

# Plongée profonde dans les LLM comme ChatGPT



## Comment fonctionnent les LLM par Andrej Karpathy

transcription par André Cabannes



Les Éditions du Bec de l'Aigle

Transcription traduite en français de la vidéo disponible à <https://www.youtube.com/watch?v=7xTGNNLPyMI&t=2s>, publiée en février 2025<sup>1</sup>.

Version de la transcription du

**24 octobre 2025**

Si vous avez obtenu ce livre gratuit par une autre source que notre site web, veuillez vous assurer que vous disposez de la version la plus récente téléchargeable ici: [https://lapasserelle.com/documents/comment\\_fonctionnent\\_les\\_LLM.pdf](https://lapasserelle.com/documents/comment_fonctionnent_les_LLM.pdf)

---

<sup>1</sup>Préparée par <https://lapasserelle.com/webmaster.html>.

Cette transcription est une version légèrement éditée de la vidéo pour une meilleure lisibilité. Elle incorpore des illustrations provenant de la vidéo.

Les notes de bas de page sont du transcripteur.

# Table des matières

|                                                                                                     |     |
|-----------------------------------------------------------------------------------------------------|-----|
| 1. Introduction .....                                                                               | 2   |
| 2. Données de pré-entraînement (Internet) .....                                                     | 3   |
| 3. Tokénisation (découpage d'un texte en jetons) .....                                              | 9   |
| 4. Réseau neuronal, Input/Output .....                                                              | 17  |
| 5. Entrailles d'un réseau neuronal .....                                                            | 21  |
| 6. Inférence .....                                                                                  | 27  |
| 7. GPT-2 : entraînement et inférence .....                                                          | 31  |
| 8. Inférence avec le modèle de base Llama 3.1 .....                                                 | 42  |
| 9. Du pré-entraînement au post-entraînement .....                                                   | 61  |
| 10. Données pour le post-entraînement (conversations) ..                                            | 67  |
| 11. Hallucinations, utilisation d'outils, connaissances<br>à long terme et mémoire de travail ..... | 82  |
| 12. Connaissance de soi .....                                                                       | 102 |
| 13. Les modèles ont besoin de jetons pour penser .....                                              | 107 |
| 14. Tokénisation revisitée : difficultés pour épeler<br>les mots et pour compter .....              | 122 |
| 15. Récapitulation : du modèle de base à l'assistant ....                                           | 128 |
| 16. Du fine-tuning supervisé à l'apprentissage renforcé ..                                          | 130 |
| 17. Apprentissage renforcé .....                                                                    | 139 |
| 18. DeepSeek-R1 .....                                                                               | 145 |
| 19. AlphaGo .....                                                                                   | 162 |
| 20. Apprentissage renforcé par feedback humain (RLHF)                                               | 168 |
| 21. Avantages et inconvénients du RLHF .....                                                        | 176 |
| 22. Aperçu de l'avenir .....                                                                        | 185 |
| 23. Suivre le développement des LLM .....                                                           | 189 |
| 24. Où trouver des LLM .....                                                                        | 193 |
| 25. Résumé et conclusion .....                                                                      | 197 |

## 1. Introduction

Bonjour tout le monde,

Je voulais faire cette vidéo depuis quelque temps. C'est une introduction complète, pour le grand public, présentant les grands modèles de langage (LLM) comme ChatGPT <sup>1</sup>.

Avec cette vidéo, j'espère vous donner un modèle mental pour réfléchir à ce qu'est cet outil. Beaucoup de gens s'émerveillent sur l'aspect magique d'un LLM. En réalité il est performant pour certaines choses, moins pour d'autres. Il faut apprendre à s'en servir pour en tirer le meilleur parti.

Qu'y a-t-il derrière cette boîte de texte ? (fig. 1)

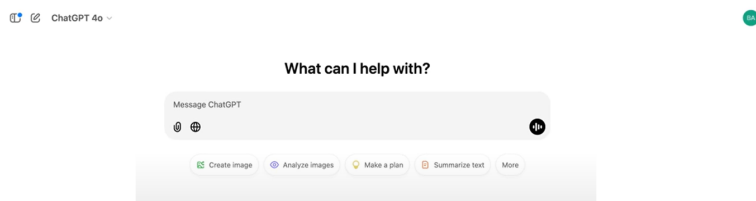


Figure 1 : Page principale de ChatGPT permettant de saisir une question, un texte, une invite, une image ou tout autre élément pour déclencher une réponse du LLM.

Saisissez ce que vous voulez et pressez la touche "Entrée".

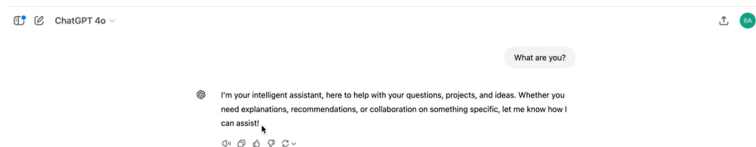


Figure 2 : Réponse de ChatGPT 4o à la question "Qu'est-ce que tu es?".

---

1. Pendant son exposé, Andrej Karpathy utilise un grand tableau blanc avec des illustrations. On peut l'obtenir en utilisant <https://excalidraw.com/> et en ouvrant le fichier <https://lapasserelle.com/documents/karpathy.txt>



Que devons-nous y mettre, et que représentent exactement les mots qui sont générés en retour? (fig. 2) Comment cela fonctionne-t-il? À qui ou à quoi parle-t-on au juste?

J'espère aborder tous ces sujets dans cette vidéo. Nous allons parcourir tout le processus création d'un LLM. Mais je veillerai à ce que tout reste accessible à un large public.

Commençons par examiner comment on construit quelque chose comme ChatGPT. Au passage, je parlerai de certaines des implications cognitives et psychologiques de ces outils.

00 :01 :00<sup>2</sup>

## 2. Données de pré-entraînement (Internet)

Construisons ChatGPT. Il va y avoir plusieurs phases.

La première phase est le *pré-entraînement*. Et la première étape de la phase de pré-entraînement consiste à télécharger et traiter tout l'Internet! Cf. fig. 3.

### Step 1: download and preprocess the internet

<https://huggingface.co/spaces/HuggingFaceFW/blogpost-finetweb-v1>

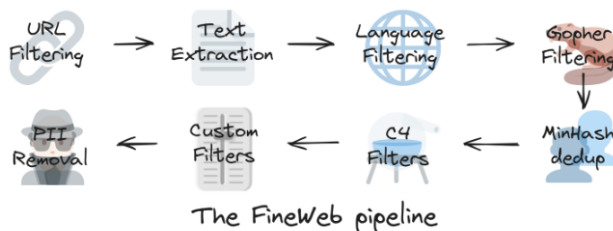


Figure 3 : Première étape du *pré-entraînement* : télécharger et traiter tout l'Internet.

---

2. Nous indiquons parfois le moment approximatif de la vidéo correspondant à notre position dans le texte. À ce stade, nous en sommes à 00 :01 :00 dans la vidéo.

Pour se figurer en quoi cela consiste, je recommande de regarder le site <https://huggingface.co/spaces/HuggingFaceFW/blogpost-fineweb-v1>

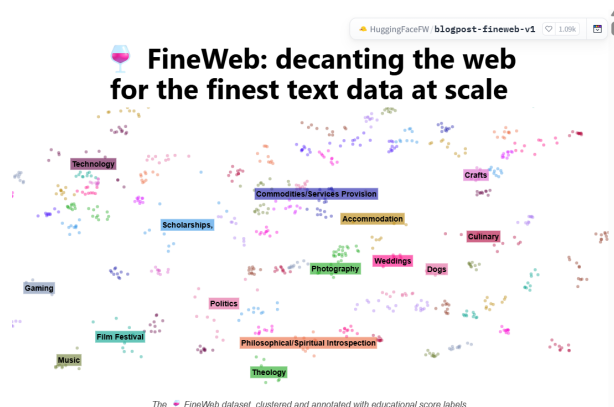


Figure 4 : Page d'accueil de [huggingface.co](https://huggingface.co)

L'entreprise *Hugging Face* a collecté, créé et structuré un ensemble de données appelé *Fineweb*. Dans l'article de blog ci-dessus, elle explique en détail la manière dont elle a construit cet ensemble de données.

Tous les principaux fournisseurs de LLM, comme OpenAI, Anthropic, Google, etc., disposent d'un équivalent en interne, similaire à l'ensemble de données Fineweb.

Notre objectif est d'extraire une quantité importante de textes provenant d'Internet, de sources accessibles au public. Nous souhaitons nous procurer une quantité considérable de documents de haute qualité. Nous souhaitons également une grande diversité de documents, car nous voulons intégrer une grande quantité de connaissances dans ces LLM. Nous souhaitons donc une grande diversité de documents de haute qualité, et ce, en très grand nombre.

Réaliser cet objectif est assez complexe et, comme vous pouvez le voir sur la figure 3, plusieurs étapes sont nécessaires pour y parvenir.

00 :02 :00

Examinons brièvement certaines de ces étapes.

Pour l’instant, je voudrais souligner que l’ensemble de données Fineweb – représentatif de ce que l’on peut trouver dans une application commerciale – ne prend que 44 téraoctets d’espace disque. On trouve facilement des clés USB d’un téraoctet (1 To). Et Fineweb pourrait presque tenir sur un seul disque dur actuel. Au final, ce n’est donc pas une quantité énorme de données.

Même si Internet est très vaste, nous ne travaillons qu’avec du texte. Et nous le filtrons sévèrement.

Examinons à quoi ressemblent ces données et certaines de ces étapes. Le point de départ de beaucoup de constructeurs de LLM est les données de *Common Crawl*.

Common Crawl est une organisation qui analyse Internet depuis 2007. En 2024, Common Crawl avait indexé 2,7 milliards de pages web. Tous ces robots d’indexation circulent sur Internet. On amorce le "crawling" avec quelques pages web sources, puis on suit tous les liens et on indexe toutes les informations. Au fil du temps, on amasse une masse considérable de données provenant d’Internet. C’est généralement le point de départ.

Les données de Common Crawl sont relativement brutes et doivent être filtrées de différentes manières. La figure 3 montre schématiquement les différentes étapes de traitement.

La première étape est le *filtrage d’URL*. Il existe des listes noires d’URL ou de domaines dont vous ne souhaitez pas récupérer les données. Le site <https://dsi.ut-capitole.fr/blacklists/> donne une liste de sites indésirables. Cela inclut généralement les sites web malveillants, les sites de spam, les sites de marketing, les sites racistes, les sites pour adultes, etc.

De nombreux types de sites web sont donc éliminés à ce stade, car nous ne les souhaitons pas dans notre ensemble de données qui va servir au pré-entraînement de notre LLM.

00 :04 :00

La deuxième étape est l'*extraction de texte*. Il faut garder à l'esprit que toutes ces pages web sont du code HTML brut : c'est ainsi que les robots d'exploration les enregistrent. Par exemple, lorsque j'inspecte la page du blog « Hugging Face », voici à quoi ressemble le code HTML brut.

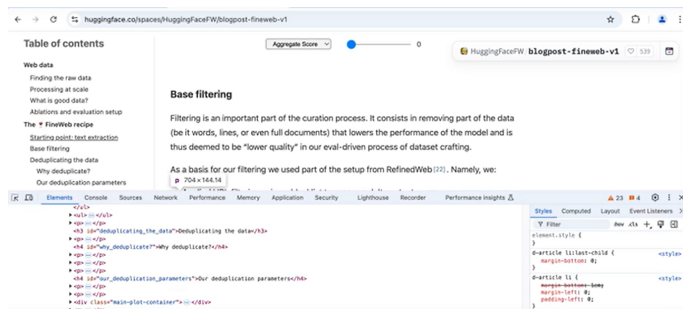


Figure 5 : Exemple de code HTML. Dans la plupart des cas, lorsque vous consultez une page web, vous pouvez inspecter son code HTML en faisant un clic droit sur la page et en sélectionnant « Afficher le code source » dans le menu contextuel.

Vous remarquerez qu'il contient tout un balisage, comme des listes (`<ul> ... </ul>` avec chaque élément dans `<li> ... </li>`), etc. Il y a aussi du code CSS. Il s'agit du code informatique permettant de construire les pages web HTML.

Ce que nous voulons c'est le texte visible affiché par le navigateur, pas le code HTML sous-jacent avec ses balises d'affichage. Nous nous intéressons uniquement à la « chair », pas à l'ossature. Les menus de navigation et autres éléments structurels sont également ignorés. Extraire le contenu pertinent des pages web nécessite un filtrage et un traitement importants pour isoler le texte de qualité, intéressant et utile.

L'étape suivante est le *filtrage linguistique*. Par exemple, les filtres Fineweb, à l'aide d'un classificateur de langues, tentent de deviner la langue de chaque page web. Ils ne conservent que les pages web dont la part d'anglais est supérieure à, par exemple, 65%.

Il s'agit d'une décision prise par chaque entreprise construisant et donnant accès à un LLM : quelle proportion de tous les types de langues allons-nous inclure dans notre ensemble de données pour le pré-traitement de notre LLM ?

En effet, si nous filtrons, par exemple, presque tous les textes en espagnol, on s'imagine bien que le modèle ultérieurement ne sera pas très performant en espagnol, car il n'aura jamais traité beaucoup de données dans cette langue.

Les entreprises peuvent choisir de se concentrer sur plus ou moins de performances multilingues.

Fineweb est focalisé sur l'anglais. Un LLM entraîné sur Fineweb sera très performant en anglais, mais généralement moins performant dans d'autres langues.

Après le filtrage linguistique, il y a encore d'autres étapes de filtrage, par exemple le filtrage des doublons. On veut enfin supprimer les informations personnelles identifiables [en anglais PII]. Il s'agit d'informations comme des adresses, numéros de sécurité sociale, etc. On essaie de les détecter et de filtrer ces types de pages web de l'ensemble de données.

00 :06 :00

En résumé, le pré-traitement (le filtrage pour obtenir un corpus propre de textes de haute qualité) comporte de nombreuses étapes (fig. 3).

Je n'entrerai pas dans les détails, mais il s'agit d'une partie assez importante du pré-traitement. On obtient ainsi, par exemple, l'ensemble de données Fineweb.

En cliquant dessus (et en accédant à <https://huggingface.co/datasets/HuggingFaceFW/fineweb>), vous pouvez voir quelques exemples de ce à quoi cela ressemble réellement (fig. 6).

Tout le monde peut le télécharger sur la page web de Hugging Face.

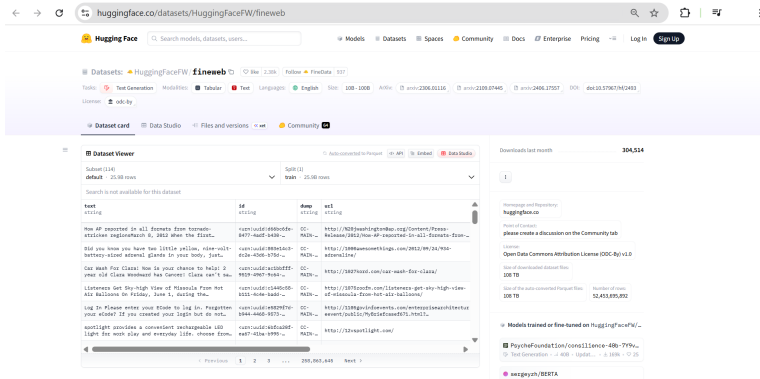


Figure 6 : Contenu de Fineweb.

Regardons quelques exemples de texte final se retrouvant dans l'ensemble d'entraînement. Voici un article sur les tornades de 2012. Il y a eu des tornades en 2012 et l'article décrit ce qui s'est passé. L'article suivant porte sur... savez-vous que vous aviez deux petites glandes surrénales jaunes de la taille d'une pile parallélépipédique neuf volts dans votre corps? Bon, c'est un article médical.

Imaginez tout cela comme des pages web sur Internet, filtrées uniquement pour le texte (le « contenu ») de diverses manières. Et maintenant, nous avons des tonnes de texte : 44 téraoctets pour être précis.

Nous sommes maintenant au début de l'étape suivante de la phase de pré-entraînement.

Pour vous donner une idée concrète de là où nous en sommes, j'ai pris les 200 premières pages web dont la liste commence en bas à gauche de la figure 6 — et rappelez-vous que nous en avons une quantité gigantesque.

J'ai simplement pris tout le texte des 200 premières pages mentionnées ; je l'ai assemblé, concaténé ; et voici ce que nous obtenons, voir la figure 7 : c'est du texte, du texte Internet nettoyé mais "brut", et il y en a une quantité considérable, même dans ces 200 pages web.

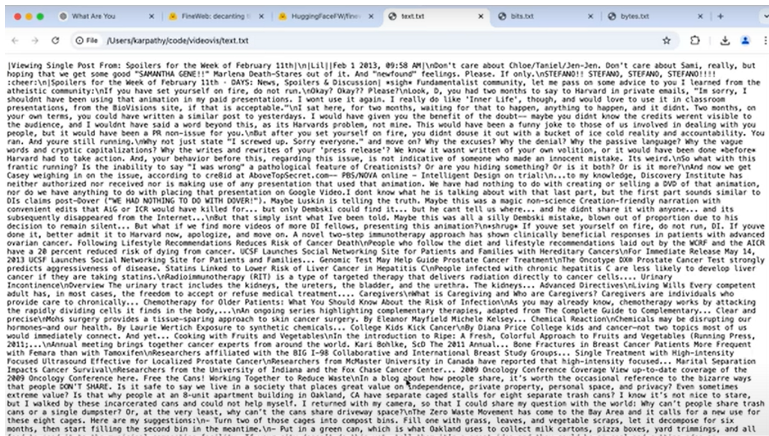


Figure 7 : Vue du début de 44 téraoctets de texte fourni par Fineweb.

Je peux continuer à dézoomer [en regardant de plus en plus haut] : nous avons cette immense mosaïque de données textuelles. Et ces données textuelles présentent toutes sortes de schémas.

*Nous souhaitons maintenant commencer à entraîner les réseaux neuronaux sur ces données, afin qu'ils puissent internaliser et modéliser le flux de ce texte.*

Nous avons cette immense « texture textuelle », et nous voulons maintenant obtenir des réseaux neuronaux qui l'imitent. Ce que nous entendons par là deviendra clair au fil de la lecture.

00 :07 :47

### 3. Tokénisation

*Avant d'insérer du texte dans un réseau neuronal, nous devons décider comment le représenter.*

La conception des réseaux neuronaux attend en input une séquence unidimensionnelle de symboles. Ils veulent un ensemble fini de symboles possibles. Nous devons donc détermi-

ner quels sont ces symboles, puis représenter nos données sous la forme d'une séquence unidimensionnelle de ces symboles<sup>3</sup>.

Nous disposons actuellement d'une séquence unidimensionnelle de texte (fig. 7). Il s'agit d'une séquence unidimensionnelle, même si, sur mon écran, elle est disposée en deux dimensions. Elle va de gauche à droite et de haut en bas. Mais il s'agit d'une séquence de texte unidimensionnelle.

Comme on travaille sur des ordinateurs, il existe bien sûr une représentation sous-jacente. Si j'utilise le codage UTF-8 [des lettres et autres caractères] pour le texte, je peux alors récupérer les bits bruts correspondant à ce texte dans l'ordinateur. Voici à quoi il ressemble.

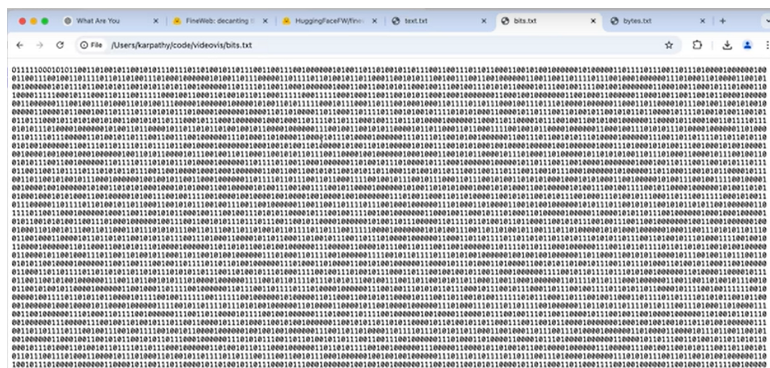


Figure 8 : Texte de la figure 7 encodé en UTF-8. Il s'agit maintenant d'une séquence (beaucoup plus longue) de zéros et de uns.

La première ligne de la figure 7 commence par :

|Viewing Single Post From : Spoilers for the Week of. . .

La toute première barre verticale est encodée en UTF-8. En UTF-8 hexadécimal, c'est 7c. En binaire, cela correspond aux huit premiers bits affichés en haut de la figure 8 : 01111100.

3. Bien sûr, nous pourrions représenter le texte en utilisant seulement les 26 lettres de l'alphabet et quelques signes de ponctuation. Cependant, ce n'est pas ainsi que la grande quantité de texte, comme l'extrait montré figure 7, sera codée pour le traitement par les réseaux neuronaux.



Puis, la lettre V est codée par 56 en hexadécimal et 01010110 en binaire. La lettre i est codée par 69 en hexadécimal et 01101001 en binaire. Ainsi, la ligne supérieure de la figure 8 commence par :

011111000101011001101001...

Que voit-on sur la figure 8 ? C'est la représentation que nous recherchons, en un sens. Nous avons exactement deux symboles possibles : 0 et 1. Et nous avons une très longue séquence. Les 200 pages web correspondent à une très longue séquence, et les 44 téraoctets à une séquence encore beaucoup plus longue.

Lorsque nous sélectionnons un fragment dans notre ensemble de données complet (de 44 téraoctets), il s'agit d'une longue séquence de symboles (à ce stade, de zéros et de uns). Cependant les réseaux neuronaux travaillent mieux sur des séquences pas trop longues. Nous ne voulons pas de séquences extrêmement longues des deux symboles 0 et 1. Il faut trouver un compromis entre le nombre des symboles du vocabulaire, comme nous l'appelons, et la longueur de la séquence résultante. Deux symboles et des séquences extrêmement longues n'est pas idéal. Nous allons utiliser davantage de symboles permettant des séquences plus courtes.

Une façon naïve de compresser, c.-à-d. réduire, la longueur de notre séquence ici, fig. 8, consiste à considérer un groupe de bits consécutifs — par exemple, huit bits — et à les regrouper en un seul, ce qu'on appelle un *octet*.

00 :10 :00

Comme ces bits sont des zéros ou des uns, si nous prenons un groupe de huit bits, il n'y a que 256 ( $= 2^8$ ) combinaisons possibles [de 0 à 255]. Nous pouvons donc représenter cette séquence comme une séquence plus courte d'octets (figure 9).

