour un modèle Mistral :

Une	phrase	en	français
16803	15572	1249	15067

Ou pour un modèle Gemma :

Une	phrase	en	français
19750	20911	659	24913

# COLLECTER ET TRANSFORMER DU TEXTE

## COMMENT RENDRE INTELLIGIBLE DU TEXTE POUR UN MODÈLE DE LANGAGE ?

[Sennrich et al., 2015] adapte l'algorithme Byte-Pair Encoding (BPE) initialement construit pour la compression [Gage, 1994]. Cet algorithme a besoin du résultat d'un tokenizer : un texte tokenisé et le vocabulaire associé, mais également d'une taille de vocabulaire souhaité.

### Exercice 1 (Byte-Pair Encoding)

On considère le vocabulaire suivant  $V = [a, c, e, h, i, n, p, s, t]$  et les mots suivant avec leurs fréquences :

$(\text{"chat"}, 5), (\text{"chats"}, 3), (\text{"chien"}, 2), (\text{"patte"}, 5)$

1. Après avoir écrit les mots avec le vocabulaire de base, quelle est la paire la plus fréquente ?
2. Réécrire les mots avec un nouveau symbole, associé à la paire la plus fréquente.
3. Recommencer les deux premières étapes.
4. Quelle était la longueur du texte avant ? Et maintenant ?



## COLLECTER ET TRANSFORMER DU TEXTE

### COMMENT RENDRE INTELLIGIBLE DU TEXTE POUR UN MODÈLE DE LANGAGE ?

WordPiece est introduit initialement pour un problème de traduction Japonais-Coréen [Schuster and Nakajima, 2012] puis décrit plus en détail dans [Wu et al., 2016]. L'algorithme est très similaire à BPE et se différencie par le choix de la paire à former.

En reprenant l'exercice précédent, la paire la plus fréquente reste "at", mais comme les caractères "a" et "t" sont présent souvent, on obtient un score de :

$$S("at") = \frac{\text{Fréquence de "at"}}{\text{Fréquence de "a"} \times \text{Fréquence de "t"}} = \frac{13}{13 \times 18} \simeq 0.05$$

La paire "ch" est la plus vraisemblable :

$$S("ch") = \frac{10}{10 \times 10} = 0.1$$

## COLLECTER ET TRANSFORMER DU TEXTE

### COMMENT RENDRE INTELLIGIBLE DU TEXTE POUR UN MODÈLE DE LANGAGE ?

L'algorithme SentencePiece [Kudo and Richardson, 2018] a été proposé pour résoudre un problème dont les précédents souffrent : les mots ne sont pas forcément séparés par des espaces dans toutes les langues. Ainsi, SentencePiece inclut les espaces dans le jeu de caractères à utiliser.

Modèle	Année	Algorithme	Vocabulaire
Bert	2018	WordPiece	30k
GPT2	2019	BPE	50k
LLaMa 1	2023	SentencePiece	32k
LLaMa 2	2023	SentencePiece	32k
Mistral 7B	2023	SentencePiece	32k
Mixtral 8x7B	2024	SentencePiece	32k
Gemma	2024	SentencePiece	256k

**Table** – Algorithme de tokenization et taille du vocabulaire

Puisque le tokenizer doit être public, [Hayase et al., 2024] montre que l'on peut inférer à partir des règles de tokenization la composition du dataset d'entraînement.

# COLLECTER ET TRANSFORMER DU TEXTE

## EN RÉSUMÉ

- 1 Collecter et transformer du texte . . . . . 3**
  - 1.1 Comment constituer un corpus? . . . . . 3
  - 1.2 Comment rendre intelligible du texte pour un modèle de langage? . . . . . 14
- 2 Alimenter un modèle avec des tokens . . . . . 19**
- 3 Quelques tendances . . . . . 28**

## ALIMENTER UN MODÈLE AVEC DES TOKENS

### EMBEDDING : DU NOMBRE VERS LE VECTEUR

Un embedding est une manière de représenter un mot par un vecteur dense de taille fixe. Par exemple avec trois modalités pour quatre observations, le one-hot-encoding le fait avec des vecteurs *sparse* :

$$\begin{pmatrix} \text{reine} \\ \text{roi} \\ \text{chien} \\ \text{reine} \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Ici, les trois mots sont à distances égales donc représente mal les relations entre les mots. De plus, la taille de la matrice d'embedding peut être très grande. Ce n'est pas une solution satisfaisante.