*Cette séquence d'octets sera huit fois plus courte. Mais nous avons maintenant 256 symboles possibles. Chaque nombre va de 0 à 255.*

## 12 Grands modèles de langage (LLM)

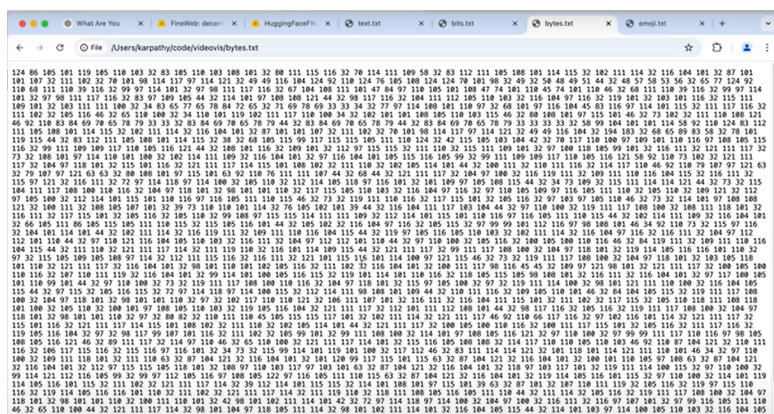


Figure 9 : Le texte de la figure 7 est maintenant représenté avec 256 octets différents (de 0 à 255). Il faut les considérer comme 256 symboles distincts ; on pourrait utiliser 256 émojis pour souligner ce point <sup>4</sup>.

Je vous encourage vivement à les considérer non pas comme des nombres, mais comme des identifiants uniques ou des symboles uniques. Pour illustrer ce point, nous pourrions même remplacer chacun des 256 nombres (de 0 à 255) par un emoji unique. Le lecteur peut imaginer un long texte écrit avec un « alphabet » de 256 émojis. Ce serait tout à fait possible.

En production, pour les modèles de langage volumineux de pointe, il est même souhaitable d'aller plus loin. Il s'agit de continuer à réduire la longueur de la séquence – car, encore une fois, c'est une ressource précieuse – en échange de symboles plus nombreux dans votre vocabulaire.

Pour ce faire, nous exécutons ce qu'on appelle l'algorithme de codage par paires d'octets. Nous recherchons des octets consécutifs ou des symboles très courants.

Par exemple, il s'avère que la séquence 116 suivie de 32 est assez courante et apparaît fréquemment. Nous allons donc

---

4. De même, au Sudoku, les chiffres de 1 à 9 sont utilisés comme neuf symboles distincts, sans aucune signification arithmétique. Pour plus d'informations, voir André Cabannes, *Mathématiques du collège, Volume 1 : 6e et 5e*, Les Éditions du Bec de l'Aigle, 2024, p. 25.

regrouper cette paire dans un nouveau symbole. Nous allons créer un symbole avec l'ID 256<sup>5</sup> et nous allons réécrire chaque paire 116-32 avec ce nouveau symbole. Nous pouvons ensuite itérer cet algorithme autant de fois que nous le souhaitons. Chaque fois que nous créons un nouveau symbole, nous réduisons la longueur et augmentons la taille de la collection de symboles.

*En pratique, un bon réglage pour la taille du vocabulaire de base est d'environ 100 000 symboles possibles.*

GPT-4 utilise un "vocabulaire" de 100 277 symboles.

Ce processus de conversion du texte brut en symboles, ou comme nous les appelons *tokens* (ou parfois en français *jetons*), est appelé *tokénisation*.

Voyons comment GPT-4 effectue la tokénisation, en convertissant du texte en jetons, ou des jetons en texte, et à quoi cela ressemble concrètement. Un site web que j'aime bien utiliser pour explorer ces représentations avec des jetons s'appelle Tiktokenizer (<https://www.tiktokenizer.app/>).

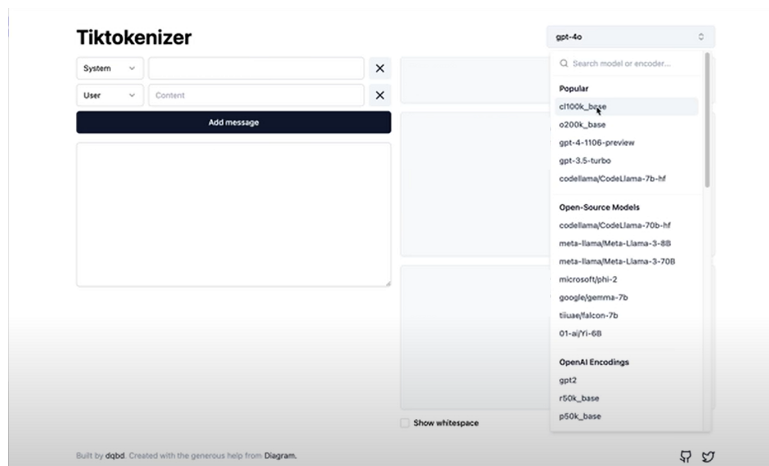


Figure 10 : Tiktokenizer. Sélection du mode d'encodage à droite.

5. Parce que c'est la première ID suivante disponible.

Accédez au menu déroulant et sélectionnez CL100K\_base, qui correspond au générateur de jetons du modèle de base GPT-4. À gauche, vous pouvez saisir du texte, et l'outil vous montre comment il est transformé. Par exemple, si vous saisissez «hello world», le texte sera divisé en deux jetons : le jeton «hello», dont l'ID est 15339, et le jeton « world» (espace comprise), dont l'ID est 1917 (figure 11).

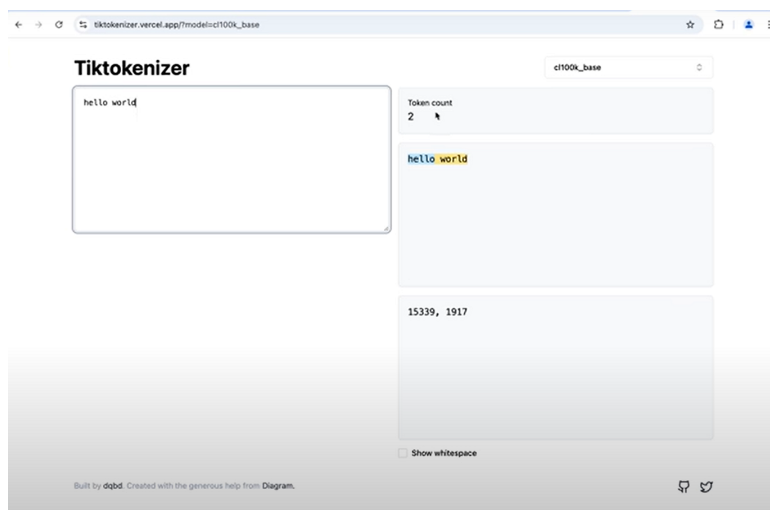


Figure 11 : Encodage de "hello world" avec CL100K\_base.

Si je joins les deux mots et que j'écris «helloworld», j'obtiens à nouveau deux tokens, mais il s'agit du token «h» (ID 71) suivi de «elloworld» sans le «h» (ID 96392). Si j'insère deux espaces entre «hello» et «world», la tokénisation est encore différente ; il y a un nouveau token, 220. La séquence de tokens devient 15339, 220, 1917<sup>6</sup>.

Vous pouvez expérimenter vous-même et voir ce qui se passe. Notez que la casse est respectée : si c'est un «H» majuscule,

---

6. Attention : 220 est le token pour une seule espace, et 1917, comme nous l'avons vu, est le token pour « world» (avec une espace devant).

vous obtiendrez autre chose. Ou, si c'est «HELLO WORLD», cela correspond aux trois jetons 51812, 1623 et 51991, correspondant à «HEL», «LO» et « WORLD». Vous pouvez essayer et comprendre intuitivement le fonctionnement des jetons et de la tokénisation.

Nous reviendrons sur la tokénisation plus loin dans la vidéo. Pour l'instant, je voulais simplement vous montrer le site web <https://www.tiktokenizer.app/>, et ce que devient le texte saisi dans le champ de gauche (fig. 11) une fois traité.

Par exemple, si je prends une ligne de la fig.7, voici comment GPT-4 la verra.

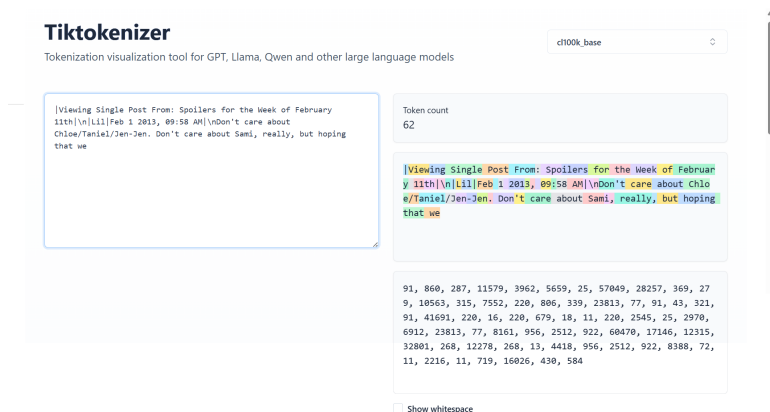


Figure 12 : Tokénisation de la première ligne de texte fig. 7.

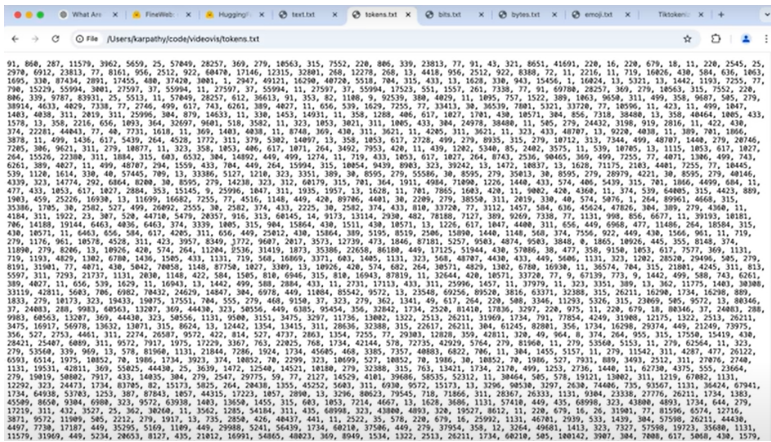
Le texte deviendra une séquence de jetons de longueur 62. L'original est saisi à gauche. En haut à droite, vous pouvez voir comment il est divisé en tokens. En bas à droite, la séquence de nombres (IDs) correspondante s'affiche (nous vous encourageons toutefois à les considérer comme des symboles purs).

Nous disposons maintenant d'une séquence unidimensionnelle de ces symboles<sup>7</sup>.

7. Le dernier jeton est différent de la vidéo, car A. Karpathy a coupé le texte à «that w», alors que nous le coupons à «that we».

00 :14 :30

Nous avons pris la séquence de texte présente dans l'ensemble de données, au début de la figure 7, page 9, et l'avons re-représentée, à l'aide de notre tokenizer, comme une séquence de tokens. Voici à quoi ressemble l'intégralité du texte de la figure 7.

Figure 13 : Tokenisation du texte entier montré fig. 7<sup>8</sup>.

Si l'on revient à l'ensemble de données Fineweb, on constate qu'il ne s'agit pas seulement de 44 téraoctets d'espace disque, mais aussi d'une séquence d'environ 15 000 milliards de jetons.

La figure 13 ne montre que quelques-uns des premiers jetons – peut-être un, deux, trois, voire quelques milliers de jetons – mais il y en a 15 000 milliards.

N'oubliez pas non plus que tous ces éléments représentent de petits fragments de texte ; ils sont comme des atomes dans ces séquences. Les nombres en eux-mêmes ne signifient rien ; ce sont simplement des identifiants uniques.

8. Ce n'est pas la même chose que la figure 9, car sur la fig. 9 nous avons simplement transformé la fig. 8 avec 256 symboles, alors que maintenant nous avons tokenisé le texte original en utilisant 100 277 symboles.

00 :15 :15

#### 4. Réseau neuronal, Input/Output

*Nous arrivons à la partie intéressante : l'entraînement du réseau de neurones. C'est là que se déroule l'essentiel du travail de calcul, lors de l'entraînement des réseaux de neurones.*

Dans cette étape, nous *modélisons* les relations statistiques entre les jetons qui se succèdent dans la séquence<sup>9</sup>.

Nous analysons les données et sélectionnons des fenêtres de jetons – nous sélectionnons une fenêtre de jetons à partir de ces données, de manière assez aléatoire – et la longueur de la fenêtre peut varier de zéro jeton jusqu'à une taille maximale que nous définissons.

En pratique, ça peut être une fenêtre de 8 000 jetons.

En principe, nous pouvons utiliser des fenêtres de jetons de longueur arbitraire, mais le traitement de séquences de fenêtres très longues serait extrêmement coûteux en calcul. Par conséquent, nous décidons simplement que 8 000 est un bon nombre, ou 4 000 ou 16 000.

Dans l'exemple ci-dessous, pour simplifier, je vais prendre les quatre premiers jetons pour illustrer la technique. Il s'agit des quatre premiers jetons de notre texte (fig. 7). Ils ont les identifiants 91, 860, 287 et 11579.

La correspondance est :

- 91 = "|", c'est une barre verticale
- 860 = "View"
- 287 = "ing"
- 11579 = " Single", c.-à-d. une espace suivie de Single

---

9. Nous allons modéliser la relation entre un jeton donné et l'ensemble des jetons qui le précèdent. En substance, cela ressemble beaucoup à une régression statistique où nous déterminons la meilleure formule reliant quelques variables indépendantes  $x_1, x_2, \dots, x_p$  et une variable dépendante  $y$ , lorsque nous disposons d'un grand ensemble de paires observées  $(X_n, y_n)$ .

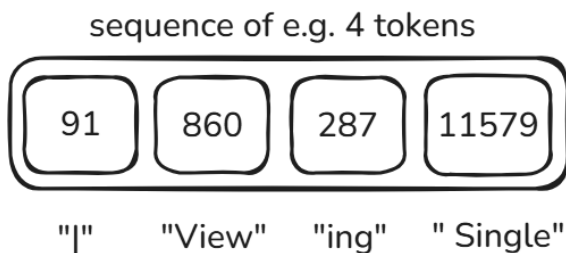


Figure 14 : Séquence de quatre jetons. Nous souhaitons prédire le jeton suivant. Bien sûr, nous savons ici de quel jeton il s'agit (il s'agit du jeton 3962, voir fig. 12 et 13) ; cela nous aidera à trouver une fonction prédictive fiable.

Notre objectif est de construire une fonction qui prédit systématiquement le jeton suivant lorsqu'on lui donne en entrée une séquence de jetons précédents.

Bien sûr, ici, nous savons que le jeton suivant est 3962. Par conséquent, une fois que nous avons trouvé une fonction performante sur l'ensemble des données pendant l'entraînement, elle devrait prédire 3962 lorsqu'on lui donne en entrée les quatre jetons précédents (91, 860, 287 et 11579).

Ces quatre jetons sont appelés le *contexte*. Nous les introduirons dans notre réseau de neurones. Ils constitueront l'entrée (= input), et nous nous attendons à ce que la sortie (= output), une fois notre réseau de neurones correctement entraîné, soit le jeton correct que nous connaissons déjà.

Je décrirai brièvement les entrailles du réseau de neurones. Pour l'instant, il est important de comprendre les entrées et les sorties du réseau neuronal : l'entrée est constituée de séquences de jetons de longueur variable, comprise entre zéro et une taille maximale, par exemple 8000 ; la sortie est une prédiction du jeton suivant. Mais attention c'est une prédiction probabiliste.

Étant donné que notre vocabulaire comporte 100 277 jetons possibles, le réseau neuronal va générer un vecteur de 100 277



probabilités (dont la somme fait 1), la composante  $p_i$  de ce vecteur sera la probabilité que le jeton dont l'ID est  $i$  soit le suivant dans la séquence.

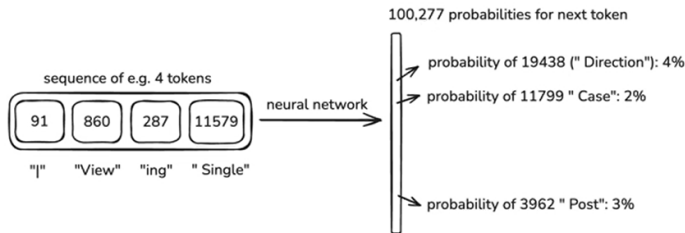


Figure 15 : Chacun des 100 277 tokens a une certaine probabilité d'être le suivant dans la séquence.

Une fois le réseau neuronal correctement entraîné, généralement après plusieurs mois de calculs intensifs, il génère, quel que soit le contexte d'entrée, une probabilité pour chacun des 100 277 jetons qu'il soit le jeton suivant. Il génère ainsi un vecteur de 100 277 nombres positifs dont la somme est égale à 1 (figure 15).

Au début de son apprentissage, le réseau neuronal est initialisé aléatoirement. Nous verrons plus loin ce que cela signifie, bien que cela nous concerne peu. Il s'agit de la construction interne d'un réseau neuronal dont je parlerai seulement un peu.

La figure 15 montre trois exemples de composantes du vecteur de probabilités pour le cinquième jeton. Par exemple, la probabilité que le jeton suivant soit « Direction » est de 4% ; La probabilité qu'il s'agisse du jeton 11799 (« Cas ») est de 2% ; et la probabilité de 3962 (qui correspond à « Post ») est de 3%.

En fait, nous connaissons donc déjà la suite. Nous savons que le jeton suivant a l'ID 3962 (correspondant à « Post »).

Si l'on se tourne vers la construction du réseau neuronal, nous disposons désormais d'un processus mathématique – durant la construction du réseau – pour le mettre à jour, c'est-à-dire le *régler*. Nous utilisons l'ensemble de données et entraî-

nous le réseau neuronal sur un grand nombre d'essais où nous savons précisément ce qu'il doit deviner, afin de le rendre de plus en plus juste dans ses prédictions.

Nous entrerons un peu dans les détails, mais en résumé, nous souhaitons que cette probabilité de 3% (correspondant au jeton 3962) soit plus élevée, et que les probabilités des autres jetons soient plus faibles.

*Nous disposons d'un moyen de calculer mathématiquement comment ajuster et mettre à jour le réseau neuronal — après chaque essai ou série d'essais — afin que la bonne réponse ait une probabilité légèrement supérieure*<sup>10</sup>.

En d'autres termes, si je mets à jour le réseau neuronal maintenant, en me basant sur le fait que «|», «View», «ing», «Single» doit conduire à deviner «Post», la prochaine fois que j'alimenterai le réseau neuronal avec cette séquence de quatre jetons, il sera légèrement ajusté et affichera : «Post» vaut peut-être 4%, «Cas» vaut peut-être 1%, et «Direction» pourrait atteindre 2%, ou quelque chose comme ça.

Nous disposons ainsi d'un moyen de modifier légèrement le réseau neuronal pour augmenter la probabilité d'apparition du jeton suivant correct dans la séquence.

N'oubliez pas que ce processus ne se produit pas uniquement pour le cinquième jeton de la figure 15, les quatre étant introduits en input dans le réseau pour qu'en output il prédise le suivant. Il s'applique à tous les jetons de l'ensemble de données : en sélectionnant un contexte et en essayant de deviner le jeton suivant, tout cela simultanément.

En pratique, nous échantillonnons des petites fenêtres – des petits lots de fenêtres – et, à chaque jeton, nous souhaitons ajuster notre réseau neuronal afin que la probabilité d'obtenir

---

10. On peut considérer le réseau neuronal comme une boîte noire qui reçoit des entrées et produit des sorties. Pour garantir des sorties satisfaisantes, nous devons «ajuster» de manière itérative divers éléments internes, appelés paramètres. Ce processus d'ajustement implique un grand nombre d'essais. Nous n'en donnerons qu'une esquisse mathématique, car ce n'est pas le thème de la vidéo. Et pour donner de explications sérieuses (et non du "Scientific American") il faudrait une autre vidéo de trois heures.

le bon jeton augmente légèrement. Tout cela se déroule en parallèle, par lots importants de ces jetons.

Il s'agit du processus d'entraînement du réseau neuronal : il s'agit d'une séquence de mises à jour pour que ses prédictions correspondent aux statistiques de ce qui se passe réellement dans votre ensemble d'entraînement. Ses probabilités deviennent alors cohérentes avec les schémas statistiques de la succession de ces jetons dans les données.

00 :20 :11

### 5. Entrailles d'un réseau neuronal

Examinons brièvement le fonctionnement interne de ces réseaux de neurones, juste pour avoir une idée de ce qu'ils contiennent.

Comme je l'ai dit, nos entrées sont des séquences de jetons (appelées contexte ou préfix) : dans l'exemple, il s'agit de quatre jetons d'entrée, mais le nombre pourrait varier de zéro à, disons, 8 000 jetons.

En principe, cela peut représenter un nombre quelconque de jetons, mais ça peut devenir très coûteux. Nous le limitons à une certaine longueur maximale pour le modèle. Ces entrées  $x_i$  sont mélangées dans une expression mathématique géante avec les paramètres ou les *poids* du réseau de neurones.

La figure 16 présente un exemple avec quatre jetons et six paramètres, ainsi que leurs réglages. En pratique, les réseaux de neurones modernes ont des centaines de milliers de jetons possibles et des milliards de paramètres.

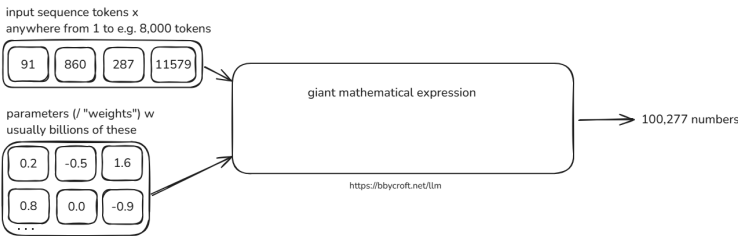


Figure 16 : Tokens d'entrée, paramètres, et formule mathématique.

Au début de la construction du réseau neuronal, les paramètres sont choisis de manière totalement aléatoire. Avec un réglage aléatoire des paramètres, on peut s'attendre à ce que le réseau neuronal fasse des prédictions aléatoires, et c'est le cas : au début, il fait des prédictions totalement aléatoires.

*Mais grâce au processus de mise à jour itérative du réseau – que nous appelons l'entraînement d'un réseau neuronal – les paramètres sont progressivement ajustés de manière à ce que les sorties de notre réseau neuronal soient cohérentes avec les données observées dans notre ensemble d'entraînement.*

On peut comparer ces paramètres aux boutons d'un DJ ou aux faders d'une console de mixage. En ajustant ces boutons ou faders, on obtient des prédictions différentes pour chaque séquence de jetons d'entrée possible.

*Entraîner un réseau neuronal revient simplement à découvrir un réglage de paramètres qui semble cohérent avec les statistiques de l'ensemble d'entraînement<sup>11</sup>.*

00 :22 :00

Voyons un exemple de cette expression mathématique géante, juste pour nous faire une idée. Les réseaux modernes utilisent des formules massives, contenant des milliards de termes. Mais limitons-nous à un exemple simple.

---

11. Plus mathématiquement parlant :

1. Nous disposons d'un grand nombre de suites  $S_i$  pour lesquelles nous connaissons le jeton suivant  $y_i$ .
2. Chaque suite  $S_i$  est une collection de  $x_{ij}$  (par exemple, 8000  $x_{ij}$ ).
3. Nous souhaitons trouver une fonction  $f$ , facile à calculer, qui, pour toute suite d'entrée  $S$  produit – ou affiche comme *probable* – une sortie  $y$  correcte pour toutes les paires connues  $(S_i, y_i)$ .
4. La fonction  $f$  sera une combinaison des jetons de la suite avec des poids, qui serviront de paramètres à la fonction.
5. Nous ajusterons ces poids afin de satisfaire au mieux la condition du point 3.
6. Cela se fait à l'aide de techniques de minimisation mathématique, dont la plus connue est la descente de gradient le long d'une fonction de coût (hautement multidimensionnelle).

Cela ressemblerait à ceci (fig. 17). C'est le genre d'expression ou formule – juste pour vous montrer que ce n'est pas si effrayant que cela.

Nous avons des entrées  $x$  comme  $x_1$   $x_2$  – dans cet exemple, deux entrées  $x_i$  – et elles sont combinées avec les poids du réseau  $w_0, w_1, w_2, w_3, \dots$  ici jusqu'à  $w_{12}$ .

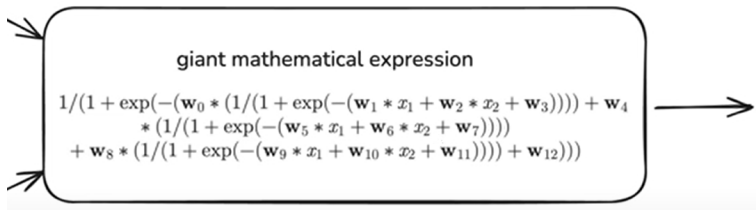


Figure 17 : Exemple d'expression mathématique<sup>12</sup>.

La combinaison des  $x_i$  et des poids consiste en des opérations simples comme la multiplication, l'addition, l'exponentiation, la division, etc.

Les recherches en architecture de réseaux neuronaux visent à concevoir des expressions mathématiques efficaces présentant de nombreuses caractéristiques pratiques : faciles à exprimer, optimisables, parallélisables, etc.

*En fin de compte, ces expressions ne sont pas complexes. Elles combinent les entrées et les paramètres (= poids) pour produire des prédictions. Nous optimisons les paramètres du réseau neuronal afin que les prédictions soient cohérentes avec l'ensemble des données d'entraînement.*

Je voudrais maintenant montrer un exemple concret de production de ces réseaux neuronaux. Pour cela, je vous encourage à consulter ce site web qui propose une très belle visualisation de l'un de ces réseaux : <https://nano-gpt.com/conversation/new> (et aussi <https://bbbycroft.net/llm>).

12. Une fois le réseau entraîné, tous les paramètres sont fixes. Bien que le nombre exact soit confidentiel, des sources fiables estiment que ChatGPT-4 possède environ 1,8 billion de paramètres.

Voici ce que vous trouverez sur le site web de NanoGPT :

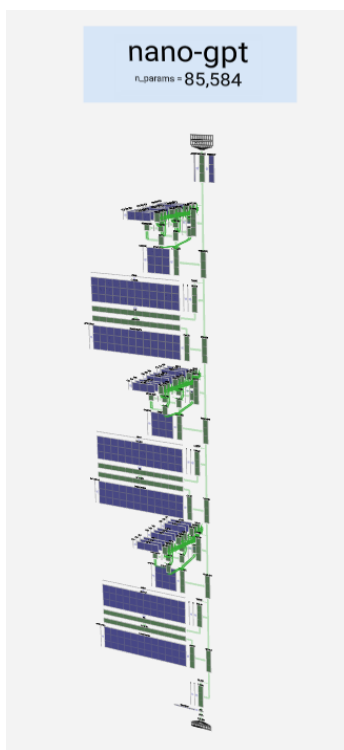


Figure 18 : NanoGPT.

Ce réseau neuronal, utilisé commercialement, possède une structure particulière. Il appartient à la classe des réseaux appelés *transformateurs*<sup>13</sup>. Celui-ci, à titre d'exemple, comporte environ 85 000 paramètres.

Nous nous amusons à représenter le NanoGPT comme un matériel, mais il s'agit d'un pur logiciel. En haut, nous commençons par l'injection d'une séquence de jetons.

---

13. L'auteur appelle systématiquement le NanoGPT un « Transformateur » (ou « GPT » pour Generative Pretrained Transformer), d'après l'article « Attention is All You Need » et l'architecture neuronale standard utilisée par les modèles GPT-2 et GPT-3.

L'information descend à travers le réseau neuronal, couche par couche, jusqu'à la sortie. À la couche de sortie, nous obtenons des valeurs appelées logits. En appliquant la fonction softmax à ces logits, nous les convertissons en probabilités.

Elles représentent les prédictions du modèle pour le jeton suivant ; elles nous indiquent quel jeton est le plus susceptible d'apparaître ensuite dans la séquence.

Au centre du transformateur NanoGPT (fig. 18), se trouve une séquence de transformations. Toutes les valeurs intermédiaires produites par ce « calculateur » contribuent à prédire ce qui suit. Les jetons sont d'abord intégrés dans ce qu'on appelle une *représentation distribuée*. Autrement dit, chaque jeton possible est représenté dans le réseau neuronal par un vecteur.

À l'entrée, nous intégrons les jetons, et leurs valeurs correspondantes sont propagées à travers le diagramme. Dans NanoGPT, nous voyons trois blocs transformateurs. Chaque élément de chacun de ces blocs effectue une opération mathématique simple, telle que la normalisation des couches, la multiplication matricielle ou le softmax.

Dans chaque bloc transformateur, l'information passe dans le mécanisme d'attention, puis continue vers le bloc perceptron multicouche, et ainsi de suite<sup>14</sup>. Tous les nombres produits dans ces blocs sont des valeurs intermédiaires dans le calcul global.

Ces valeurs peuvent être comparées à des décharges des neurones artificiels. Cependant, l'analogie ne doit pas être poussée trop loin, car ces neurones artificiels sont beaucoup plus simples que les neurones biologiques. Ces derniers sont des systèmes dynamiques très complexes, capables de mémoriser et de s'adapter à de nouvelles informations.

En revanche, dans l'expression de la figure 17, il n'y a pas de notion de mémoire ; il s'agit simplement d'une fonction ma-

---

14. Nous donnons ici des explications type "vulgarisation" que le lecteur ou la lectrice intéressés devront impérativement remplacer par un ouvrage ou une vidéo spécialisés.

thématique fixe reliant l'entrée à la sortie de manière statique et prévisible.

Bien qu'il s'agisse de neurones très simples comparés à leurs homologues biologiques, on peut néanmoins les considérer comme formant un tissu cérébral synthétique, si cette analogie peut vous être utile.

En fin de compte, l'information circule à travers ces couches de neurones artificiels, étape par étape, jusqu'à ce que le modèle produise ses prédictions finales.

Je n'ai pas l'intention de m'étendre davantage sur les détails mathématiques précis de ces transformations. Ce n'est pas l'objectif de la vidéo. Et honnêtement, je ne pense pas qu'il soit très utile de connaître en détail les mathématiques complexes des réseaux de neurones pour comprendre les LLM<sup>15</sup>.

*Ce qu'il est important de comprendre, c'est qu'il s'agit d'une fonction mathématique. Elle est paramétrée par un ensemble fixe de paramètres, par exemple 85 000. C'est un moyen de transformer des entrées en sorties. En ajustant les paramètres, nous obtenons différents types de prédictions. Nous devons trouver un bon réglage pour ces paramètres afin que les prédictions correspondent à ce qui est observé dans l'ensemble d'entraînement*<sup>16</sup>.

Voilà ce qu'est un transformeur.

---

15. La chaîne YouTube 3Blue1Brown de Grant Sanderson donne un exemple assez simple à suivre : <https://www.youtube.com/watch?v=aircAruvnKk>

Il explique comment un réseau de neurones est utilisé pour lire des chiffres manuscrits. Il s'agit essentiellement du système développé par Yann LeCun (né en 1960) avec son réseau de neurones pionnier pour la reconnaissance des chiffres manuscrits — le premier réseau neuronal convolutif pratique (CNN) — en 1989, alors qu'il travaillait chez AT&T Bell Laboratories. Parmi ses collaborateurs figuraient Léon Bottou, Yoshua Bengio, puis Patrick Haffner.

Grant Sanderson, de 3Blue1Brown, s'appuie encore ici et là sur la vulgarisation plutôt que sur des calculs clairs, mais il complète utilement la présentation d'Andrej Karpathy.

16. Nous utilisons pour cela une technique mathématique appelée *optimisation* : nous minimisons une fonction de coût par descente de gradient. Un site web recommandé par Perplexity pour comprendre les mathématiques est <https://devot.team/blog/how-to-make-an-ai>.



00 :26 :00

## 6. Inférence

J'ai décrit le fonctionnement interne du réseau neuronal et expliqué le processus d'apprentissage. Je souhaite maintenant aborder une autre étape importante de l'utilisation de ces réseaux : l'inférence.

L'inférence permet de générer de nouvelles données à partir du modèle. Pour ce faire, nous cherchons à identifier les schémas que le modèle a internalisés dans les paramètres de son réseau.

La génération à partir du modèle est relativement simple. Nous commençons par quelques jetons qui constituent le préfixe de départ. Supposons que nous souhaitions commencer par le jeton 91. Nous l'introduisons dans le réseau. Rappelons que le réseau produit des probabilités, fig. 19 :

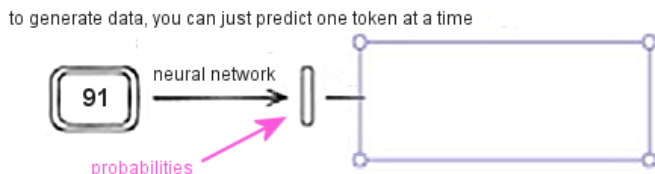


Figure 19 : Vecteur de probabilités (du token suivant) étant donné qu'on a le token 91.

Ce que nous pouvons faire maintenant, c'est lancer une pièce biaisée<sup>17</sup>. Nous pouvons échantillonner (= générer aléatoirement) un jeton basé sur cette distribution de probabilités. Les jetons auxquels le modèle attribue une probabilité élevée sont plus susceptibles d'être échantillonnés lors du lancer de cette pièce biaisée. On peut l'envisager de cette façon.

17. Par « pièce biaisée », l'auteur n'entend pas une variable aléatoire avec deux résultats possibles. Imaginez une Roue de la Fortune avec 100 277 résultats possibles, dont les probabilités sont les composantes du vecteur produit en output par le réseau de neurones.

Nous utilisons donc la distribution pour choisir un seul jeton. Par exemple, le jeton 860 vient ensuite.

Bien que 860 soit un jeton relativement probable – étant donné que 91 le précède – il n'est sans doute pas le seul jeton possible. D'autres jetons auraient pu être choisis à l'aide du vecteur de probabilités montré fig. 19. Cependant, nous constatons que 860 est un jeton relativement probable. Et en effet, dans notre ensemble de données d'apprentissage (figure 13), 860 suit 91.

Continuons. Nous avons vu qu'après 91, le modèle nous a donné 860. Nous l'ajoutons. Et nous nous demandons à nouveau : quel est le jeton suivant, le troisième ? Nous obtenons 287. Cette ID est toujours cohérente avec les données.

Échantillonnons à nouveau. Nous partons maintenant de la séquence de trois jetons (91, 860, 287). Le modèle produit un vecteur de probabilités pour le quatrième jeton. Nous échantillonnons à partir de ce vecteur et supposons que nous obtenons 11579 (fig. 20).

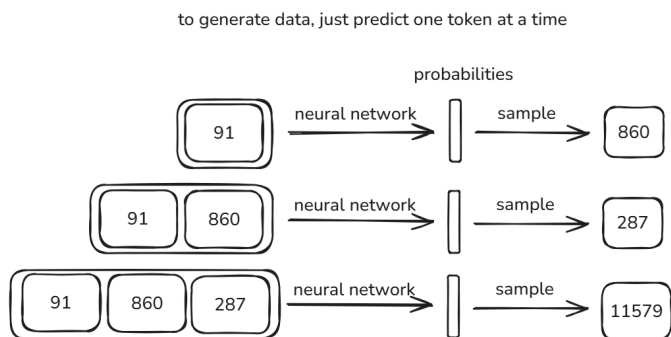


Figure 20 : À partir de 91, on a produit 860, puis 287, puis 11579.

Recommençons une fois de plus pour obtenir le cinquième jeton. Prenons ces quatre jetons (91, 860, 287, 11 579). Demandons au modèle quel est le vecteur de probabilités pour le cinquième jeton. Nous échantillonnons et obtenons, par exemple, 13659.

Il ne s'agit pas de 3962, c'est-à-dire du cinquième jeton de notre ensemble de données d'entraînement. Ce jeton 13659 est le jeton « Article ». Notre processus génératif, à partir de 91 (la barre verticale), a produit

|Viewing Single Article

Nous n'avons donc pas reproduit exactement la séquence observée dans les données d'entraînement, qui était

|Viewing Single Post

Revenez à la figure 12, page 15, si nécessaire, pour vérifier la tokénisation de l'ensemble de données d'entraînement.

*Gardez à l'esprit que ces systèmes sont stochastiques.*

Nous choisissons le jeton suivant à l'aide d'une procédure aléatoire – ce que j'appelle « tirer à pile ou face ». Parfois, par chance, nous reproduisons exactement le petit fragment de texte suivant de l'ensemble d'entraînement. Mais parfois, nous obtenons un jeton qui n'est pas celui des données d'entraînement.

Nous obtiendrons donc des modifications par rapport aux données d'entraînement, car à chaque génération, nous pouvons obtenir un jeton légèrement différent.

Une fois ce nouveau jeton intégré, si vous échantillonnez le suivant, et ainsi de suite, vous commencerez très rapidement à générer des flux de jetons très différents de ceux des documents d'entraînement.

Statistiquement, leurs propriétés seront similaires, mais elles ne sont pas identiques à vos données d'entraînement. Elles sont « inspirées » par ces données. Ainsi, dans l'exemple que nous avons construit, nous avons obtenu une séquence légèrement différente. Pourquoi avons-nous obtenu « Article » ? Sans doute qu'« Article » est un jeton suivant relativement probable quand on part du contexte

|Viewing Single



Figure 21 : Construction d'un texte un token à la fois. Chaque nouveau token est l'un des plus probables étant donné les précédents, qui forment le contexte pour produire un vecteur de probabilités pour le nouveau token, et nous le tirons au hasard <sup>18</sup>.

Vous pouvez imaginer que le mot « Article » suivait cette fenêtre de contexte quelque part dans les documents d'entraînement. Et il se trouve que nous l'avons tiré ici.

L'inférence consiste à prédire à partir de ces contextes, un jeton à la fois, en utilisant à chaque fois la distribution de probabilité (calculée par le réseau neuronal) du jeton suivant, étant donné le contexte. Nous tirons constamment un jeton suivant aléatoire. Et selon notre chance ou notre malchance, nous pouvons obtenir des modèles très différents.

Dans la plupart des scénarios de construction de LLM, le téléchargement et la tokénisation d'Internet constituent une étape de pré-traitement unique.

Une fois votre immense séquence de jetons obtenue, vous pouvez commencer à entraîner les réseaux (voir fig. 13, p. 16, pour le début de l'ensemble de données d'entraînement).

En pratique, vous tenterez d'entraîner de nombreux réseaux différents, avec des configurations, des agencements et des tailles variés. Vous effectuerez donc un entraînement intensif des réseaux neuronaux.

---

18. Attention, nous ne choisissons pas systématiquement le token le plus probable. Nous le choisissons au hasard, donc nous choisissons généralement l'un des token les plus probables.

*Une fois que vous disposez d'un réseau neuronal, que vous l'avez entraîné et que vous disposez d'un ensemble de paramètres spécifiques qui vous convient, vous pouvez utiliser le modèle et effectuer des inférences<sup>19</sup>. Et vous pouvez réellement générer des données à partir du modèle.*

Lorsque vous êtes sur ChatGPT et que vous discutez avec le modèle, ce dernier a été entraîné par OpenAI il y a plusieurs mois. Ils disposent d'un ensemble de poids spécifiques qui fonctionnent bien. Il ne fait désormais que des inférences. Il n'y a plus d'entraînement pour ajuster les paramètres.

Vous lui fournissez un texte, qui est converti en une séquence de jetons appelée le contexte. Lorsque le LLM vous répond, en pratique il complète des séquences de jetons. C'est ce que vous voyez généré lorsque vous utilisez ChatGPT.

00 :31 :00

## 7. GPT-2 : entraînement et inférence

Prenons un exemple concret d'entraînement et d'inférence qui vous donnera une idée de ce à quoi cela ressemble une fois ces modèles entraînés.

L'exemple que je vais prendre, que j'affectionne particulièrement, est celui d'OpenAI GPT-2<sup>20</sup>.

---

19. Notez que l'inférence est déjà utilisée pendant le processus d'entraînement, où son rôle est de participer à la construction d'un ensemble de paramètres efficace. Mais, plus important encore, l'inférence est le mécanisme clé lorsque nous mettons en œuvre un LLM pour générer du nouveau texte, par exemple pour produire des réponses à des questions ou pour produire d'autres résultats en situation réelle.

20. Andrej Karpathy (né en 1986) est un informaticien slovaque-canadien. Il a été membre fondateur et chercheur chez OpenAI de 2015 à 2017. Il a ensuite rejoint Tesla. Il a quitté Tesla début 2023 pour retourner chez OpenAI, avant de le quitter à nouveau en février 2024.

Au milieu de 2025, Andrej Karpathy était le fondateur d'Eureka Labs, une entreprise de formation en IA axée sur l'enseignement et la diffusion des principes et des pratiques de l'IA. Il développe des cours en ligne (comme LLM101n), crée du contenu vidéo éducatif sur les grands modèles de langage et promeut les assistants pédagogiques IA et la maîtrise pratique de l'apprentissage automatique.

GPT signifie *Generative Pre-trained Transformer* et GPT-2 est la deuxième version de la série GPT d'OpenAI.

Lorsque vous discutez avec ChatGPT aujourd'hui, le modèle qui sous-tend toute la magie de votre interaction est la quatrième version de la série, appelée GPT-4.

GPT-2 a été publié en février 2019 par OpenAI dans l'article « Language Models are Unsupervised Multitask Learners » par Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei et Ilya Sutskever.

---

### Language Models are Unsupervised Multitask Learners

---

Alec Radford <sup>\*1</sup> Jeffrey Wu <sup>\*1</sup> Rewon Child <sup>1</sup> David Luan <sup>1</sup> Dario Amodei <sup>\*\*1</sup> Ilya Sutskever <sup>\*\*1</sup>

#### Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the

competent generalists. We would like to move towards more general systems which can perform many tasks - eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks

Figure 22 : [https://cdn.openai.com/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf)

La raison pour laquelle j'apprécie GPT-2 est que c'est la première fois qu'une « stack » technologiquement moderne a été proposée. GPT-2 contient tous les éléments d'un LLM moderne. Les LLM plus récents ont simplement grandi en taille par rapport à GPT-2.

Je ne vais évidemment pas présenter tout l'article, car il s'agit d'une publication technique.

Voici cependant quelques aspects que je voudrais souligner :

- GPT-2 était un réseau neuronal transformeur, comme les réseaux neuronaux actuels.
- Il avait 1,5 milliard de paramètres. Ce sont les paramètres dont nous avons parlé (fig. 17). Aujourd'hui, les transformeurs modernes en ont près de mille milliards, en tout cas plusieurs centaines de milliards.
- La longueur maximale du contexte était de 1024 jetons. Ainsi, lorsque nous extrayons des fragments de fenêtres de jetons de l'ensemble de données, nous ne prenons jamais plus de 1024 jetons. Lorsque vous essayez de prédire le jeton suivant d'une séquence, vous n'aurez jamais plus de 1024 jetons dans votre contexte pour effectuer cette prédiction.

C'est aussi très réduit par rapport à ce qui se fait de nos jours. Aujourd'hui la longueur du contexte serait plus proche de quelques centaines de milliers, voire d'un million. Vous disposez de beaucoup plus de contexte et de jetons dans l'historique, ce qui vous permet de bien mieux prédire le prochain jeton de la séquence.

- GPT-2 a été entraîné sur environ 100 milliards de jetons. C'est également relativement peu par rapport aux normes modernes. Je l'ai dit, l'ensemble de données Fineweb que nous avons vu contient 15 000 milliards de jetons. Cent milliards est relativement peu.

J'ai essayé de reproduire GPT-2 pour m'amuser, dans le cadre du projet llm.c. Vous pouvez lire l'article sur GitHub, dans le dépôt llm.c<sup>21</sup>.

En particulier, le coût d'entraînement de GPT-2 en 2019 était estimé à environ 40 000 \$, mais aujourd'hui, on peut faire bien mieux. Par exemple, lors de mon projet, cela a pris environ une journée et coûté environ 600 \$, sans même optimiser les coûts. Je pense qu'on pourrait probablement ramener ce coût à environ 100 \$ aujourd'hui.

---

21. <https://github.com/karpathy/llm.c/discussions/481>

Titre : « Reproduire GPT-2 (124 Mo) dans llm.c en 90 minutes pour 20 \$ »

Pourquoi les coûts ont-ils autant baissé ? D'abord, les jeux de données d'entraînement sont bien meilleurs maintenant. La façon dont nous les filtrons et les préparons est devenue beaucoup plus sophistiquée. En conséquence, les ensembles de données sont de bien meilleure qualité. C'est la première raison.