# ALIMENTER UN MODÈLE AVEC DES TOKENS

## EMBEDDING : DU NOMBRE VERS LE VECTEUR

Pour une séquence de longueur  $n \in \mathbb{N}^*$  tokenizé avec un vocabulaire de taille  $V$ , que l'on veut représenter par un embedding de taille  $d \in \mathbb{N}^*$ , on a :

$$E = S \times M$$

Matrice OHE de la séquence de tokens de taille  $n \times V$

Embedding de taille  $n \times d$

Matrice d'embedding de taille  $V \times d$

Les valeurs de la matrice  $M$  sont ceux que le modèle cherche à apprendre pour représenter au mieux les mots. Nous avons deux choix :

- ▶ **Apprendre** l'embedding au cours de l'entraînement
- ▶ **Réutiliser** un embedding pré-entraîné

# ALIMENTER UN MODÈLE AVEC DES TOKENS

## ET LA POSITION ?

Le mécanisme au coeur des modèles de langage type Transformers n'est pas capable de *comprendre* la position d'un token *nativement*. Pour la phrase<sup>2</sup> :

Un	ét	udiant	expl	ique	à	un	autre	ét	udiant
1844	14240	41939	3327	2428	3869	653	47838	14240	41939

La représentation de *étudiant* sera identique dans l'espace d'embedding : il sera impossible pour le modèle d'identifier qu'il s'agit de deux personnes différentes.

---

2. Tokenisé selon un modèle LLaMa 3

## ALIMENTER UN MODÈLE AVEC DES TOKENS

### ET LA POSITION ?

La solution proposée initialement dans l'article et se résume avec ces équations :

Position du token

$$\begin{cases} PE(x, 2i) = \sin\left(\frac{x}{10000^{2i/d}}\right) \\ PE(x, 2i + 1) = \cos\left(\frac{x}{10000^{2i/d}}\right) \end{cases}$$

Indice du vecteur d'embedding

Taille du vecteur d'embedding

Cette nouvelle matrice d'embedding ainsi produite est ajoutée à l'embedding précédent pour enrichir les informations que l'on fournit au modèle.

## ALIMENTER UN MODÈLE AVEC DES TOKENS

### AJUSTER LES PROBABILITÉS

Considérons un modèle de langage qui pour une séquence de token prédit le prochain. Alors il produit un vecteur de taille  $d \in \mathbb{N}$  avec  $d$  le nombre de token dans le vocabulaire du modèle représentant un vecteur de probabilité :

$i$ -ème valeur initiale du vecteur

$$p_i = \frac{\exp\left(\frac{-\varepsilon_i}{\tau}\right)}{\sum_{j=1}^d \exp\left(\frac{-\varepsilon_j}{\tau}\right)}$$

Température

Le paramètre  $\tau$  est appelé la température par inspiration du domaine de la thermodynamique en physique. Étudions plus en détail les possibilités de cette fonction d'activation.



# ALIMENTER UN MODÈLE AVEC DES TOKENS

## AJUSTER LES PROBABILITÉS

### Exercice 2 (Température)

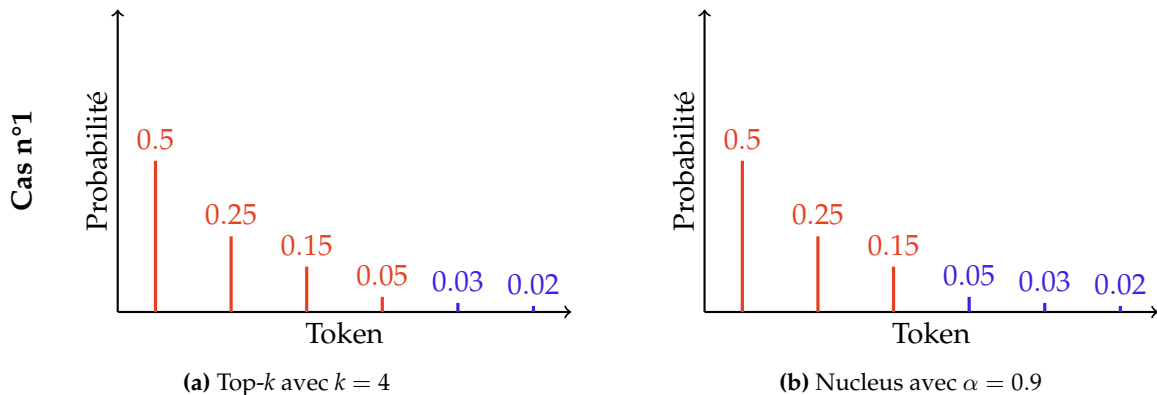
On considère  $\varepsilon_1, \varepsilon_2 \in \mathbb{R}$  tels que  $\varepsilon_1 < \varepsilon_2$ . On définit  $0 < \tau_1 < \tau_2$  deux températures.

1. Pour une température  $\tau > 0$  fixée, on note  $p_1$  et  $p_2$  les valeurs associées à la transformation softmax de  $\varepsilon_1$  et  $\varepsilon_2$ . A-t-on que  $p_1 < p_2$  ?
2. Comment varie la valeur de  $p$  quand  $\tau$  varie ?
3. Calculer  $\lim_{\tau \rightarrow +\infty} p_i$ .

# ALIMENTER UN MODÈLE AVEC DES TOKENS

## AJUSTER LES PROBABILITÉS

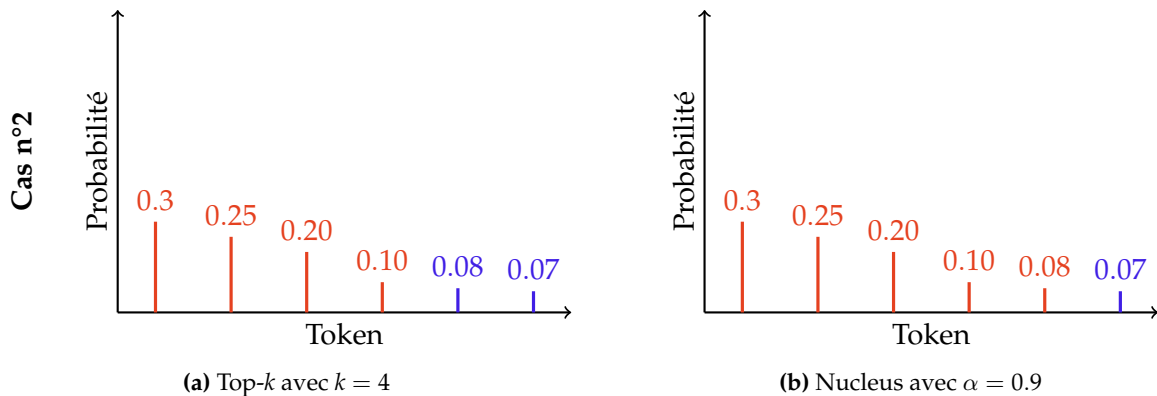
[Fan et al., 2018] propose la méthode *top-k sampling* et [Holtzman et al., 2019] proposent le *nucleus sampling*.



**Figure** – Exemple de deux stratégies de sampling pour **sélections** les tokens

## ALIMENTER UN MODÈLE AVEC DES TOKENS

### AJUSTER LES PROBABILITÉS



**Figure** – Exemple de deux stratégies de sampling pour **sélections** les tokens

Une valeur bien choisie pour  $\tau$  permet de mieux calibrer la valeur  $\alpha$  du nucleus sampling. En revanche, cela n'a pas d'impact pour le top- $k$  sampling.

# ALIMENTER UN MODÈLE AVEC DES TOKENS

## EN RÉSUMÉ

<b>1</b>	<b>Collecter et transformer du texte</b>	<b>3</b>
<b>2</b>	<b>Alimenter un modèle avec des tokens</b>	<b>19</b>
2.1	Embedding : du nombre vers le vecteur	19
2.2	Et la position?	21
2.3	Ajuster les probabilités	23
<b>3</b>	<b>Quelques tendances</b>	<b>28</b>

## QUELQUES TENDANCES

### LES *Scaling laws*

[Kaplan et al., 2020] propose les premières *Scaling Laws* pour les réseaux de neurones avec une architecture transformers. Il existe plusieurs équations dans ces lois, nous n'en considérerons ici que deux :