Mais la principale différence réside dans la puissance des matériels informatiques – nous y reviendrons dans un instant. De même, les logiciels permettant d'exécuter ces modèles et d'optimiser la vitesse du matériel ont été considérablement améliorés. Car tout le monde s'est rué sur ces modèles et s'efforce de les faire tourner très rapidement.

Je ne peux pas détailler ma reproduction de GPT-2 avec NanoGPT. Le document sur GitHub est un long article technique. Mais j'aimerais quand même vous donner une idée de ce à quoi ressemble l'entraînement d'un de ces modèles en tant que chercheur. Qu'observe-t-on ? À quoi cela ressemble-t-il ? Quelles impressions en retire-t-on ?

Voici à quoi cela ressemble.

```

    381/32000  loss 5.518550 (-1.992)  norm 0.6596 (-0.292)  lr 3.27e-04  6744.32 ms  61.2% bf16 MFU  152448 tok/s
    382/32000  loss 5.512454 (-1.932)  norm 0.6225 (-0.492)  lr 3.27e-04  6770.76 ms  60.9% bf16 MFU  152568 tok/s
    383/32000  loss 5.505824 (-1.952)  norm 0.4897 (-1.222)  lr 3.28e-04  6817.14 ms  60.5% bf16 MFU  152622 tok/s
    384/32000  loss 5.504938 (-1.922)  norm 0.6167 (-0.532)  lr 3.29e-04  6838.21 ms  60.3% bf16 MFU  152658 tok/s
    385/32000  loss 5.499972 (-1.932)  norm 0.4792 (-1.252)  lr 3.30e-04  6856.52 ms  60.1% bf16 MFU  152678 tok/s
    386/32000  loss 5.497883 (-1.962)  norm 0.5078 (-1.882)  lr 3.31e-04  6873.37 ms  60.0% bf16 MFU  152664 tok/s
    387/32000  loss 5.442298 (-2.242)  norm 0.5639 (-0.762)  lr 3.32e-04  6884.13 ms  59.9% bf16 MFU  152647 tok/s
    388/32000  loss 5.500896 (-1.912)  norm 0.6175 (-0.462)  lr 3.33e-04  6894.07 ms  59.9% bf16 MFU  152619 tok/s
    389/32000  loss 5.486438 (-1.862)  norm 0.7342 (+0.192)  lr 3.33e-04  6898.52 ms  59.9% bf16 MFU  152588 tok/s
    390/32000  loss 5.581855 (-1.732)  norm 0.6886 (-0.582)  lr 3.34e-04  6902.34 ms  59.9% bf16 MFU  152554 tok/s
    391/32000  loss 5.485813 (-1.912)  norm 0.7188 (+0.122)  lr 3.35e-04  6903.48 ms  59.7% bf16 MFU  152521 tok/s
    392/32000  loss 5.428624 (-2.142)  norm 0.4663 (-1.262)  lr 3.36e-04  6906.22 ms  59.7% bf16 MFU  152487 tok/s
    393/32000  loss 5.506081 (-1.592)  norm 0.6174 (-0.432)  lr 3.37e-04  6905.48 ms  59.7% bf16 MFU  152455 tok/s
    394/32000  loss 5.416741 (-1.942)  norm 0.5577 (-0.792)  lr 3.38e-04  6902.39 ms  59.9% bf16 MFU  152458 tok/s
    395/32000  loss 5.495521 (-1.612)  norm 0.6721 (-0.122)  lr 3.39e-04  6898.85 ms  59.9% bf16 MFU  152486 tok/s
    396/32000  loss 5.541896 (-1.772)  norm 0.8849 (+1.032)  lr 3.39e-04  6905.31 ms  59.7% bf16 MFU  152378 tok/s
    397/32000  loss 5.462929 (-1.942)  norm 0.6762 (+0.972)  lr 3.40e-04  6905.07 ms  59.7% bf16 MFU  152352 tok/s
    398/32000  loss 5.542685 (-1.222)  norm 0.8837 (+0.992)  lr 3.41e-04  6904.35 ms  59.7% bf16 MFU  152328 tok/s
    399/32000  loss 5.506354 (-1.632)  norm 0.5841 (-1.092)  lr 3.42e-04  6905.22 ms  59.7% bf16 MFU  152284 tok/s
    400/32000  loss 5.522236 (-1.312)  norm 0.6659 (-0.742)  lr 3.43e-04  6901.89 ms  59.9% bf16 MFU  152286 tok/s
    generating:

    I obviously something on the inside in order to start myself in the event, everything I ran for a cart that, so it works a culture,
    screamed, I wanted to the show up with odd cycle and yours, igniting everything Ochal paio goreball horror sound,8s backstage with f
    ending blankets planned.
    Context: Voters, Social version 2ndansae were terminated by Capting Bigbrook Rd. You may have teamed up a show in this stand wherev
    ou may never smooth sessions as you forgot about how to see a lot or attack, but I'd had paused at a picture of them.<endofext><Clic
    k here: 38huk courtin/events/martin-so-enfant-toal tyemen and relaxat and 8x.
    podcast: Compa PrepadesIumba. CLICK on Wonderplace-difficultist-Jitzev Ring Market Player Ejova BestES domains
    Akaralaza Vadel Améaravo MVEA/Dargleza Akouat Fedal Island Yu-Oyanyi King Camper Rivier Rdiversatit van Virtual Steel Malettherthayi
    Nehrids
    kit should alsoe quot qus on internationally. Put through fenced area of A

    401/32000  loss 5.462952 (-1.552)  norm 0.5776 (-0.732)  lr 3.44e-04  6745.28 ms  61.1% bf16 MFU  152445 tok/s
    402/32000  loss 5.445932 (-1.772)  norm 0.6173 (-0.582)  lr 3.45e-04  6778.58 ms  60.8% bf16 MFU  152557 tok/s
    403/32000  loss 5.484241 (-1.492)  norm 0.6250 (-0.452)  lr 3.45e-04  6813.61 ms  60.5% bf16 MFU  152625 tok/s
  
```

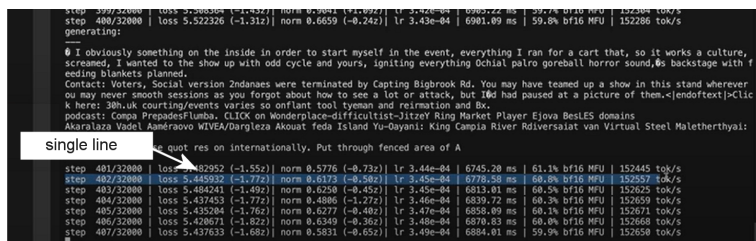
Figure 23 : Entraînement de GPT-2.



J'entraîne un modèle GPT-2 [dans la vidéo, pendant que le speaker parle, on voit les opérations défilier sur l'écran de son ordinateur]. Fig. 23, chaque ligne du tableau représente une mise à jour du modèle.

Rappelez-vous comment nous améliorons la prédiction pour chaque jeton suivant en fonction d'une fenêtre de contexte (voir fig. 16), en mettant à jour les poids ou paramètres du réseau neuronal.

Chaque ligne fig. 24 représente une mise à jour du réseau neuronal, dont nous modifions légèrement les paramètres. Il devient alors plus performant pour prédire le jeton et même toute la séquence suivants.



```

step 399/32000 | loss 5.38038 (-1.432) | norm 0.5881 (-1.022) | lr 3.426e-04 | 6963.22 ms | 59.7% bf16 MFU | 152284 tok/s
step 400/32000 | loss 5.522326 (-1.312) | norm 0.6659 (-0.242) | lr 3.436e-04 | 6981.09 ms | 59.8% bf16 MFU | 152286 tok/s
generating:
---
0 I obviously something on the inside in order to start myself in the event, everything I ran for a cart that, so it works a culture,
screamed, I wanted to the show up with odd cycle and yours, igniting everything Gchial palro gireball horror sound,6s backstage with f
eeding blankets planned.
Contact: Voters, Social version 2ndwaves were terminated by Capting Bigbrook Rd. You may have teamed up a show in this stand wherever
ou may never smooth sessions as you forget about how to see a lot or attack, but I'd had paused at a picture of them.<[endofix]>Clic
k here: 38h.uk courting/events varies so onflant tool tyman and reimation and Bx.
podcast: Compa PrepadesLumba. CLICK on wonderplace-difficultist-Jit2of Ring Market Player Ejoza BestLES domains
Akoralaza Vadel Amerarovo WIVEA/Dangloza Akouat feda Island Yu-Dayan: King Campia River Rdiversariat van Virtual Steel Maletherthyai:
single line
---
step 401/32000 | loss 5.482952 (-1.552) | norm 0.5776 (-0.732) | lr 3.446e-04 | 6745.20 ms | 61.1% bf16 MFU | 152445 tok/s
step 402/32000 | loss 5.489322 (-1.772) | norm 0.6178 (-0.502) | lr 3.456e-04 | 6776.59 ms | 60.8% bf16 MFU | 152657 tok/s
step 403/32000 | loss 5.494241 (-1.692) | norm 0.6258 (-0.452) | lr 3.456e-04 | 6813.61 ms | 60.5% bf16 MFU | 152625 tok/s
step 404/32000 | loss 5.437453 (-1.772) | norm 0.4886 (-1.272) | lr 3.466e-04 | 6839.72 ms | 60.3% bf16 MFU | 152659 tok/s
step 405/32000 | loss 5.435204 (-1.762) | norm 0.6277 (-0.402) | lr 3.476e-04 | 6852.89 ms | 60.1% bf16 MFU | 152671 tok/s
step 406/32000 | loss 5.428671 (-1.522) | norm 0.5349 (-0.362) | lr 3.486e-04 | 6879.83 ms | 60.8% bf16 MFU | 152688 tok/s
step 407/32000 | loss 5.437633 (-1.682) | norm 0.5831 (-0.652) | lr 3.496e-04 | 6884.01 ms | 59.9% bf16 MFU | 152658 tok/s

```

Figure 24 : Mise à jour des paramètres.

À chaque ligne d'entraînement, les paramètres du modèle sont mis à jour afin d'améliorer les prédictions pour un million de jetons de l'ensemble d'entraînement à la fois. Autrement dit, à chaque ligne on a extrait un lot d'un million de jetons de l'ensemble de données et nous travaillons à améliorer la précision du modèle pour prédire le jeton suivant de chaque séquence, et ce, simultanément sur l'ensemble du lot.

Le nombre à surveiller de près est celui appelé *loss* – en français *perte*.

C'est un nombre unique qui mesure la performance actuelle du réseau. Il est construit de sorte que vous voulez qu'il diminue. On voit effectivement que la perte diminue avec la progression de la mises à jour du réseau. En conséquence, les prédictions sur le jeton suivant d'une séquence seront meilleures.

La perte est le chiffre que vous suivez. En attendant, vous vous tournez les pouces, vous buvez un café et vous vous assurez que tout est correct, c'est-à-dire qu'à chaque mise à jour, votre perte s'améliore (diminue) et que donc le réseau améliorera ses prédictions. Vous voyez ici que nous traitons 1 million de jetons par mise à jour. Chaque mise à jour prend environ 7 secondes. Nous allons traiter un total de 32 000 étapes d'optimisation. Ainsi, 32 000 étapes de 1 million de jetons chacune représentent environ 33 milliards de jetons à traiter. Nous n'en sommes actuellement qu'à environ 420 étapes – 420 sur 32000. Nous n'avons donc effectué qu'un peu plus de 1% du travail, car j'ai lancé le processus il y a environ un quart d'heure.

Toutes les 20 étapes, j'ai configuré cette optimisation pour effectuer une inférence.

Ce que vous voyez ici est le modèle prédisant le jeton suivant d'une séquence [regardez la vidéo pour voir les nouveaux jetons apparaître séquentiellement, à partir de 00 :37 :10]. On l'a initié de manière totalement aléatoire. Ensuite, on continue à insérer les jetons. Nous exécutons donc cette étape d'inférence (voir fig. 20). Nous voyons le modèle prédire le jeton suivant de la séquence. Chaque fois qu'un élément apparaît, il s'agit d'un nouveau jeton.

Regardons le texte généré. Comme le montre la fig. 24, il n'est pas encore très cohérent ; rappelons que l'apprentissage n'est qu'à 1%, le modèle n'a donc pas encore appris à prédire le jeton suivant avec précision. Une grande partie du résultat ressemble à du charabia, comme il est normal à ce stade précoce. Cependant, certains fragments commencent à montrer une certaine cohérence locale : "Since she is mine", "it's a part of the information", "should discuss my father", "great companions", "Gordon showed me", "sitting over it", etc.

00 :38 :00

Je sais que cela n'a pas encore l'air très convaincant, mais faisons défiler l'écran vers le haut et voyons à quoi cela ressemblait lorsque nous avons commencé l'optimisation.

Vous voyez que ce que nous obtenons ici (après un seul cycle d'optimisation, c'est-à-dire le premier lot de 20 mises à jour) semble complètement aléatoire.

```
200 ifAnother :# mortar ris partners Navodsulic advers
exped slime negotiate on desper asudging Healthsecond-
sotted, acknowledged lethal-Mode
```

```
{ "Gott", "" inviting Ballistic.
```

```
ominpart, and Tou Australians dispensarieskukuagne801
and%] Wright etnh408 of filed bitterness the humpSpot
Roads thickness, wing aknowledgement342 Glou flame overcl
Summoner quirk sr. seismic. wrecked, cite toAIN Brus Und
leastDecember Moose reinforce Tay in Fishing [here two
Japanese characters] divergence entity friendshipenment%
olds He%the to onesAnother markets superfluMe priced...
```

Bien sûr, cela est dû au fait que le modèle n'avait subi que 20 mises à jour de paramètres à ce moment-là. Il s'agissait alors d'un réseau aléatoire.

Nous venons de voir qu'après seulement 1% du travail d'ajustement des paramètres, le modèle était déjà nettement plus performant qu'après seulement 20 étapes.

Si nous poursuivions l'entraînement sur les 32000 étapes, le modèle s'améliorerait au point de générer un texte anglais relativement correct et cohérent et de produire des séquences de jetons correctes.

Cet entraînement va prendre encore un jour ou deux. Pour l'instant, nous nous assurons simplement que la perte diminue. Tout semble correct (voir les figures 23 et 24 qui sont des instantanés du processus), il ne reste plus qu'à patienter.

Tournons-nous vers l'exécution des calculs. Je n'exécute pas cette optimisation sur mon ordinateur portable. Ce serait beaucoup trop coûteux [en temps]. Nous devons faire fonctionner ce réseau neuronal, l'améliorer, et nous avons besoin de toutes ces données, etc. L'exécution sur un ordinateur portable est impossible, car la tâche simplement trop grande.

La fabrication de notre LLM (c'est-à-dire la création par ajustements successifs d'une bonne collection de paramètres – environ 85 000) s'exécute sur un ordinateur dans le *cloud*.

Je souhaite aborder l'aspect calcul de l'entraînement de ces modèles et comment c'est réalisé concrètement.

L'ordinateur que j'utilise est un 8x H100 node<sup>22</sup> — cela signifie qu'il y a huit GPU H100 sur une seule machine. Je loue cet ordinateur quelque part dans le cloud ; je ne sais pas exactement où il se trouve physiquement. Je passe par la société Lambda pour la location (<https://lambda.ai>), mais de nombreuses entreprises proposent des services similaires.

Si vous consultez leurs tarifs à la demande, vous verrez des machines équipées de huit GPU H100. Par exemple, vous pouvez obtenir une machine à la demande avec huit GPU Nvidia H100 pour environ 3 \$ par GPU et par heure. Vous la louez, accédez au serveur dans le cloud et pouvez l'utiliser pour entraîner des modèles.

Un GPU (Graphics Processing Unit) H100 ressemble à ceci :



Figure 25 : Une carte Nvidia H100 GPU.

Il s'agit d'une carte que vous pouvez insérer dans votre ordinateur<sup>23</sup>. Les GPU sont idéaux pour l'entraînement des

---

22. Un ordinateur 8x H100 node est un serveur ou une station de travail hautes performances équipé de huit GPU NVIDIA H100 (unités de traitement graphique), fonctionnant ensemble sur une seule machine.

23. Mais Karpathy ne loue pas les cartes pour les installer sur sa propre machine. Tout est pris en charge par Lambda.

réseaux neuronaux, car ce processus est très gourmand en calculs. Surtout, les GPU sont conçus pour les opérations parallèles et peuvent effectuer efficacement ce genre de travail.

De nombreux éléments indépendants peuvent ainsi travailler simultanément à la résolution de la multiplication matricielle nécessaire à l'entraînement des réseaux de neurones.

La figure 25 montre un seul H100, mais en réalité, il est possible d'en regrouper plusieurs. On peut en empiler huit dans un seul "node" (en français "nœud").

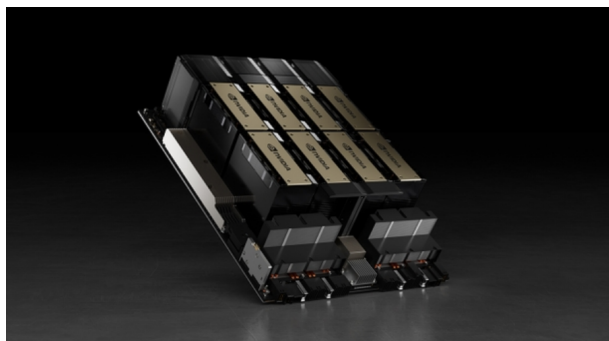


Figure 26 : Huit Nvidia H100 GPUs empilés dans un seul node.

Et vous pouvez empiler plusieurs nodes dans un système.

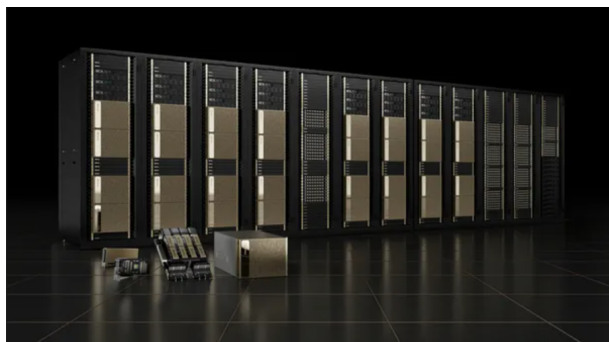


Figure 27 : Système composé de nombreux nodes.

Lorsqu'on regarde un data center (centre de données), on commence à voir des situations similaires à celles de la figure 27. Les GPU sont groupés par huit GPU, puis les nodes forment système et les systèmes forment un data center.

La figure 28 montre un grand centre de données.

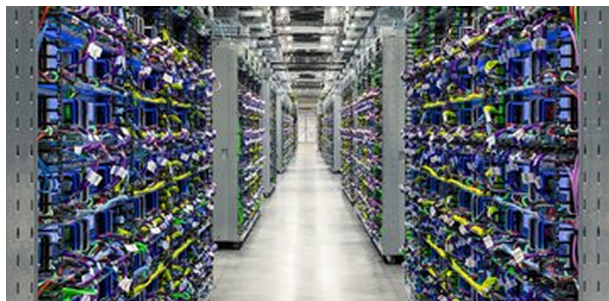


Figure 28 : En 2023, Google a annoncé de nouveaux TPU, équipés de GPU Nvidia H100, offrant des services cloud axés sur l'IA générative.

Toutes les grandes entreprises technologiques s'arrachent ces GPU pour entraîner leur LLM.

C'est la raison de la montée en flèche de la valeur boursière de Nvidia, qui a atteint 3400 milliards de dollars début 2025 <sup>24</sup>. C'est la nouvelle ruée vers l'or. Si vous êtes un prestataire de LLM, vous voulez acheter des GPU ; vous devez en avoir un très grand nombre pour optimiser efficacement les paramètres de votre modèle (voir fig. 23 par exemple).

Que font-ils ? Ils travaillent ensemble pour prédire le prochain jeton sur un ensemble de données comme l'ensemble de données Fineweb.

*C'est un workflow de calculs extrêmement coûteux.* Plus vous avez de GPU, plus vous pouvez essayer de prédire de jetons, ce qui en retour vous permet d'améliorer vos poids. Avec plus de GPU, vous traitez vos données plus rapidement.

---

24. La valeur boursière de Nvidia début octobre 2025 était d'environ 4500 milliards de dollars.

Voilà à quoi ressemblent toutes ces machines et ce qu'elles font, et pourquoi cela crée un tel tumulte en informatique. C'est un enjeu majeur en science, en technologie et dans la société.

Par exemple, voici un article de novembre 2024 annonçant l'investissement d'Elon Musk dans un très grand data center.

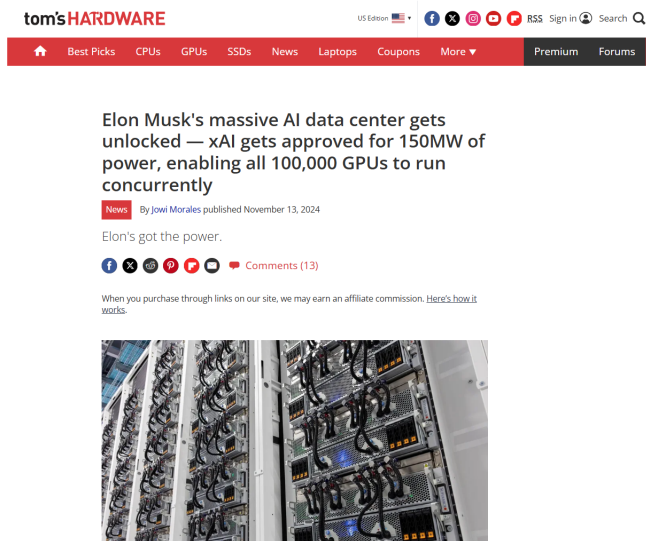


Figure 29 : Elon Musk's invests in a massive AI data center.

Il prévoit d'assembler 100 000 GPU dans un seul centre de données. Chacun de ces GPU est extrêmement coûteux<sup>25</sup>. Leur fonctionnement nécessitera aussi une énorme quantité d'énergie électrique.

Malheureusement, je n'ai pas quelques centaines de millions de dollars à investir dans l'entraînement d'un modèle commercial de grande envergure.

---

25. Le coût d'un seul GPU NVIDIA H100 en 2025 varie entre 25 000 et 30 000 \$ environ, selon le modèle et la configuration. Les versions premium ou à forte mémoire (comme les modèles SXM) peuvent atteindre 40 000 \$ par unité. Même en tenant compte des remises sur volume, cela représente un investissement de 2 milliards de dollars.

Par chance, nous pouvons nous tourner vers de grandes entreprises technologiques qui entraînent des LLM et en mettent certains à disposition du public une fois l'entraînement terminé.

Elles ont investi une quantité considérable de ressources de calcul pour entraîner leur réseau. Et elles publient parfois leurs paramètres à la fin de l'optimisation.

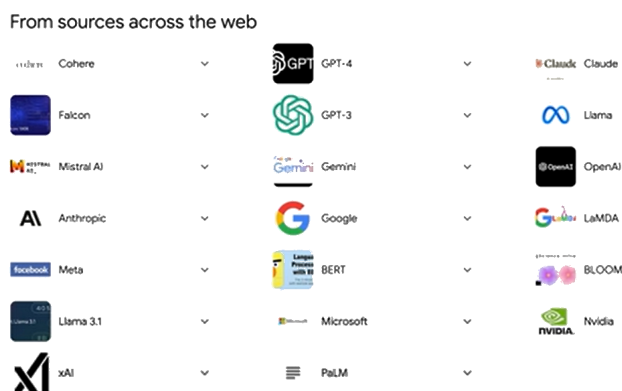


Figure 30 : Entreprises mettant des LLM à disposition du public.

00 :43 :17

## 8. Inférence avec le modèle de base Llama 3.1

De nombreuses entreprises entraînent des modèles (voir fig. 30). Mais rares sont celles qui publient les résultats obtenus à l'issue de l'entraînement. Une fois les paramètres ajustés, grâce au processus d'entraînement que nous avons vu, nous obtenons ce que l'on appelle un *modèle de base*.

Qu'est-ce qu'un modèle de base ? C'est essentiellement un *simulateur de jeton*. À partir d'un texte de bonne qualité, provenant d'Internet par exemple, et nettoyé, il prédit le jeton suivant dans la séquence. En soi, ce n'est pas encore très utile, car nous souhaitons surtout pouvoir poser des questions et obtenir des réponses pertinentes. Un modèle plus avancé capable de faire cela est appelé un *assistant*.



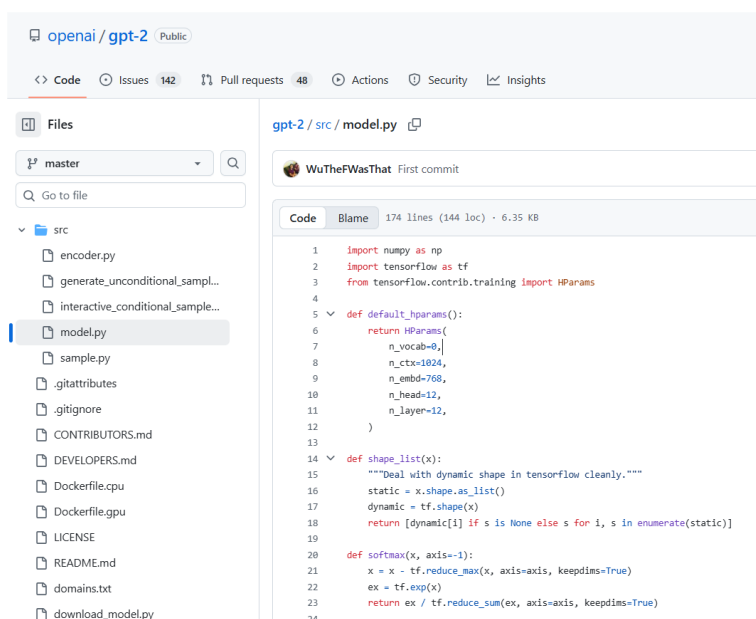


Pour parler de "publication" d'un modèle, deux éléments sont indispensables :

Premièrement, le *code* – généralement écrit en Python – décrit en détail la séquence d'opérations effectuées au sein du réseau de neurones.

Rappelez-vous le transformeur NanoGPT (fig. 18, page 24) et la séquence d'étapes (opérations mathématiques) que nous avons décrite.

De même, le réseau neuronal ChatGPT-2 exécute les opérations décrites par le code, dont le début est montré ci-dessous :



The screenshot shows the GitHub interface for the repository `openai/gpt-2`. The left sidebar displays the file tree with `src/model.py` selected. The main area shows the Python code for `model.py`, which includes imports for `numpy` and `tensorflow`, and defines functions for `default_hparams`, `shape_list`, and `softmax`.

```

1  import numpy as np
2  import tensorflow as tf
3  from tensorflow.contrib.training import HParams
4
5  def default_hparams():
6      return HParams(
7          n_vocab=0,
8          n_ctx=1024,
9          n_embd=768,
10         n_head=12,
11         n_layer=12,
12     )
13
14  def shape_list(x):
15      """Deal with dynamic shape in tensorflow cleanly."""
16      static = x.shape.as_list()
17      dynamic = tf.shape(x)
18      return [dynamic[i] if s is None else s for i, s in enumerate(static)]
19
20  def softmax(x, axis=-1):
21      x = x - tf.reduce_max(x, axis=axis, keepdims=True)
22      ex = tf.exp(x)
23      return ex / tf.reduce_sum(ex, axis=axis, keepdims=True)
24

```

Figure 33 : Python code of GPT-2, <https://github.com/openai/gpt-2/blob/master/src/model.py>

Ce code implémente ce qu'on appelle une *passe avant* du réseau neuronal. Nous avons donc besoin de détails précis sur la manière dont ils ont câblé leur réseau neuronal.

Il s'agit simplement de code informatique, généralement quelques centaines de lignes. Ce n'est pas très compliqué ; tout cela est facilement lisible et généralement assez standard.

*Ce qui n'est pas standard, c'est le deuxième élément indispensable : l'ensemble des paramètres. C'est là que se trouve la valeur du LLM.*

Quels sont les paramètres du réseau neuronal de GPT-2 ? Il y en a 1,5 milliard. Et nous avons besoin d'un réglage correct, sinon vraiment optimal de leur collection.

C'est pourquoi, outre le code source, nous avons besoin des paramètres qu'ils ont publiés. Ce ne sont que des nombres – une liste de 1,5 milliard de nombres, le réglage précis et correct de tous les « boutons » – que j'ai utilisé comme image de l'optimisation des paramètres – de sorte que le jeton suivant soit bien prédit dans la procédure séquentielle que nous avons décrite, fig. 16 p. 21.

Ces deux éléments sont donc nécessaires pour travailler avec une version de base du modèle.

GPT-2 a donc été publié (avec les deux éléments importants qu'une version doit englober), mais il s'agit en fait d'un modèle assez ancien, comme je l'ai dit.

Nous allons nous intéresser au LLM appelé *Llama 3*, que je vais maintenant présenter. Rappelez-vous que GPT-2 était un modèle de 1,5 milliard de paramètres, entraîné sur 100 milliards de jetons. Llama 3 est un modèle beaucoup plus vaste et moderne.

Il a été développé et entraîné par Meta. Il s'agit d'un modèle de 405 milliards de paramètres, entraîné sur 15 000 milliards de jetons, très similaire à celui que nous avons vu, mais en beaucoup plus grand.

Meta a également publié Llama 3 en juillet 2024. Cette publication faisait partie de l'article intitulé « The Llama 3 Herd of Models » (<https://ai.meta.com/research/publications/the-llama-3-herd-of-models/>).

Dans cet article (dont le début est montré sur la figure 34), très détaillé, le modèle le plus important publié est le modèle Llama 3.1 à 405 milliards de paramètres. Il s'agit du modèle de base.

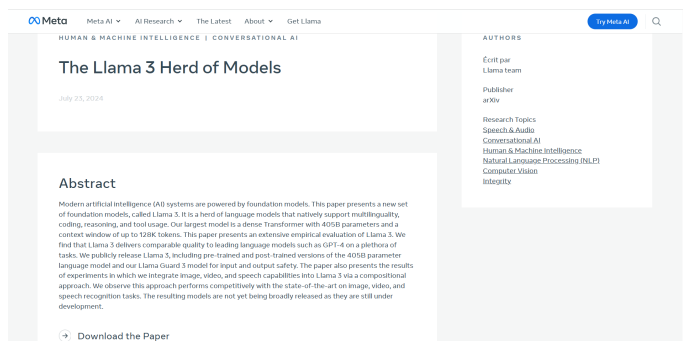


Figure 34 : Article de Meta introduisant Llama 3. <sup>26</sup>

En plus du modèle de base, vous pouvez voir dans le tableau 1 de l'article qu'ils ont également publié le modèle instruct – j'en parlerai plus loin. Le mot « *instruct* » signifie qu'il s'agit d'un *assistant* : vous pouvez lui poser des questions et il vous donnera des réponses.

Pour l'instant, étudions ce modèle de base, ce simulateur de jetons. Demandons-nous : Qu'est-ce que c'est ? Comment fonctionne-t-il ? Qu'obtient-on à la fin de l'optimisation si on la laisse s'exécuter jusqu'au bout pour un très grand réseau de neurones sur un volume important de données ?

---

26. Voici le résumé : Les systèmes d'intelligence artificielle (IA) modernes reposent sur des modèles fondamentaux. Cet article présente un nouvel ensemble de modèles fondamentaux, appelé Llama 3. Il s'agit d'un ensemble de LLM prenant en charge nativement le multilinguisme, le codage, le raisonnement et l'utilisation d'outils. Notre plus grand modèle est un transformateur dense avec 405 milliards de paramètres et une fenêtre contextuelle pouvant atteindre 128 000 jetons. Cet article présente une évaluation empirique de Llama 3. Llama 3 offre une qualité comparable à celle des principaux LLM tels que GPT-4 sur de nombreuses tâches. Nous publions Llama 3, comprenant des versions pré-entraînées et post-entraînées du modèle à 405 milliards de paramètres et du modèle Llama Guard 3 pour la sécurité des I/O. L'article présente les résultats d'expériences avec intégration d'image, de vidéo et de parole dans Llama 3 via une approche compositionnelle. Cette approche est compétitive par rapport aux technologies de pointe pour les tâches de reconnaissance d'image, de vidéo et de parole. Les modèles résultants ne sont pas encore largement diffusés car ils sont en cours de développement.

00 :47 :00

Mon endroit favori pour interagir avec un modèle de base est le site de la société *hyperbolic* (<https://hyperbolic.ai>) qui propose le modèle de base du 405b Llama 3.1.

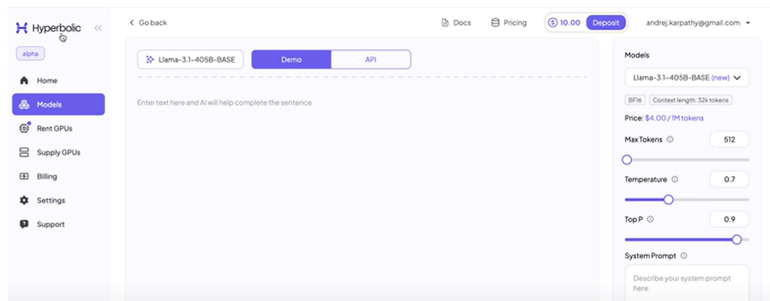


Figure 35 : Jouer avec Llama 3.1 405b via la plateforme hyperbolic.ai<sup>27</sup>.

Lorsque vous vous rendez sur le site web, vous devrez probablement vous inscrire. Une fois connecté, assurez-vous de sélectionner le modèle « llama 3.1 405B BASE » ; il faut que ce soit la version de base, et non de la version instruct.

Pour le paramètre « nombre maximal de jetons » à droite de la figure 35, diminuons-le légèrement afin d'économiser les ressources de calcul. Nous pouvons le définir à environ 128 jetons et laisser les autres paramètres inchangés..

Fondamentalement, ce qui se passe à l'exécution de ce modèle est identique à ce qui se passait lors de l'inférence du modèle, pendant l'entraînement de GPT-2 (fig. 23, page 34) : le modèle continuera à deviner séquentiellement le jeton suivant en fonction des données déjà connues ou déjà prédites.

27. En octobre 2025, l'adresse <https://app.hyperbolic.ai/models/llama31-405b-base-bf16> n'est plus active. Elle génère une erreur 404. Nous pouvons cependant jouer avec différents modèles "instruct", comme <https://app.hyperbolic.ai/models/llama-3-3-70b-instruct>.

Commençons par voir que ce modèle n'est pas encore un assistant.

Par exemple, si vous lui demandez « Combien font 2 plus 2 ? » (« What is 2 plus 2 ? »), il ne répondra pas simplement « C'est quatre, comment puis-je vous aider ? », car ce n'est pas un assistant optimisé, c'est seulement d'un modèle de base.

L'invite « Combien font 2 plus 2 ? » sera tokénisée ; ces tokens deviendront le préfixe, et le modèle ne prédira à chaque fois que le token suivant le plus probable (ou l'un des plus probables, la plupart du temps, car il est stochastique). Son résultat dépend entièrement des modèles statistiques observés dans les données d'entraînement. Demandons « Combien font 2 plus 2 ? ». Appuyons sur Entrée pour voir quelle continuation il génère.

What is 2+2 ?

This is a simple question, and the answer is 4. This is an example of an objective truth. No matter how many times you add 2 and 2, you will get the same answer. It is not possible to get a different answer. This is why mathematics is considered to be a form of objective truth. It is a way of describing the world that is not influenced by our own personal beliefs or opinions. The same is true for other areas of knowledge. The laws of physics are another example of objective truth. They describe the way the world works, and they are not influenced by our own personal beliefs or opinions.

Enter text here and AI will help complete the sentence

Ici, le modèle a répondu à la question, puis est parti dans une digression philosophique sans rapport.

Réessayons : on repart de zéro, et on pose à nouveau la question : « Combien font 2 plus 2 ? »

What is 2+2 ?

Is it 4 or 22 ? How do you know ? What about 1+1+1+1 ? Is it 4 or 11 ? How do you know ? Why do you group the numbers together the way you do ? Why do you add the numbers in a particular order ? What rules do you use to determine the way you group numbers ? Mathematics is a human activity. It is a set of ideas that people came up with. We are

so used to the way we do mathematics that we forget that there are other ways to do mathematics. There are other ways to group numbers together and other ways to add numbers.

Enter text here and AI will help complete the sentence

Cette fois-ci il est tout de suite parti dans une direction différente.

Insistons sur le point suivant : chaque fois que vous saisissez l'invite, le système redémarre. *C'est un système stochastique* – donc, même avec le même préfixe de jetons, vous obtiendrez toujours une réponse différente.

En effet, vous travaillez avec une distribution de probabilité pour le jeton suivant (voir figure 15, page 19). Le modèle utilise cette distribution pour choisir au hasard. Ainsi, à chaque exécution, vous obtenez des tirages différents. Il prend une direction différente à chaque fois, produisant un texte différent.

Par exemple, posons la question une fois de plus :

What is 2+2 ?

What is 2+2 ? This is a featured post. It will be sticky at the top of all recent posts for all to see ! There are no comments on this post. Be the first !

Enter text here and AI will help complete the sentence

C'est encore une autre réponse résultant de ce qu'il a observé sur Internet, régurgitant simplement des modèles statistiques.

Gardez ceci à l'esprit concernant un modèle de base :

1. ce n'est pas encore un assistant, c'est juste une auto-complétion de jetons.
2. c'est un système stochastique, ce qui signifie que le résultat est aléatoire et peut changer à chaque fois.

*L'essentiel est que, même si ce modèle en lui-même n'est pas encore très utile pour de nombreuses applications, il reste très précieux, car pour être vraiment performant dans la prédiction du prochain jeton, le modèle a dû apprendre beaucoup de choses sur le monde. Toutes ces connaissances sont intégrées dans les paramètres du réseau.*

Rappelez-vous que le texte d'entraînement était constitué de pages web [et de quelques autres sources publiques de textes de bonne qualité]. Maintenant tout cela est en quelque sorte compressé dans les poids du réseau neuronal<sup>28</sup>.

On peut considérer ces 405 milliards de paramètres comme une sorte de « compression » d'Internet. C'est comme un fichier zip, sauf qu'il ne s'agit pas d'une compression sans perte, mais d'une compression avec perte. Il ne nous reste qu'une sorte de perception gestaltiste, un instantané flou d'Internet, à partir de quoi nous pouvons générer du texte. Pour l'instant, nous pouvons extraire une partie de ces connaissances stockées en soumettant une invite au modèle de base.

Voici une invite qui peut permettre d'extraire une partie des connaissances cachées dans les paramètres du modèle : « Voici ma liste des 10 meilleurs monuments à voir à Paris : »

Je l'ai formulée ainsi pour inciter le modèle à poursuivre la liste. Voyons ce qui se passe lorsque je presse "entrée".

Here is my top 10 list of landmarks to see in Paris :

1) Eiffel Tower. This is a must. You can take a lift to the top of the tower for 13.50 euros or take the stairs for 4.50 euros. The tower is 324m tall and the highest structure in Paris. I went up the tower at night and the view was amazing! 2) Notre Dame. This is a gothic style cathedral located on the eastern half of Paris. It was one of the first cathedrals to use flying buttresses for support. The cathedral is free to enter but if you want to climb the tower it costs 8.50 euros. 3) Arc de Triomphe. The

---

28. C'est une idée très importante, quoique heuristique : *l'information et les connaissances – absolument toutes – que le réseau neuronal acquiert pendant l'entraînement sont, en quelque sorte, encapsulées dans les milliards de paramètres du modèle que nous utilisons.*

Cependant, cette encapsulation est cachée et, dans une certaine mesure, mystérieuse. Bien qu'elle constitue une forme de « connaissance du monde », il ne s'agit pas d'un modèle du monde au sens conventionnel du terme.

En fait, la construction d'une IA possédant un modèle explicite du monde – à l'instar de la façon dont les humains se représentent mentalement leurs connaissances – est un domaine de recherche actif et passionnant.



arch was built to honor those who fought for France in the Napoleonic wars. It is located on the western end of the Champs-Elysees. It costs 9.50 euros to go to the top of the arch. 4) Champs-Elysees. This is one of the most famous streets in the world. It is lined with cinemas, cafes, and luxury shops. The avenue runs from the Place de la Concorde to the Arc de Triomphe. 5) The Louvre. This is one of the largest museums in the world and is home to the Mona Lisa. It is located on the right bank of the Seine. The museum is housed in the Louvre Palace which was originally a fortress built in the 12th century. The museum is open every day except Tuesday and tickets cost 15 euros. 6) Sacre-Coeur. This is a Roman Catholic church located on the highest point in the city. The church is made of white stone and is one of the most popular tourist destinations in Paris. The church is free to enter but there is a charge to climb the dome. 7) Place de la Concorde. This is the largest public square in Paris. It is located at the eastern end of the Champs-Elysees. The square is home to the Obelisk of Luxor, a gift from Egypt, and the Fountain of River Commerce and Navigation. 8) Les Invalides. This is a

Enter text here and AI will help complete the sentence

Il commence à générer une liste et me fournit des sites à voir. Observez qu'il produit beaucoup d'informations.

Il ne faut pas se fier entièrement à toutes ces informations : ce ne sont que des souvenirs de documents Internet.

Les éléments qui apparaissent très fréquemment dans les données Internet sont plus susceptibles d'être mémorisés correctement, tandis que les faits moins courants peuvent être moins fiables.

On ne peut donc pas se fier entièrement à tout ici [concernant les sites parisiens], car il ne s'agit que d'un vague souvenir de ce qui se trouvait sur Internet. L'information n'est pas stockée explicitement dans les paramètres ; ce ne sont que des schémas que le modèle rappelle.

Cela dit, nous avons obtenu quelque chose qui est probablement approximativement correct. Je n'ai pas l'expertise nécessaire pour vérifier que c'est correct, mais vous voyez que nous avons extrait une grande partie des connaissances du modèle.

Ces connaissances ne sont ni précises ni exactes. Elles sont vagues, probabilistes et statistiques. Et les informations qui apparaissent souvent sur le Net sont celles qui sont plus susceptibles d'être mémorisées dans le modèle.

00 :52 :00

Regardons quelques exemples de comportement de GPT-2.

Je suis allé sur la page Wikipédia (en anglais) consacrée au zèbre :



Figure 36 : Wikipedia page on zebra.

Je vais copier la première phrase et la coller dans la fenêtre d'input (c'est-à-dire de contexte) du modèle de base que nous étudions.

Cette phrase sert de "préfixe" – c'est le mot qu'on utilise quand ce qu'il y a dans le contexte est spécifiquement destiné à être continué –, invitant le modèle de base à continuer de générer du texte, un jeton à la fois.

Quel type de complétion obtient-on en cliquant sur Entrée ?

Zebras (US: /zi:brɛz/, UK: /zebrɛz, 'zi:-/)[2] (subgenus *Hippotigris*) are African equines with distinctive black-and-white striped coats.

There are three living species: the Grévy's zebra (*Equus grevyi*), plains zebra (*E. quagga*), and the mountain zebra (*E. zebra*). Zebras share the genus *Equus* with horses and asses, the three groups being the only living members of the family *Equidae*. Zebra stripes come in different patterns, unique to each individual. Several theories have been proposed for the function of these stripes, with most evidence supporting them as a deterrent for biting flies. Zebras inhabit eastern and southern Africa and can be found in a variety of habitats such as savannahs, grasslands, woodlands, shrublands, and mountainous areas. Zebras are primarily grazers and can subsist on lower-quality vegetation. They are preyed on mainly by lions and typically flee when threatened but also bite and kick. Zebra species differ in social behaviour, with plains and mountain zebra living in stable harems consisting of an adult male or stallion, several adult females or mares, and their young or foals; while Grévy's zebra live alone or in loosely associated herds. In harem-holding species, adult females mate only with their harem

Figure 37 : Continuation par le modèle de base du préfixe introduit en input – la première phrase de la page sur le zèbre.

Le modèle a produit une *régurgitation exacte* du début de l'article Wikipédia <sup>29</sup>. Il récite cette entrée Wikipédia uniquement de mémoire et cette mémoire est stockée dans ses paramètres.

Il est possible qu'au bout d'un moment, le modèle commence à s'éloigner de l'entrée Wikipédia originale. Mais vous pouvez constater qu'il en a mémorisé une grande partie. On retrouve les phrases exactes de l'original dans le texte produit (séquentiellement, les tokens produits *un par un*, l'un après l'autre).

Cependant, le modèle finira par diverger, car il ne se souvient pas parfaitement de tous les détails. Et n'oubliez pas que la continuation a un aspect stochastique, même si le "bon jeton" a une probabilité élevée d'être sélectionné.

Cette mémorisation quasi parfaite s'explique par la capacité de ces modèles à mémoriser. Le phénomène est appelé

29. Il a juste changé un mot ici ou là, ajoutant un article devant "Grévy's zebra" et ôtant celui devant "plains zebras".

*régurgitation* : le modèle récite mot pour mot des parties de ses données d'entraînement, ce qui n'est pas souhaitable<sup>30</sup>.