Nombre de paramètres

$$\mathcal{L}(N) = \left( \frac{N_c}{N} \right)^{\alpha_N}$$

$$\text{avec } \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13}$$

$$\mathcal{L}(D) = \left( \frac{D_c}{D} \right)^{\alpha_D}$$

$$\text{avec } \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13}$$

Taille du dataset

Le troisième paramètre identifié pour obtenir des performances est le budget de calcul  $C$ , compté en FLOPS, qui a lui aussi sa loi :

$$FLOPS(N, D) = k ND$$

Constante dépendante du matériel

## QUELQUES TENDANCES

### LES *Scaling laws*

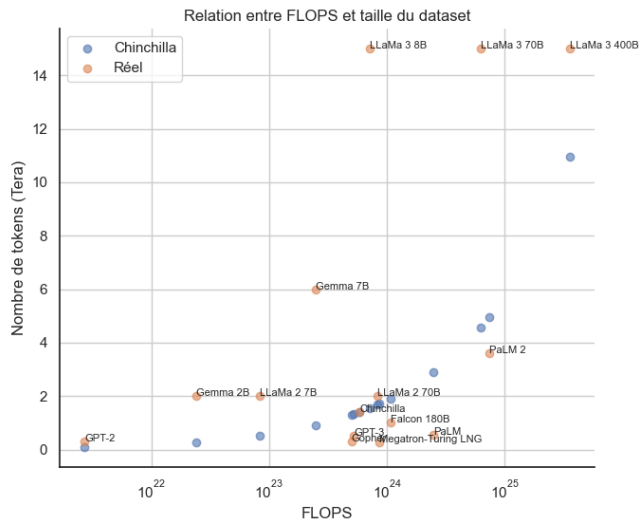
A partir des résultats précédents, [Kaplan et al., 2020] conclut que l'optimal est une taille de modèle  $N$  de l'ordre de  $C^{0.73}$  et une taille de dataset  $D$  de l'ordre de  $C^{0.27}$ .

Cependant, une étude de Deepmind propose d'autres valeurs pour les scaling laws, et les *prouve* avec son modèle Chinchilla [Hoffmann et al., 2022]. Selon Deepmind on a cette fois  $N$  est de l'ordre de  $C^{0.5}$  et  $D$  est de l'ordre de  $C^{0.5}$ . Autrement dit, il faut entraîner un modèle sur plus de tokens!

Pour comprendre l'écart entre les deux prédictions, [Pearce and Song, 2024] pointe deux principales raisons :

1. Le nombre de paramètre n'est pas calculé de la même manière : les lois de Kaplan excluent les paramètres d'embedding alors que Chinchilla les inclut
2. La taille des modèles utilisés par Kaplan pour apprendre les bons paramètres de lois étaient de trop petites tailles et ont donc sous-estimé l'impact du dataset

## LES *Scaling laws*



30 / 38

## QUELQUES TENDANCES

### LA quantization

Les modèles de langage sont stockés sous forme de fichiers contenant énormément de nombre flottant. Que se passerait-il si on stockait ces mêmes nombres mais avec un nombre de bits plus restreint ? Cela s'appelle la **quantization**.

[Kumar et al., 2024] montre qu'un modèle quantisé jusqu'à 4 bit obtient des performances similaires au modèle avec la précision maximale. Une quantization plus prononcée dégrade très fortement en revanche la qualité du modèle.

Modèle	Initial	Q8	Q6	Q4
LLaMa 3.2 3B	6.43	3.42	2.74	2.11
LLaMa 3.2 1B	2.48	1.32	1.09	0.87
Mistral 7B v0.3	14.50	7.70	5.95	4.37

**Table** – Taille du fichier (GB) contenant le modèle selon le niveau de quantization



## QUELQUES TENDANCES

### LA quantization

Les modèles les plus récents semblent privilégier la quantité de donnée au nombre de paramètres pour un budget de calcul fixé<sup>3</sup>. L'article propose l'équation suivante pour modéliser l'impact sur la performance du modèle d'une quantization de précision  $P$  notée  $\delta$  :

$$\delta(N, D, P) = C \left( \frac{D^{\gamma_D}}{N^{\gamma_N}} \right) e^{-\frac{P}{\gamma_P}}$$

Où  $C$ ,  $\gamma_D$ ,  $\gamma_N$  et  $\gamma_P$  sont des constantes positives.

### Exercice 3 (Taille critique)

On considère un modèle entraîné en pleine précision, puis il est quantifié. La loi d'échelle s'écrit comme :

$$\mathcal{L}(N, D, P) = AN^{-\alpha} + BD^{-\beta} + \delta_{PTQ}(N, D, P)$$

Trouver la taille  $D_{crit}$  telle que ajouter des tokens lors de l'entraînement va réduire la performance du modèle quantifié.