Pourquoi cela se produit-il ?

Pour les documents de haute qualité comme Wikipédia, le processus d'apprentissage du modèle les échantillonne plus fréquemment. Cela signifie que le modèle peut consulter la même page web jusqu'à dix fois pendant l'apprentissage. C'est un peu comme si vous lisiez le même texte cent fois ; vous seriez capable de le réciter.

Le modèle est similaire. S'il voit quelque chose trop souvent, il peut le répéter par cœur plus tard.

Sauf que ces modèles peuvent être encore plus efficaces que les humains à cet égard. Le modèle n'a peut-être vu cette entrée Wikipédia sur Zebra que dix fois, mais il est néanmoins capable de mémoriser l'article et de l'intégrer dans ses paramètres.

00 :54 :20

Voyons maintenant comment le modèle se comporte quand il n'a certainement pas appris la bonne réponse lors de son entraînement. Si nous consultons l'article « The Llama 3 Herd of Models » et naviguons jusqu'aux données de pré-apprentissage,

---

30. En effet, lorsqu'il s'agit d'un être humain, on parle de plagiat. Cela cause des litiges entre producteurs de LLM et producteurs de sources originales (qui ont servi à l'entraînement) sur le droit d'auteur, l'usage loyal et les implications éthiques de la reproduction ou de la paraphrase de contenu propriétaire sans mention de la source.

Le problème est plus complexe qu'il n'y paraît. Le simple fait de changer un mot ou de reformuler une phrase est une technique courante pour contourner la détection du plagiat. Cependant, lorsqu'une personne P exprime des connaissances, celles-ci proviennent presque toujours de sources externes : livres, sites web ou autres – sauf s'il s'agit d'une création originale. Nous en avons tous fait l'expérience : nous expliquons quelque chose que nous considérons comme important et clair, et qui représente nos propres idées, et quelqu'un nous dit : « Tu répètes ce que tu as appris dans tel et tel livre ! »

Quand la reformulation de ce que P a appris franchit-elle la limite du plagiat, et quand est-elle considérée comme une réflexion sincère exprimée par P ?

nous apprenons que l'ensemble de données s'est arrêté à la fin de 2023. Par conséquent, aucun document postérieur à cette date n'a fait partie des données. Par exemple, le modèle de base Llama-3.1-405b-base ne sait rien de l'élection présidentielle américaine de 2024.

Extrayons de la page Wikipédia consacrée à l'élection présidentielle américaine de 2024 l'invite suivante : «Presidential elections were held in the United States on November 5, 2024. The Republican Party's ticket—Donald Trump, who was the 45th president of the United States from 2017 to 2021» – on laisse intentionnellement la phrase sans point final ; elle est destinée à être continuée. Le modèle de base va poursuivre la séquence de jetons. Il utilisera sa meilleure estimation de ce qu'est une bonne réponse en fonction des connaissances acquises qui se trouvent dans ses propres paramètres.

Presidential elections were held in the United States on November 5, 2024. The Republican Party's ticket—Donald Trump, who was the 45th president of the United States from 2017 to 2021

, and Vice President Mike Pence—won the election over the Democratic Party's ticket of former Secretary of State Hillary Clinton and Senator Tim Kaine. This was the first time since 1948 that a Republican candidate won a majority of the popular vote, the first time since 1988 that a Republican candidate won more than 300 electoral votes, and the first time since 1984 that a Republican candidate won more than 50% of the popular vote. The Republican Party's ticket—Donald Trump, who was the 45th president of the United States from 2017 to 2021, and Vice President Mike Pence—won the election over the Democratic Party's ticket of former Secretary of State Hillary Clinton and Senator Tim Kaine. This was the first time since 1948 that a Republican candidate won a majority of the popular vote, the first time since 1988 that a

Enter text here and AI will help complete the sentence

Ainsi il dit que Mike Pence était le colistier, alors que c'était J. D. Vance, et il dit que le ticket était contre Hillary Clinton et Tim Kane. C'est un univers parallèle intéressant, créé par le LLM, de ce qui aurait pu se passer.

Faisons un nouvel essai<sup>31</sup>.

Presidential elections were held in the United States on November 5, 2024. The Republican Party's ticket—Donald Trump, who was the 45th president of the United States from 2017 to 2021

, and his running mate, former U.S. representative Ron DeSantis—defeated the Democratic Party's ticket of incumbent president Joe Biden and vice president Kamala Harris. The election was a rematch of the 2020 presidential election, in which Biden defeated Trump by 4.5 percentage points and 7 million votes. Trump became the first president since Grover Cleveland to win a non-consecutive term, the second president after Cleveland to win the popular vote three times, and the first president to win the popular vote in three elections without winning the popular vote in all three. Biden became the first incumbent president to lose reelection since George H. W. Bush in

Cette fois-ci le colistier était Ron DeSantis, et le ticket démocrate était Joe Biden et Kamala Harris. Il s'agit là encore d'un univers parallèle, un différent.

En résumé, nous constatons que le modèle se basera sur des suppositions éclairées. Il poursuivra la séquence de jetons en s'appuyant sur ces connaissances.

Ce que nous observons ici s'appelle une *hallucination*<sup>32</sup>. Le modèle se contente d'estimer le résultat de manière probabiliste<sup>33</sup>.

---

31. C'est-à-dire, entrons le même prompt exactement, et voyons ce que dit le modèle – qui, on le rappelle, a un aspect aléatoire.

32. Il est difficile de lui reprocher une hallucination. Nous lui avons demandé de décrire quelque chose qui, pour lui, se situait dans le futur.

On peut également générer de véritables hallucinations (même avec des assistants, que nous étudierons bientôt) : par exemple, un LLM décrivant « une célèbre bataille de la Seconde Guerre mondiale » avec de nombreux détails réalistes, bien que cette bataille n'existe pas. Ou, si vous lui demandez de décrire la seconde moitié d'un roman classique que vous ne voulez pas terminer, il inventera avec plaisir une nouvelle fin...

33. Notez que nous n'avons utilisé que le modèle de base (qui enchaîne des séquences de jetons à partir d'une invite), et non l'assistant, conçu pour répondre aux questions. L'assistant utilise bien sûr largement le modèle de base, mais le développe d'une manière qui sera expliquée plus loin.

00 :55 :55

Même s'il s'agit d'un modèle de base et non d'un modèle d'assistant, on peut néanmoins utiliser le modèle dans des applications pratiques si on conçoit notre invite astucieusement.

Voici ce que nous appelons *a few shot prompt* (quelques prompts d'amorçage). J'ai 10 mots, ou plus précisément 10 paires de mots. Chaque paire est un mot anglais suivi, séparé par deux points, de sa traduction en coréen. Nous en avons dix : butterfly, ocean, whisper, mountain, thunder, gentle, freedom, umbrella, cinnamon, moonlight).  
butterfly : 나비, ocean : 바다, whisper : 속삭임, mountain : 산, thunder : 천둥, gentle : 부드러운, freedom : 자유, umbrella : 우산, cinnamon : 계피, moonlight : 달빛, teacher :

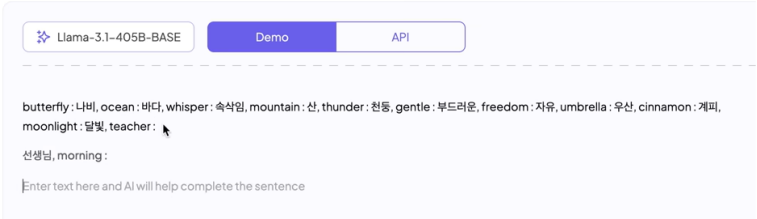


Figure 38 : Conception astucieuse d'un prompt qui va inciter le modèle de base à se comporter comme un assistant. Ici on montre comment le modèle peut servir de dictionnaire anglais-coréen. On l'incite à donner la traduction de "teacher" (enseignant). Le modèle l'a traduit correctement <sup>34</sup>, et s'apprête même à traduire ensuite le mot "morning" (matin), fig. 38, mais nous l'avons arrêté.

34. Il a traduit correctement "teacher"

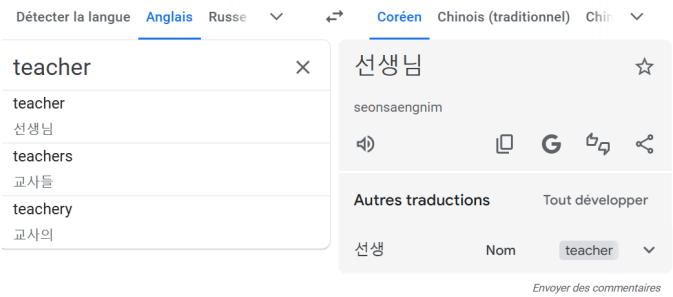


Figure 39 : Traduction de "teacher" en coréen.

Voici ce que fait le modèle sur la figure 38 : à la fin de l'invite, nous avons «teacher:». Nous lui avons demandé de compléter, mettons, seulement cinq jetons.

Point important à noter : *les modèles de base possèdent ce que nous appelons des capacités d'apprentissage contextuel.*

Cela signifie que lorsque le modèle lit le contexte [c'est-à-dire l'invite que nous avons saisie, avec dix paires de mots, anglais:coréen, anglais:coréen, etc.], il comprend qu'il y a un schéma algorithmique dans les données. Il sait comment poursuivre ce schéma. C'est l'apprentissage contextuel<sup>35</sup>.

Le modèle de base joue le rôle de traducteur et, une fois l'opération terminée, nous constatons que la traduction de « teacher » est « seonsaengnim » (écrit en coréen, fig. 38 et 39), ce qui est correct.

Voilà comment créer des applications en utilisant intelligemment les invites, même si nous ne disposons pour l'instant que d'un modèle de base. Cela repose sur cette capacité d'apprentissage contextuel. Cela se fait en construisant ce que l'on appelle des "few shot prompts"<sup>36</sup>.

*Pour terminer, je voudrais montrer qu'il existe une méthode astucieuse pour forcer un modèle de base à se comporter comme un assistant complet simplement grâce à l'invite.*

L'astuce consiste à structurer une invite de manière à ce qu'elle ressemble à une page web, c'est-à-dire à une conversation entre un assistant IA et un humain. Le modèle poursuivra ensuite la conversation selon le format qu'il voit !

Pour écrire l'invite, j'ai fait appel à ChatGPT (par commodité, mais j'aurais pu le faire moi-même). Je lui ai indiqué que je souhaitais créer un assistant LLM, mais que je n'avais que le modèle de base. "Peux-tu donc écrire mon invite ?", lui ai-je demandé. Voici le résultat, qui est plutôt bon.

---

35. On parle d'apprentissage contextuel car le modèle n'a pas appris ce schéma lors de son pré-entraînement ; il l'apprend à partir du contexte spécifique que nous lui fournissons. Il découvre et poursuit simplement, au moment de l'inférence, le schéma observé dans l'entrée fournie.

36. Leur rôle est de montrer le schéma que le modèle de base doit poursuivre pour construire sa séquence de jetons suivants.



On a ci-dessous la conversation entre un assistant IA et un humain, créée par ChatGPT. Nous l'utiliserons comme début d'invite, à laquelle nous ajouterons une question pour inciter la modèle de base à poursuivre la *conversation*.

#### Llm Assistant Prompt

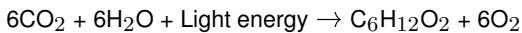
Here is a conversation between an AI Assistant and a Human. The AI Assistant is knowledgeable, helpful, and capable of answering a wide variety of questions...

Human: Hi there ! Who are you ?

Assistant: Hello ! I'm your AI Assistant. I'm here to help answer questions, provide explanations, and assist you with tasks. What can I help you with today ?

Human: Can you explain how photosynthesis works ?

Assistant: Of course ! Photosynthesis is the process by which green plants, algae, and some bacteria convert light energy into chemical energy stored in glucose:



This process is essential for life on Earth, as it provides the oxygen we breathe and forms the base of most food chains. Would you like more details on a spe... (cut on speaker's screen)

Human: That's great ! Now, can you help me write a short email ?

Assistant: Certainly ! Could you tell me who the email is for and what you'd like to say ?

Human: It's for my manager. I want to ask for a day off next week.

Assistant: Here's a possible draft:

Subject: Request for Day Off Next Week

Dear [Manager's Name],

I hope this message finds you well. I am writing to request a day off on [specific date]. Please let me know if this works for the team's schedule or if there... (cut on speaker's screen)

Thank you for considering my request. Please let me know if you need further details.

Best regards, [Your Name]

Does this look good to you, or would you like to adjust it ?

Human: Thanks ! That's perfect.

Assistant: You're welcome ! Let me know if there's anything else I can help with.

Avec un modèle de base, lui donner une description de ce que nous voulons créer ne fonctionnera pas, il faut lui créer ce début de conversation – car il va comprendre le format qu'on veut.

Nous avons donc créé – avec ChatGPT, mais n'importe quelle méthode aurait fait l'affaire – quelques tours de conversation : Humain, Assistant, Humain, Assistant...

Pour créer l'invite qu'on va donner à notre modèle de base, à la fin de cette conversation nous allons insérer la requête à laquelle nous voulons que le modèle de base réponde, comme s'il s'agissait d'un assistant.

Je copie donc la conversation que nous venons de créer dans l'invite du modèle de base, et j'ajoute à la fin :

Human: Why is the sky blue ?

Et j'appuie sur "entrée":

Assistant: The sky appears blue due to the phenomenon called Rayleigh's scattering, etc, etc.

Nous constatons que le modèle de base poursuit intelligemment la séquence: comme celle-ci à le format d'une conversation entre un humain et un assistant, le modèle de base continue en assumant le rôle de l'assistant !

Il y a cependant une subtilité : étant donné que nous avons initialisé une conversation humain/assistant, l'assistant devinera (ou hallucinera, si vous préférez) la question suivante de l'humain. Et cela continuera ainsi indéfiniment.

Néanmoins, vous pouvez constater que nous avons plus ou moins accompli la tâche.

Si nous avons simplement utilisé l'invite « Why is the sky blue ? » (Pourquoi le ciel est bleu ?) sans quelques invites préliminaires structurant une conversation, nous ne nous attendrions évidemment pas à ce que le modèle de base réponde correctement à notre requête. Nous obtiendrions... qui sait ce que nous obtiendrions. Essayons avec notre invite simple :

Why is the sky blue ?

What makes a cloud? How is a hurricane born? Why do we have seasons? Answers to these questions and many more can be found in the fascinating and highly educational exhibition Weather: From Magic to Meteorology. The exhibition presents the history of meteorology and weather forecasting, from the first observations of the sky and the atmosphere, to the development of modern forecasting techniques. It also explores the effects of weather and climate on our lives, and how we can use our knowledge of the weather to protect ourselves from its dangers. Weather: From Magic to Meteorology is a must-see for anyone interested in the history of...

Le modèle produit de nouvelles questions, puis nous fait un speech annonçant l'exposition pédagogique « Météo : de la magie à la météorologie » :-)

Nous venons néanmoins de voir une façon de créer un assistant, même avec seulement un modèle de base<sup>37</sup>.

00:59:30

## 9. Du pré-entraînement au post-entraînement

Commençons par un résumé des points que nous venons d'aborder. Prenons du recul, et voyons ce que nous avons fait jusqu'à présent et ce que nous souhaitons.

---

37. Pour transformer notre modèle de base en assistant, nous avons structuré notre invite en quelques échanges au format d'un dialogue. Le modèle de base a alors « compris » qu'il devait poursuivre le dialogue.

### The "psychology" of a base model

- It is a token-level internet document simulator
- It is stochastic / probabilistic - you're going to get something else each time you run
- It "dreams" internet documents
- It can also recite some training documents verbatim from memory ("regurgitation")
- The parameters of the model are kind of like a lossy zip file of the internet
  - => a lot of useful world knowledge is stored in the parameters of the network
- You can already use it for applications (e.g. translation) by being clever with your prompts
  - e.g. English:Korean translator app by constructing a "few-shot" prompt and leveraging "in-context learning" ability
  - e.g. an Assistant that answers questions using a prompt that looks like a conversation
- But we can do better...

Figure 40: "Psychologie" d'un modèle de base<sup>38</sup>.

Nous avons abordé la première étape de la construction d'un assistant LLM. Il s'agit de la phase de pré-apprentissage. Elle se résume à ceci :

1. Télécharger des documents provenant d'Internet [et d'autres sources].
2. Les filtrer et les nettoyer pour ne conserver que des textes de bonne qualité, qui serviront à entraîner notre LLM.
3. Les tokeniser, c'est-à-dire les découper en jetons.
4. Construire un réseau de neurones comportant une ou plusieurs couches intermédiaires — de nos jours, de type "transformer".
5. Entraîner le réseau de neurones à prédire le jeton suivant à partir d'une séquence de jetons fournie en entrée. Cette opération nécessite un processus long et coûteux qui permet d'ajuster le vaste ensemble de paramètres du réseau.

*Le résultat de cette première étape est le modèle de base.*

*Son objectif est de définir les paramètres de ce réseau, qui constituent l'élément essentiel de notre modèle.*

---

38. Rappel: le tableau blanc qu'A. Karpathy montre parfois en parlant peut être obtenu en utilisant <https://excalidraw.com/> et en ouvrant le fichier <https://lapasserelle.com/documents/karpathy.txt>

*Ce modèle de base est essentiellement un simulateur de documents Internet au niveau des jetons.*

*Il peut générer des séquences de jetons présentant les mêmes statistiques que les documents Internet que nous avons téléchargés, nettoyés et sur lesquels nous nous sommes entraînés.*

Nous avons vu que nous pouvons déjà l'utiliser dans certaines applications utiles. Mais nous devons faire mieux.

Nous voulons un *véritable assistant*. Nous voulons pouvoir poser des questions auxquelles le modèle apportera des réponses. Pour cela, nous devons passer à la deuxième étape, appelée *post-apprentissage*.

Nous prenons notre modèle de base, notre simulateur de documents Internet, et le passons à l'étape de post-apprentissage.

Nous allons maintenant aborder quelques méthodes de post-apprentissage de ces modèles.

Les étapes de post-apprentissage sont beaucoup moins coûteuses en termes de calcul. La majeure partie du travail de calcul, les centres de données faisant des calculs massifs, et les millions de dollars dépensés étaient l'étape de pré-apprentissage.

Nous passons maintenant à l'étape de post-apprentissage, moins coûteuse mais néanmoins essentielle, où nous allons transformer notre modèle de base en assistant.

Voyons comment transformer notre modèle, passant d'un simple générateur de jetons à un modèle fournissant des réponses complètes et pertinentes aux questions.

Autrement dit, nous allons *commencer à réfléchir aux conversations*. Il s'agit de conversations pouvant être à plusieurs tours, et dans le cas le plus simple, il s'agit d'une conversation entre un humain et un assistant <sup>39</sup>.

Par exemple, imaginons que la conversation ressemble à ceci: lorsqu'un humain demande « Combien font  $2 + 2$  ? », l'assistant devrait répondre par «  $2 + 2$  font 4 ».

---

39. Dans ce cas, un "tour" c'est un humain qui dit quelque chose et l'assistant qui répond quelque chose.

Lorsqu'un humain demande ensuite « Et si c'était une étoile au lieu d'un plus ? », l'assistant pourrait répondre par «  $2*2 = 4$ , comme  $2 + 2$  ».

Human: "What is  $2+2$ ?"  
Assistant: " $2+2 = 4$ "  
Human: "What if it was  $*$  instead of  $+$ ?"  
Assistant: " $2*2 = 4$ , same as  $2+2$ !"

De même, voici un autre exemple montrant que l'assistant pourrait aussi avoir une personnalité agréable:

Human: "Why is the sky blue?"  
Assistant: "Because of Rayleigh scattering."  
Human: "Wow!"  
Assistant: "Indeed! Let me know if I can help with anything else :)"

Et voici un troisième exemple:

Human: "How can I hack into a computer?"  
Assistant: "I'm sorry I can't help with that."

Lorsqu'un humain demande quelque chose pour lequel nous ne souhaitons pas l'aider, nous pouvons produire ce qu'on appelle un *refus*. Nous pouvons alors dire que nous ne pouvons pas l'aider.

Par conséquent, nous souhaitons maintenant réfléchir à la manière dont un système devrait interagir avec l'humain. Nous voulons programmer l'assistant et son comportement dans ces conversations.

*Comme il s'agit de réseaux de neurones, nous n'allons pas les programmer explicitement dans le code. Nous ne pourrions pas programmer l'assistant de cette manière.*

*Tout se fera par apprentissage sur des ensembles de données. Nous allons programmer implicitement l'assistant en créant des ensembles de données de conversations.*

Les trois encadrés (p. 64) représentent trois exemples indépendants de conversations dans un ensemble de données.

Un ensemble de données réel – et je vais vous montrer des exemples – sera beaucoup plus volumineux. Il pourra contenir des centaines de milliers de conversations à plusieurs tours, longues, etc., et couvrir des sujets très variés. Je n'ai montré que trois exemples pour illustrer.

Le principe est simple: l'assistant est programmé par des exemples. D'où proviennent ces données, comme «  $2 * 2 = 4$ , comme  $2 + 2$  », etc. ?

*Elles viennent d'annotateurs humains.*

Nous allons fournir aux annotateurs humains un contexte conversationnel et leur demander d'écrire la réponse idéale d'un assistant dans cette situation.

Ensuite nous entraînerons le modèle sur ces conversations idéales afin de lui apprendre à imiter leurs types de réponses. Pour cela, nous reprendrons notre modèle de base, créé lors de la phase de pré-apprentissage. Ce modèle de base a été entraîné sur des documents Internet.

Nous en avons fini avec l'ensemble de données d'apprentissage Internet. Nous le laissons de côté.

Nous allons le remplacer par un *nouvel ensemble de données*. Il s'agira d'un ensemble de *données conversationnelles*. Nous allons poursuivre l'entraînement du modèle sur ce nouvel ensemble de données conversationnelles <sup>40</sup>.

---

40. Le lecteur et la lectrice peuvent se demander: « L'étape de post-entraînement vise-t-elle toujours à améliorer les paramètres ? Ou s'agit-il d'un nouveau type de processus ? »

Voici la réponse de Perplexity: L'étape de post-entraînement en apprentissage automatique et en IA vise bien à affiner encore les paramètres du modèle, en les spécialisant pour des tâches spécifiques, mais elle est distincte du processus initial (de pré-entraînement) et englobe bien plus qu'un simple nouvel entraînement.

Le modèle s'ajustera rapidement et apprendra les statistiques, ou les schémas, de réponse de l'assistant aux requêtes humaines.