---






3. On dit que les modèles sont sur-entraînés.

# QUELQUES TENDANCES






## EN RÉSUMÉ

<b>1</b>	<b>Collecter et transformer du texte</b>	<b>3</b>
1.1	Comment constituer un corpus?	3
1.2	Comment rendre intelligible du texte pour un modèle de langage?	14
<b>2</b>	<b>Alimenter un modèle avec des tokens</b>	<b>19</b>
2.1	Embedding : du nombre vers le vecteur	19
2.2	Et la position?	21
2.3	Ajuster les probabilités	23
<b>3</b>	<b>Quelques tendances</b>	<b>28</b>
3.1	Les <i>Scaling laws</i>	28
3.2	La <i>quantization</i>	31






## BIBLIOGRAPHIE I

-  [Broder, A. Z. \(1997\).](#)  
**On the resemblance and containment of documents.**  
*In Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171).*
-  [Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. \(2022\).](#)  
**Quantifying memorization across neural language models.**  
*arXiv preprint arXiv :2202.07646.*
-  [Fan, A., Lewis, M., and Dauphin, Y. \(2018\).](#)  
**Hierarchical neural story generation.**  
*arXiv preprint arXiv :1805.04833.*
-  [Gage, P. \(1994\).](#)  
**A new algorithm for data compression.**  
*The C Users Journal.*
-  [Hayase, J., Liu, A., Choi, Y., Oh, S., and Smith, N. A. \(2024\).](#)  
**Data mixture inference : What do bpe tokenizers reveal about their training data ?**  
*arXiv preprint arXiv :2407.16607.*






## BIBLIOGRAPHIE II

-  Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022).  
**Training compute-optimal large language models.**  
*arXiv preprint arXiv :2203.15556.*
-  Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2019).  
**The curious case of neural text degeneration.**  
*arXiv preprint arXiv :1904.09751.*
-  Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016).  
**Bag of tricks for efficient text classification.**  
*arXiv preprint arXiv :1607.01759.*
-  Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020).  
**Scaling laws for neural language models.**  
*arXiv preprint arXiv :2001.08361.*
-  Kudo, T. and Richardson, J. (2018).  
**Sentencepiece : A simple and language independent subword tokenizer and detokenizer for neural text processing.**  
*arXiv preprint arXiv :1808.06226.*

## BIBLIOGRAPHIE III

-  Kumar, T., Ankner, Z., Spector, B. F., Bordelon, B., Muennighoff, N., Paul, M., Pehlevan, C., Ré, C., and Raghunathan, A. (2024).  
**Scaling laws for precision.**  
*arXiv preprint arXiv :2411.04330.*
-  Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. (2021).  
**Deduplicating training data makes language models better.**  
*arXiv preprint arXiv :2107.06499.*
-  Liping Tang, Nikhil Ranjan, O. P. X. L. Z. W. L. A. B. R. L. J. H. W. Z. C. S. S. C. M. V. M. X. M. Y. P. Z. L. E. P. X. (2024).  
**Txt360 : A top-quality llm pre-training dataset requires the perfect blend.**  
Hugging Face Space.
-  Liu, Y., Cao, J., Liu, C., Ding, K., and Jin, L. (2024).  
**Datasets for large language models : A comprehensive survey.**  
*arXiv preprint arXiv :2402.18041.*
-  Pearce, T. and Song, J. (2024).  
**Reconciling kaplan and chinchilla scaling laws.**  
*arXiv preprint arXiv :2406.12907.*

## BIBLIOGRAPHIE IV

-  [Penedo, G., Kydlíček, H., von Werra, L., and Wolf, T. \(2024\).](#)  
**Fineweb.**
-  [Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. \(2021\).](#)  
**Scaling language models : Methods, analysis & insights from training gopher.**  
*arXiv preprint arXiv :2112.11446.*
-  [Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. \(2019\).](#)  
**Exploring the limits of transfer learning with a unified text-to-text transformer.**  
*Journal of machine learning research.*
-  [Schuster, M. and Nakajima, K. \(2012\).](#)  
**Japanese and korean voice search.**  
*In 2012 IEEE international conference on acoustics, speech and signal processing (ICASSP).*
-  [Sennrich, R., Haddow, B., and Birch, A. \(2015\).](#)  
**Neural machine translation of rare words with subword units.**  
*arXiv preprint arXiv :1508.07909.*

## BIBLIOGRAPHIE V

 Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016).

**Google's neural machine translation system : Bridging the gap between human and machine translation.**

*arXiv preprint arXiv :1609.08144.*