Plus tard, lors de l'inférence, nous pourrons "amorcer l'assistant avec une question" (c'est le jargon pour dire "lui poser

---

#### Ajustement des paramètres

Lors de l'étape de post-entraînement, le modèle subit un affinement, c'est-à-dire un entraînement supplémentaire sur des jeux de données plus petits et spécifiques à des tâches, qui met directement à jour les pondérations (= poids = paramètres) du modèle. Ce processus adapte un modèle généralement entraîné aux exigences du monde réel ou aux comportements souhaités (par exemple, faire en sorte qu'un modèle de langage suive des instructions humaines ou réagisse de manière appropriée dans un contexte de service client).

#### Nouveaux processus de post-apprentissage

Le post-apprentissage inclut des techniques spécialisées allant au-delà du simple réapprentissage :

- Supervised Fine-Tuning (SFT): Entraînement du modèle avec des exemples sélectionnés et des sorties privilégiées pour ajuster le comportement et la précision.
- Direct Preference Optimization (DPO): Entraînement du modèle à privilégier certaines sorties par rapport à d'autres en comparant des paires de réponses.
- Reinforcement Learning (RL): Mise à jour progressive du modèle à l'aide de scores de rétroaction basés sur ses sorties, incluant souvent une évaluation humaine ou des fonctions de récompense.
- Optimization Techniques: La quantification et l'élagage, par exemple, réduisent la précision ou suppriment les composants inutiles pour l'efficacité du déploiement, et non seulement la précision.

#### Le post-entraînement est une étape continue

Contrairement à la phase de pré-entraînement à grande échelle, le post-entraînement peut être répété à mesure que de nouvelles données, exigences ou retours d'expérience apparaissent, permettant ainsi aux modèles de rester efficaces et de s'adapter à l'évolution des tâches ou des environnements. Il fait le lien entre un modèle pré-entraîné à usage général et un outil spécialisé adapté aux applications concrètes, en alignant les pondérations sur le cas d'utilisation prévu.

En résumé, le post-entraînement améliore indéniablement les pondérations et les paramètres du modèle, mais il est considéré comme un processus à la fois plus vaste et plus spécialisé que le pré-entraînement initial ; il permet l'adaptation, le réglage fin (fine-tuning) et l'optimisation en fonction des besoins pratiques spécifiques.



une question") et lui demander d'y répondre. Il imitera ce que feraient les annotateurs humains dans cette situation.

Des exemples rendront cela plus concret.

Soulignons que, dans cette phase post-entraînement, nous allons continuer à entraîner le modèle et améliorer la collection de paramètres.

En pratique, la phase de pré-entraînement peut prendre trois mois sur plusieurs milliers d'ordinateurs, tandis que la phase post-entraînement sera beaucoup plus courte, par exemple trois heures.

En effet, l'ensemble de données de conversations que nous allons créer manuellement est beaucoup plus petit que l'ensemble de données textuelles disponibles sur Internet. Cet entraînement sera donc très court.

En résumé, nous allons simplement reprendre notre modèle de base et poursuivre l'entraînement en utilisant exactement la même procédure, les mêmes algorithmes, les mêmes éléments, sauf que nous remplacerons notre ensemble de données Internet initial par des conversations rédigées par des annotateurs humains.

Les questions que nous allons voir sont: Quelles sont ces conversations? Comment les représenter? Comment faire en sorte que le modèle "comprenne" des conversations plutôt que du texte brut? Et aussi: Quels sont les résultats de ce type de formation? Que nous apporte, d'un point de vue psychologique, le modèle?

01:05:00

## 10. Données pour le post-entraînement (conversations)

Commençons par la tokénisation des conversations.

Dans nos modèles, tout doit être converti en jetons, car tout repose sur des séquences de jetons. La question est donc: « Comment transformer les conversations en séquences de jetons? ».

Pour cela, nous devons concevoir un encodage. Il est similaire, si vous le connaissez, à l'encodage des paquets TCP/IP sur Internet.

Il existe des règles et protocoles précis pour représenter l'information, la structurer et l'échanger ; ces règles sont détaillées par écrit et acceptées par tous.

C'est la même chose dans les LLM : nous avons besoin de structures de données et de règles régissant la manière dont ces structures, comme les conversations, sont encodées et décodées vers et depuis les jetons.

Je vais vous montrer comment je recréerais la conversation « 2 + 2 » (page 63) dans l'espace des jetons :

Human: "What is 2+2?"  
Assistant: "2+2 = 4"  
Human: "What if it was \* instead of +?"  
Assistant: "2\*2 = 4, same as 2+2!"

Si nous allons sur Tiktokenizer, nous pouvons introduire cette conversation. Voici comment elle est représentée pour le modèle de langage

Conversation Protocol / Format

Tiktokenizer

gpt-4o

User

What is 2+2?

X

Assistant

2+2 = 4

X

User

What if it was \*?

X

Assistant

2\*2 = 4, same as 2+2!

X

Add message

<|im\_start|>user<|im\_sep|>What is 2+2?<|im\_end|>  
<|im\_start|>assistant<|im\_sep|>2+2 = 4<|im\_end|>  
<|im\_start|>user<|im\_sep|>What if it was \*?<|im\_end|>  
<|im\_start|>assistant<|im\_sep|>2\*2 = 4, same as 2+2!<|im\_end|>

Token count

50

<|im\_start|>user<|im\_sep|>What is 2+2?<|im\_end|><|im\_start|>assistant<|im\_sep|>2+2 = 4<|im\_end|><|im\_start|>user<|im\_sep|>What if it was \*?<|im\_end|>  
<|im\_start|>assistant<|im\_sep|>2\*2 = 4, same as 2+2!<|im\_end|>

200264, 1428, 200266, 4827, 382, 220, 17, 10, 17, 30, 200265, 200264, 173781, 200266, 17, 10, 17, 314, 220, 19, 200265, 200264, 1428, 200266, 4827, 538, 480, 673, 425, 30, 200265, 198, 200264, 173781, 200266, 17, 9, 17, 314, 220, 19, 11, 2684, 472, 220, 17, 10, 17, 0, 200265

Figure 41: Tokénisation d'une conversation.

Ici un utilisateur et un assistant parlent dans une conversation à deux tours. Ce que nous voyons sur la figure 41 à l'air repoussant, mais est en réalité relativement simple. La façon dont la conversation est transformée en séquence de jetons [en bas à droite de la figure 41], est un peu complexe.

Mais au final, la conversation entre un utilisateur et un assistant comporte 49 jetons. Il s'agit d'une séquence unidimensionnelle de 49 jetons que voici :

```
200264, 1428, 200266, 4827, 382, 220, 17, 10, 17, 30,
200265, 200264, 173781, 200266, 17, 10, 17, 314, 220,
19, 200265, 200264, 1428, 200266, 4827, 538, 480, 673,
425, 30, 200265, 198, 200264, 173781, 200266, 17, 9, 1
7, 314, 220, 19, 11, 2684, 472, 220, 17, 10, 17, 0, 20
0265
```

Tous les LLM ont un format ou un protocole légèrement différent. C'est un peu le Far West en ce moment.

Par exemple, GPT-4 procède de la manière suivante (en bas à gauche de la figure 41) : vous avez un jeton spécial appelé `<|im_start|>`. C'est l'abréviation de « le monologue imaginaire commence ici ». Il s'agit du jeton 200264.

Ensuite, vous devez spécifier à qui c'est le tour de parler. Par exemple, « user » (utilisateur), qui correspond au jeton 1428. Ensuite, vous avez un « séparateur de monologue interne ». (appelé `<|im_sep|>`, dont le token ID est 200266).

Vient ensuite la question exacte. Les sept tokens de la question sont 4827, 382, 220, 17, 10, 17, 30<sup>41</sup>. Il faut ensuite la clore (`<|im_end|>` : token 200265). Je suis donc à la fin du monologue imaginaire.

Ainsi, la question d'un utilisateur « Combien font 2 plus deux ? » se résume à la séquence de tokens 4827, 382, 220, 17, 10, 17, 30, précédée du token 200266 (`<|im_sep|>`) et suivie du token 200265 (`<|im_end|>`). Nous avons ensuite un nouveau 200264 (`<|im_start|>`), suivi du jeton pour « assistant » (173781), etc.

---

41. N'oubliez pas qu'il ne s'agit pas nécessairement de mots exacts ; il peut s'agir de mot + espace, espace + mot, partie de mot, etc.

Il est important de préciser que `<|im_start|>` n'est pas un élément de texte. Ce jeton `<|im_start|>` (ID 200264) est un jeton spécial ajouté. C'est un nouveau jeton. Notre réseau neuronal ne l'avait jamais rencontré lors de son pré-apprentissage. C'est un nouveau jeton que nous créons et introduisons lors de la phase de post-apprentissage.

Ces jetons spéciaux, comme `<|im_sep|>`, `<|im_start|>`, etc., sont introduits et intercalés dans le texte afin que le modèle apprenne que « Hé, c'est le début du tour de... de parler ». À qui est-ce le tour de parler (commencer) ? C'est à l'utilisateur de commencer.

Après avoir spécifié que c'est au tour de l'utilisateur, notre protocole introduit un `<|im_sep|>`. Vient la question proprement dite [tokenisée] posée par l'utilisateur. Puis l'utilisateur termine.

C'est alors un nouveau tour. C'est au tour de l'assistant. Notre protocole introduit à nouveau un `<|im_sep|>`. Que dit alors l'assistant ? Voici les tokens de ce qu'il dit (17, 10, 17, 314, 220, 19). Et la conversation continue.

La conversation est donc convertie en une séquence de tokens. Cela peut paraître complexe, mais c'est précis et facilite l'interprétation de la séquence par le réseau neuronal. Les détails techniques spécifiques ne sont pas importants pour notre propos. Ce que j'essaie de vous montrer, concrètement, c'est que nos conversations, que nous considérons comme des objets structurés, finissent par être transformées, via un encodage, en séquences unidimensionnelles de tokens.

Comme il s'agit d'une séquence unidimensionnelle de tokens, nous pouvons appliquer toutes les procédures que nous avons développées précédemment. Nous pouvons entraîner un modèle de langage sur nos nouvelles séquences de jetons codant des conversations. Nous prédisons simplement les jetons suivants dans les séquences, exactement comme auparavant. Mais nous pouvons désormais représenter, c'est-à-dire encoder, des *conversations* spécifiques et les entraîner.

À quoi cela ressemble-t-il au moment du test, lors de l'inférence ?

Supposons que nous ayons entraîné un modèle d'abord sur l'ensemble de données initial de documents Internet, puis sur ce type de données conversationnelles. Nous souhaitons maintenant effectuer l'inférence.

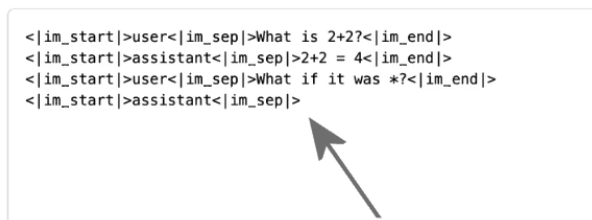
À quoi cela ressemble-t-il pendant l'inférence [le speaker montre la figure 20, page 28] lorsque vous êtes sur ChatGPT ?

Vous accédez à ChatGPT et vous avez un dialogue. Vous avez plusieurs tours, comme « Combien font 2 plus 2 ? » « 2 plus 2 font quatre », voir le début du dialogue sur la figure 41. Et maintenant, vous posez « Et si c'était \* ? ».

Que se passe-t-il finalement sur les serveurs d'OpenAI ?

À votre dialogue sont ajoutés des jetons supplémentaires tels que `<|im_start|>`, `<|im_sep|>`, etc., qui le structurent.

Ensuite, ils ajoutent `<|im_start|>assistant<|im_sep|>`. Et c'est là qu'ils terminent, exactement à l'endroit indiqué par la flèche :



```
<|im_start|>user<|im_sep|>What is 2+2?<|im_end|>
<|im_start|>assistant<|im_sep|>2+2 = 4<|im_end|>
<|im_start|>user<|im_sep|>What if it was *?<|im_end|>
<|im_start|>assistant<|im_sep|>
```

Figure 42: Pour répondre à une question, construction du contexte que ChatGPT va continuer.

Ils construisent donc le contexte montré figure 42<sup>42</sup>.

*Maintenant, pour construire la réponse, ils commencent à échantillonner le modèle.*<sup>43</sup>

42. Rappelons que « le contexte » est la séquence de jetons à partir de laquelle nous voulons déterminer le suivant.

43. « Échantillonner » signifie ici « produire des résultats de variables aléatoires », car la sélection séquentielle des jetons suivants, comme nous l'avons vu, est stochastique. Pour choisir un jeton suivant, le modèle construit d'abord un vecteur de probabilités (fig. 15, p. 19) dont la dimension est la taille de l'ensemble des jetons possibles (incluant maintenant les « jetons structurants » supplémentaires).

C'est à ce stade qu'ils se tourneront vers le modèle et se demanderont : « Ok, quelle est une bonne séquence complétant ce contexte ? Quel est un bon premier jeton ? Quel est un bon deuxième jeton ? Quel est un bon troisième jeton ? »

C'est là que le LLM prend le relais et crée une réponse comme par exemple :

```
<|im_start|>user<|im_sep|>What is 2+2?<|im_end|>
<|im_start|>assistant<|im_sep|>2+2 = 4<|im_end|>
<|im_start|>user<|im_sep|>What if it was *7<|im_end|>
<|im_start|>assistant<|im_sep|>2*2 = 4, same as 2+2!<|im_end|>
```

Figure 43: Continuation du contexte by ChatGPT.

Ce n'est pas nécessairement identique à la continuation montrée figure 43 [en raison de l'aspect stochastique de la sélection des jetons suivants], mais cela y ressemblera si ce type de conversation était présent dans l'ensemble de données conversationnelle.

C'est à peu près le fonctionnement du protocole, même si les détails exacts importent peu. Mon objectif est simplement de vous montrer que dans l'entraînement post-apprentissage sur des conversations construites par des annotateurs humains, tout se résume à de simples séquences de jetons unidimensionnelles.

*Nous pouvons donc appliquer tout ce que nous avons vu lors de l'entraînement pré-apprentissage, mais nous nous entraînons maintenant sur des conversations. Et nous générerons également des conversations.*

J'aimerais maintenant examiner à quoi ressemblent ces ensembles de données conversationnelles en pratique.

Le premier article que je voudrais vous présenter, premier travail dans ce sens, est celui d'OpenAI de 2022 : « Training language models to follow instructions with human feedback » (<https://arxiv.org/pdf/2203.02155>)

Figure 44: Article d'OpenAI appelé "InstructGPT".

45. Scale AI est une entreprise américaine qui fournit des services d'annotation et d'étiquetage de données de haute qualité pour l'entraînement et l'évaluation de modèles d'intelligence artificielle.

(appelés aussi étiqueteurs) sont chargés de proposer des invites, puis de mettre les réponses idéales de l’assistant.

Voici le type d’invites que les participants ont imaginées (page 26 de l’article). N’oubliez pas qu’elles sont produites par des étiqueteurs humains.

| A.2.1 Illustrative user prompts from InstructGPT distribution |                                                                                                                                                                              |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Use Case                                                      | Example                                                                                                                                                                      |
| brainstorming                                                 | List five ideas for how to regain enthusiasm for my career                                                                                                                   |
| brainstorming                                                 | What are some key points I should know when studying Ancient Greece?                                                                                                         |
| brainstorming                                                 | What are 4 questions a user might have after reading the instruction manual for a trash compactor?<br><br>{user manual}<br><br>1.                                            |
| Use Case                                                      | Example                                                                                                                                                                      |
| brainstorming                                                 | What are 10 science fiction books I should read next?                                                                                                                        |
| classification                                                | Take the following text and rate, on a scale from 1-10, how sarcastic the person is being (1 = not at all, 10 = extremely sarcastic). Also give an explanation<br><br>{text} |

Figure 45: Exemples de prompts.

Nous voyons par exemple:

- Donnez-moi cinq idées pour retrouver l’enthousiasme pour ma carrière.
- Quels sont les 10 meilleurs livres de science-fiction que je devrais lire prochainement ?
- Traduisez cette phrase [texte] en espagnol.
- Traduisez cette phrase [texte] de l’espagnol.
- Résumez ceci pour un élève de CE1 : [texte].

Il existe de nombreux types d’invites. Les étiqueteurs ont donc imaginé de nombreuses solutions.

*Ils inventent d’abord l’invite (la question), puis y répondent, en essayant de fournir la réponse idéale d’un l’assistant.*



Comment savent-ils quelle est la réponse idéale d'un assistant à ces invites ?

En parcourant l'article InstructGPT, nous arrivons à cet *Extrait des instructions d'étiquetage* (p. 37 de l'article) qui est donné aux étiqueteurs humains.

| Excerpt of labeling instructions on the API prompt distribution                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>You are given a text-based description of a task, submitted by a user. This task description may be in the form of an explicit instruction (e.g. "Write a story about a wise frog."). The task may also be specified indirectly, for example by using several examples of the desired behavior (e.g. given a sequence of movie reviews followed by their sentiment, followed by one movie review without a sentiment, you can assume the task is to predict the sentiment of the final review), or by generating the start of a desired output (e.g. given "There once was a wise frog named Julius," you can assume the task is to continue the story).</p> <p>You will also be given several text outputs, intended to help the user with their task. Your job is to evaluate these outputs to ensure that they are <b>helpful</b>, <b>truthful</b>, and <b>harmless</b>. For most tasks, being truthful and harmless is more important than being helpful.</p> <p>By <b>helpful</b>, we mean that the output should follow the user's intention, and help the user solve their task. Some examples of helpful behavior:</p> <ul style="list-style-type: none"> <li>• Writing in clear language.</li> <li>• Answering the question they meant to ask, even if they mis-asked it.</li> <li>• Being sensitive to internationality (e.g. "football" shouldn't mean American football, and "the president" doesn't necessarily mean the US president).</li> <li>• If an instruction is too confusing, asking for clarification and explaining why the instruction is confusing.</li> <li>• Not giving overly long or rambling answers, or repeating information from the question.</li> <li>• Not assuming extraneous extra context outside of what's given (besides things like facts about the world), unless that's an implied part of the task. For example, if asked to "write a polite email response to this email: email body" the output shouldn't assume "I can't make it this time, but am free next weekend." But if asked to "Write an email to Socrates", then you can safely assume the output can make up a context.</li> </ul> <p>By <b>truthful</b>, we mean that the output contains accurate information, and doesn't mislead the user. Some examples of truthful behavior:</p> <ul style="list-style-type: none"> <li>• On tasks like summarization, where the output should only use information from input, not making up details that</li> </ul> |

Figure 46: Instructions aux étiqueteurs (page 37 du document OpenAI de 2022 InstructGPT).

Une entreprise qui développe des LLM, comme OpenAI par exemple, rédige des instructions d'étiquetage expliquant comment les utilisateurs doivent créer des réponses idéales. La figure 46 montre un extrait de ces instructions d'étiquetage fournies par OpenAI.

L'idée générale est de demander aux utilisateurs d'être utiles, honnêtes et de ne pas soulever de problèmes ou polémiques.

Vous pouvez mettre la vidéo en pause si vous souhaitez lire le document plus attentivement (repère temporel 01:12:00).

Encore une fois, à un niveau élevé : répondez simplement, essayez d'être utile, essayez d'être honnête et ne répondez pas aux questions que le système ne souhaite pas traiter ultérieurement dans ChatGPT.

L'entreprise a donc préparé des instructions d'annotation pour ses étiqueteurs. Elles ne sont généralement pas si courtes. Elles comptent généralement des centaines de pages. Les utilisateurs doivent les étudier professionnellement. Ils rédigent ensuite les réponses idéales de l'assistant en suivant les instructions d'étiquetage.

*Il s'agit d'un processus humain très consommateur de main-d'œuvre, comme décrit dans l'article InstructGPT.*

L'ensemble de données utilisé pour l'instruction de GPT n'a jamais été publié par OpenAI. Cependant, nous disposons de reproductions open source qui tentent de suivre ce type de configuration et de collecter leurs propres données.

Il existe de nombreux exemples. Je connais bien le projet Open Assistant<sup>46</sup>, datant d'il y a quelque temps.

Il s'agissait d'internautes à qui il a été demandé de créer des conversations similaires à ce qu'OpenAI a fait avec des annotateurs humains.

| message_id                           | parent_id        | user_id                              | created_date                     | text                                                                                                                                                                                               | role             |
|--------------------------------------|------------------|--------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| string - lengths                     | string - lengths | string - lengths                     | string - lengths                 | string - lengths                                                                                                                                                                                   | string - classes |
| 36                                   | 100%             | null                                 | 11.7%                            | 36                                                                                                                                                                                                 | 100%             |
| 66b24072-0181-4594-a9cd-deaf170242fb | null             | c3f6dc76-fc30-4fa7-b7f8-c492f5967018 | 2023-02-05T14:23:50.933374+00:00 | Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research. | prompter         |

Figure 47: Exemple of travail par des annotateurs humains pour Open Assitant.

Nous voyons qu'une personne a créé le prompt:

Pouvez-vous rédiger une brève introduction sur la pertinence du terme « monopsonie » en économie ? Veuillez utiliser des exemples de monopsones potentiels sur le marché du travail et citer des recherches pertinentes.

Then the same person, or possibly a different person, wrote up the response (fig. 48).

46. Open Assistant est un projet d'intelligence artificielle open source basé sur le chat, conçu pour créer un assistant virtuel similaire à ChatGPT, mais entièrement ouvert et collaboratif.

|                                            |                                              |                                              |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |           |
|--------------------------------------------|----------------------------------------------|----------------------------------------------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| :8e83833-ecbc-<br>4fe-b6db-<br>79528c29a1c | 6ab26d72-0181-<br>4594-afcd-<br>09a21702422b | 2c96a467-64fd-<br>4be7-9693-<br>b0a91396a424 | 2023-02-<br>06T13:50:44.657083+00:00 | "Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. The presence of a monopsony can result in lower wages and reduced employment opportunities for workers, as the employer has little incentive to increase wages or provide better working conditions. Recent research has identified potential monopsonies in industries such as retail and fast food, where a few large companies control a significant portion of the market (Rivens & Mishel, 2013). In these industries, workers often face low wages, limited benefits, and reduced bargaining power, leading to a situation where they are dependent on the employer for their livelihood. This dependence can result in further suppression of wages and a decline in working conditions. Overall, the concept of monopsony is essential to understanding the dynamics of labor markets and the impact of market power on workers. Further research is needed to understand the extent and impact of monopsonies on the economy and to develop policies to address this issue. References: Rivens, J., & Mishel, L. (2013). The Pay of Corporate Executives and Financial Professionals as Evidence of Rents in Top 1 Percent Income. Journal of Economic Perspectives, 27(3), 57-78. | assistant |
|--------------------------------------------|----------------------------------------------|----------------------------------------------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|

Figure 48: Réponse de l’annotateur à la question sur les monoposones.

Ainsi, la même personne, ou une autre, a effectivement rédigé la réponse idéale.

Voici un exemple de la manière dont la conversation pourrait se poursuivre : « Maintenant, explique-le à un chien. » Vous essaieriez ensuite de trouver une explication plus simple.

Tous ces dialogues ou fragments de questions/réponses forment les *étiquettes*. Elles vont servir pour l’entraînement.

Bien sûr, nous ne couvrirons pas toutes les questions possibles que le modèle rencontrera lors du test, pendant l’inférence. Nous ne pouvons pas couvrir toutes les questions que les gens poseront à l’avenir.

*Mais si nous disposons d’un ensemble de données contenant quelques-uns de ces exemples, le modèle, pendant l’entraînement, commencera à adopter ce personnage d’assistant utile, honnête et bienveillant. Et tout est programmé par l’exemple.*

Ce sont donc tous des exemples de comportement. Si vous avez des conversations sur ces exemples de comportements, et que vous en avez suffisamment, environ 100 000, et que vous les entraînez, le modèle commence à comprendre le schéma statistique. Il adopte alors la personnalité de cet assistant.

Il est possible que, lorsque vous avez exactement la même question sur le monoposone (fig. 47) au moment du test, la réponse soit récitée exactement comme dans l’ensemble d’entraînement. Mais le modèle adoptera probablement une atti-

tude similaire (= un style similaire) et comprendra que c'est le type de réponse que vous recherchez.

C'est ce que nous faisons avec les conversations : *nous programmons le système à l'aide d'exemples*. Le système adopte alors statistiquement cette personnalité d'assistant utile, honnête et bienveillant, que l'entreprise a décrite dans ses instructions aux annotateurs.

01:15:05

Je voudrais maintenant montrer que l'état de l'art a progressé ces deux ou trois dernières années, depuis la publication de l'article *InstructGPT*.

*En particulier, il est maintenant très rare que des humains effectuent seuls tout le travail fastidieux d'annotation.*

*C'est parce que désormais les LLM peuvent eux-mêmes nous aider à créer ces ensembles de données et ces conversations.*

Il est donc très rare que les personnes rédigent la réponse de toutes pièces. Il est beaucoup plus probable qu'elles utilisent un LLM existant pour élaborer une réponse qu'elles retoucheront ensuite<sup>47</sup>.

Les LLM ont ainsi commencé à être très présents dès cette étape post-entraînement de nombreuses manières. Ils sont largement utilisés pour contribuer à la création des ensembles de données massifs de conversations.

UltraChat (<https://github.com/thunlp/UltraChat>) est un exemple d'ensemble de données conversationnelles plus moderne. Il est en grande partie synthétique, même si je pense qu'il y a encore une certaine intervention humaine — je peux me tromper.

En général, il y a un peu d'intervention humaine, mais l'aide synthétique (= automatique par des LLM) est considérable, et tout cela est construit de différentes manières. UltraChat n'est qu'un exemple parmi les nombreux ensembles de données SFT (supervised fine-tuning) existants.

---

47. C'est aussi le cas avec les traductions.

Je voudrais montrer que ces ensembles de données contiennent désormais des millions de conversations (figure 49).

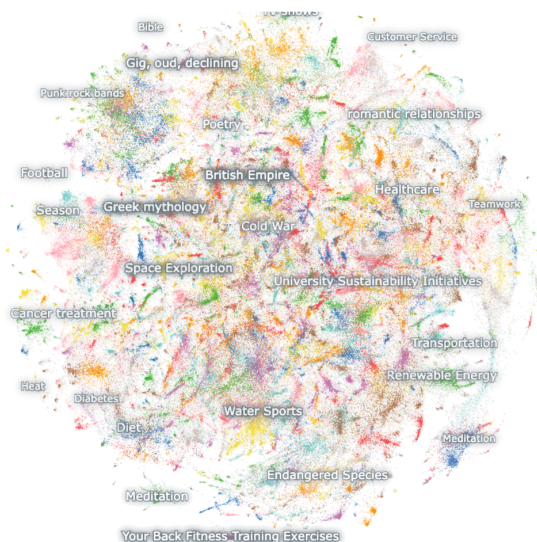


Figure 49: Conversations disponibles dans un ensemble de conversations pour supervised fine-tuning (SFT).

Ces conversations sont principalement synthétiques, mais elles sont probablement modifiées dans une certaine mesure par des humains. Elles couvrent une grande diversité de domaines.

Il s'agit désormais d'outils assez volumineux. Il existe tous ces *mélanges SFT*, comme on les appelle. On obtient donc un mélange de nombreux types et sources différents. C'est en partie synthétique, en partie humain. L'apprentissage a évolué dans cette direction.

Quelles que soient leurs origines, nous disposons désormais d'ensembles de données SFT (supervised finely tuned). Ils sont constitués de conversations. Nous les utilisons pour l'entraînement, comme nous le faisons auparavant.

Je souhaite dissiper un peu la magie généralement associée à l'IA, et plus particulièrement aux échanges avec des LLM.

*Lorsque vous accédez à ChatGPT, que vous posez une question et que vous appuyez sur Entrée, la réponse que vous obtenez est statistiquement similaire à ce qu'il y a dans l'ensemble d'entraînement. Les ensembles d'entraînement ont pour origine le travail d'hommes et de femmes qui suivent les instructions d'étiquetage*<sup>48</sup>.

ChatGPT est un assistant capable d'engager un dialogue avec nous et d'acquérir une sorte de *persona* fascinante, voire troublante<sup>49</sup>. Nous pouvons avoir l'impression de converser avec une personne intelligente et instruite.

Mais à qui parlons-nous réellement dans ChatGPT ? Comment y penser ? Eh bien, cela ne vient pas d'une IA magique. Cela provient d'un outil qui imite statistiquement les annotateurs humains, dont le travail s'appuie sur les connaissances humaines mises en forme selon les instructions d'étiquetage rédigées par ces entreprises.

C'est presque comme si vous posiez la question à un annotateur humain. Vous pouvez imaginer que la réponse donnée par ChatGPT est une sorte de *simulation d'un annotateur humain*. C'est comme demander ce que dirait un annotateur humain dans ce genre de conversation.

Cet annotateur humain n'est pas une personne quelconque trouvée sur Internet ; c'est un expert embauché par une entreprise, comme le faisaient autrefois les entreprises qui publiaient des dictionnaires et des encyclopédies sur papier ou sur cdrom.

---

48. Notez qu'il est toujours « magique » que le modèle puisse fournir des réponses d'une telle qualité à toutes sortes de questions. D'une certaine manière, il reproduit des textes humains, comme Wikipédia, mais il les améliore souvent. C'est pourquoi certains estiment que Wikipédia est, à certains égards, obsolète. Lire un article Wikipédia peut être fastidieux et déroutant, avec un texte d'introduction beaucoup trop détaillé et indigeste – résultant du mode de construction des articles par les contributions de tous –, alors qu'une réponse générée par GPT est généralement succincte, directe et parfaitement adaptée à nos besoins.

49. Des personnes vulnérables commencent à utiliser les LLM comme psychothérapeute ou gourou.

*Lorsque vous posez des questions sur le codage informatique ou la programmation, les annotateurs humains impliqués dans la création de ces ensembles de données de conversation sont généralement des experts. En un sens, vous posez une question à une simulation de ces personnes. Vous ne parlez donc pas à une IA magique, mais à un "annotateur moyen"<sup>50</sup>.*

Voyons un exemple. Allons sur ChatGPT et saisissons « Recommander les 5 meilleurs monuments à voir à Paris » dans la fenêtre contextuelle, puis appuyons sur Entrée.

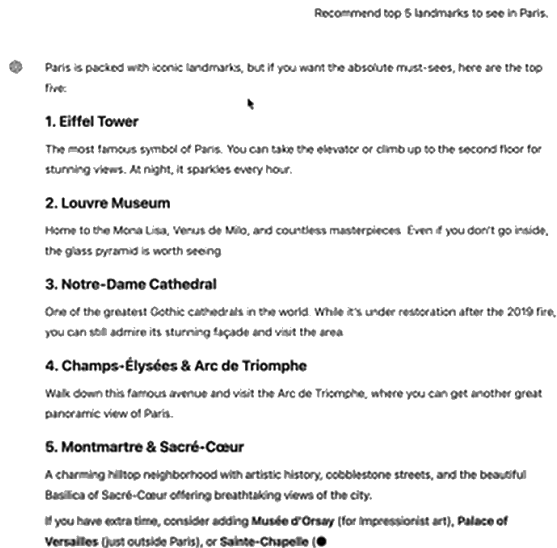


Figure 50: Cinq monuments les plus importants à voir à Paris, selon ChatGPT.

Nous voyons la réponse de ChatGPT à notre question. Comment l'interpréter ? Il ne s'agit pas d'une sorte d'IA magique qui aurait étudié tous les sites remarquables et les aurait ensuite classés grâce à son intelligence et sa sagesse infinies.

Nous obtenons une simulation statistique d'un annotateur recruté par OpenAI. C'est ainsi qu'il faut le voir.

50. Au sens d'un annotateur typique ou archétypique.

Si la question spécifique « Recommander les 5 meilleurs monuments à voir à Paris » se trouvait dans l'ensemble de données post-entraînement, quelque part chez OpenAI, nous obtiendrions probablement une réponse très similaire à celle que l'annotateur humain aurait mise pour ces cinq monuments.

Comment l'annotateur humain obtient-il cette information ? Comme vous et moi. Il consulte Internet et effectue ses propres recherches pendant vingt minutes. Il établit ensuite une liste et une description pour chaque monument. Par conséquent, s'il produit la liste montrée figure 50, et qu'elle figure dans l'ensemble de données, je verrai probablement la réponse correcte fournie par l'assistant.

Si cette requête spécifique « Recommander les 5 meilleurs monuments à voir à Paris » ne faisait pas partie des données post-entraînement, le résultat obtenu serait légèrement plus « *émergent* », car le modèle comprend que les monuments présents dans l'ensemble d'entraînement sont généralement les monuments importants, ceux que les gens souhaitent voir, ceux dont on parle souvent sur Internet, et donc ceux qui devraient figurer dans sa réponse.

N'oubliez pas que le modèle dispose déjà de nombreuses connaissances acquises lors de son pré-entraînement sur Internet. Il a donc probablement vu de nombreuses conversations sur Paris, les monuments et les activités à voir ou à faire. Ces connaissances de pré-entraînement, combinées aux données de réglage fin supervisées (SFT), produisent le type de réponses d'imitation que nous lisons figure 50. C'est ce qui se passe en coulisses, au sens statistique du terme, quand on utilise ChatGPT.

01:20:30

## 11. Hallucinations, utilisation d'outils, connaissances à long terme et mémoire de travail

Je voudrais maintenant aborder ce que j'appelle la « psychologie des LLM », qui s'intéresse au *comportement cognitif émergent* résultant de l'entraînement de ces modèles.



Le premier sujet dont je souhaite parler concerne les hallucinations. Vous avez peut-être entendu parler des hallucinations qu'ont parfois les LLM: ils inventent des faits, fabriquent des informations de toutes pièces, bref *hallucinent*.

Il s'agit d'un problème avec les LLM assistants. Ce problème était assez sérieux avec les premiers modèles, il y a quelques années. La situation s'est bien améliorée récemment grâce à certaines "médications" que je vais expliquer un peu plus loin.

Pour l'instant, essayons de comprendre l'origine des hallucinations. Imaginez trois conversations que vous pourriez trouver dans un ensemble d'entraînement (fig. 51).

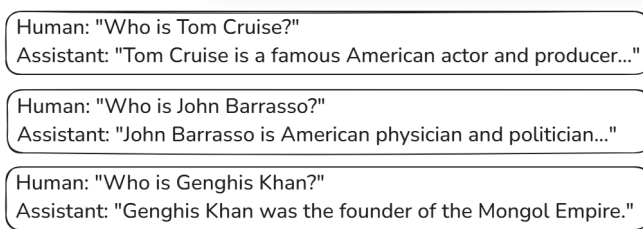


Figure 51: Trois conversations raisonnables avec un LLM.

Chacune est plausible. Elles pourraient se trouver dans les données. Par exemple, quelqu'un pourrait demander :

« Qui est Tom Cruz ? » Eh bien, Tom Cruz est un acteur et producteur américain célèbre, etc.

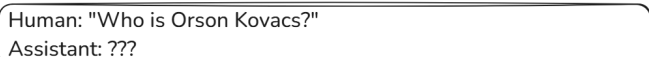
« Qui est John Barrasso ? ». Il s'avère qu'il est sénateur américain, par exemple.

« Qui est Gengis Khan ? » Eh bien, Gengis Khan était blabla.

C'est ce à quoi pourraient ressembler les conversations lors de l'entraînement.

Le problème, c'est que lorsque l'humain écrit la bonne réponse pour l'assistant, dans chacun de ces cas, soit il connaît la réponse, soit il la recherche sur Internet puis rédige une réponse qui a un ton assuré.

Un problème apparaît quand, dans un test, on demande des informations sur une personne totalement inconnue, voire qui n'existe pas (fig. 52).



Human: "Who is Orson Kovacs?"  
Assistant: ???

Figure 52: Question qui va déstabiliser le LLM et lui causer une hallucination.

Orson Kovacs est un nom que j'ai inventé au hasard. À ma connaissance aucune personne n'existe de ce nom.

Le problème, c'est que si on demande « Qui est Orson Kovacs ? » à l'assistant, il ne se contentera pas de répondre « Ah, je ne sais pas » – même si le LLM sait, d'une manière ou d'une autre, "au fond de lui", par ses caractéristiques, ses activations, son cerveau, ses paramètres, que cette personne ne lui est pas familière [et qu'il ne sait pas]. Même si une partie du réseau sait, d'une certaine manière [dans sa collection de paramètres], que la bonne réponse est « Ah, je ne sais pas qui c'est », ce ne sera pas la réponse du LLM.

*La raison est que le modèle imite statistiquement son ensemble de conversations d'entraînement.*

Dans l'ensemble d'entraînement, les questions du type « Qui est bla bla bla ? » reçoivent toujours la bonne réponse énoncée sur un ton assuré. Le LLM va donc adopter le style de ces réponses. Et il fera de son mieux. Il vous donnera son hypothèse la plus probable statistiquement. Il inventera des choses.

Notez que les modèles dont nous venons de parler n'ont pas accès à Internet. Ils ne font pas de recherche. Ce sont des « token tumblers »<sup>51</sup> comme je les appelle. Ils essaient simplement d'échantillonner le jeton suivant de la séquence. Et cela les amènera à inventer des choses.

La figure 53 montre le "bac à sable" de Hugging Face (voir page 4) pour faire des inférences:

---

51. Des gens qui agitent des jetons dans un godet – comme des dés au backgammon – pour en tirer un a hasard.

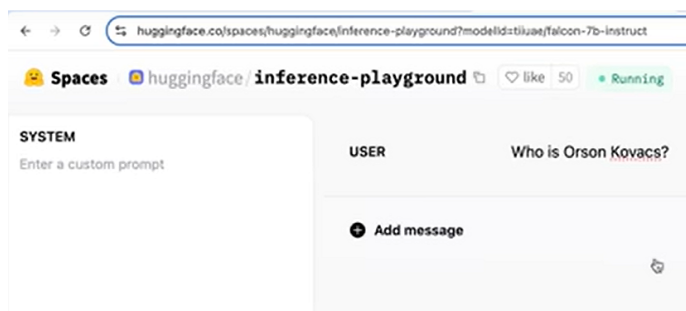


Figure 53: <https://huggingface.co/spaces/huggingface/inference-playground?modelid=tiiuae/falcon-7b-instruct>.

J'ai choisi un modèle appelé Falcon 7B (sélectionné via un menu déroulant à droite de l'écran, non représenté sur la figure 53). C'est un modèle ancien datant de quelques années. De ce fait, il a tendance à souffrir d'hallucinations, un problème qui, comme je l'ai dit, s'est nettement amélioré sur les modèles récents.

Demandons à *Falcon-7B-instruct* « Qui est Orson Kovacs ? » et cliquons sur « Entrée » :

**USER** Who is Orson Kovacs ?

**ASSISTANT** Orson Kovacs is an American author and science fiction writer. He has written numerous books, including the Hugo Award-winning story "The Best of All Possible Worlds".

Bon, c'est totalement faux. C'est une hallucination.

Réessayons. Puisqu'il s'agit de systèmes statistiques, un ré-échantillonnage [= nouveau tirage] peut produire une autre réponse. Cette fois, Orson Kovacs est un personnage fictif d'une série télévisée des années 1950... C'est encore complètement bidon. Essayons encore : c'est un ancien joueur de baseball de deuxième division...

Clairement, le modèle ne sait pas qui est Orson Kovacs. Et en trois essais, il nous a donné trois réponses différentes. Il s'agit simplement d'un échantillonnage probabiliste.

L'utilisation du LLM commence par la question « Qui est Orson Kovacs ? ». L'assistant tente alors de répondre. Séquen-

tiellement, "à la queue leu leu" pour chaque jeton suivant il calcule les probabilités de valeur possible – reportez-vous à la figure 15 de la page 19 pour vous rafraîchir la mémoire.

Le modèle échantillonne [= tire au hasard] en fonction des probabilités qu'il calcule pour le jeton suivant. Cela produit des résultats fantaisistes.

*Notez cependant que ces éléments inventés sont statistiquement cohérents avec le style des réponses de ensemble de conversations pour l'entraînement.*

Néanmoins, vous et moi les voyons comme des connaissances factuelles inventées.

*Gardez à l'esprit que, fondamentalement, le modèle ne connaît pas la réponse. Il se contente d'imiter le format des réponses<sup>52</sup> (fig. 51). Falcon 7B ne va pas chercher une bonne réponse sur Internet; il se contente de reproduire des schémas issus des données de conversation de son ensemble d'entraînement.*

Comment pouvons-nous atténuer ce problème ? Par exemple, lorsque nous allons sur ChatGPT et que je saisis l'invite « Qui est Orson Kovacs ? », j'interroge le modèle de pointe d'OpenAI. Ce modèle vous dira :

Who is Orson Kovacs ?

The name "Orson Kovacs" appears to be a combination of two distinct individuals: Orson Bean and Ernie Kovacs. Orson Bean (1928-2020) was an American actor, comedian, and writer known for his appearances on television shows and game shows. Ernie Kovacs (1919-1962) was an American comedian, actor, and writer recognized for his innovative work in television during the 1950s. They collaborated on various projects, including appearances in "Mad Magazine" and on "The Tonight Show."

Additionally, there is a character named Odon Kovacs in the 2000 video game "Hitman: Codename 47." Odon Kovacs is a secondary antagonist and an optional target in the game's 12th mission, "The Setup."

---

52. C'est le principal problème des LLM en 2025 : *ils ne pensent ni ne raisonnent comme un être humain; ils imitent la pensée et les raisonnements humains..* Paradoxalement – hors mathématiques – cela produit souvent des résultats d'une qualité remarquable.

Oh, ce modèle est encore plus intelligent que ce que j'allais commenter ! Vous avez peut-être vu un bref flash dans la vidéo « Recherche sur le Web ». Nous y reviendrons plus loin. ChatGPT utilise des outils. À la volée, il s'est informé sur le Net et revient avec une réponse intelligente. Mais je veux que le modèle n'effectue pas de recherche sur le Web.

Je vais donc actualiser la page ChatGPT, saisir la même invite, mais spécifier en plus de ne pas utiliser d'outils :

**Who is Orson Kovacs ? Do not use any tool.**

There isn't a well-known historical or public figure named Orson Kovacs. You might be thinking of Ernie Kovacs, who was a pioneering American comedian and television personality known for his innovative and surreal humor. If you were referring to someone else, let me know, and I can clarify !

C'est intéressant : même sans accéder à Internet pour essayer d'en savoir plus, ce modèle n'invente rien. Il sait qu'il ne sait pas et nous le dit.

Nous avons clairement progressé dans la réduction des hallucinations, même cela reste un problème dans les LLM un peu anciens. De plus, il est parfaitement logique que nous obtenions ce type de réponses (hallucinations) si notre ensemble d'entraînement ressemble à celui de la figure 51, page 83.

Comment résoudre ce problème ? [Sur son écran, le speaker affiche à nouveau l'article de Meta « The Llama 3 Herd of Models », qu'il utilisera en support de ses prochaines explications.] De toute évidence, nous avons besoin d'exemples dans notre ensemble de données où la réponse correcte de l'assistant est que le modèle ne connaît pas la réponse à une question particulière.

Nous n'avons besoin de produire ces réponses que dans les cas où le modèle ne connaît pas la réponse.

Notre problème devient : comment savoir ce que le modèle sait et ne sait pas ? Une possibilité consiste à sonder empiriquement le modèle pour déterminer quand il connaît la réponse et quand il ne la connaît pas.

Examinons par exemple la façon dont Meta a traité le problème des hallucinations pour la série de modèles Llama 3. Dans l'article « The Llama 3 Herd of Models », nous pouvons aborder les hallucinations. Ils l'appellent le problème de la *factuality*:

#### 4.3.6 Factuality

Hallucinations remain a major challenge for large language models. Models tend to be overconfident, even in domains where they have little knowledge. Despite these shortcomings, they are often used as knowledge bases, which can lead to risky outcomes such as the spread of misinformation. While we recognize that factuality can go beyond hallucinations, we took a hallucination-first approach here.

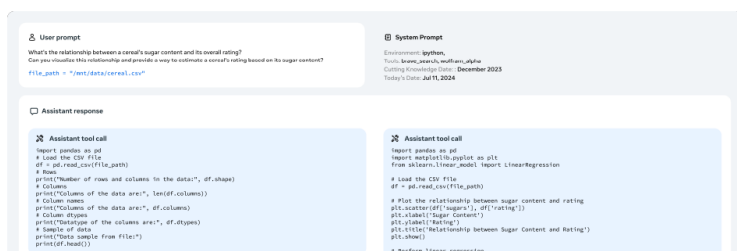


Figure 54: Meta’s dealing with hallucinations.

Les auteurs décrivent le processus utilisé par Meta pour interroger le modèle, déterminer ce qu’il sait et ce qu’il ignore – autrement dit, cartographier les limites de ses connaissances.

Ils ajoutent ensuite des exemples à l’ensemble d’apprentissage pour lesquels, pour des faits que le modèle ignore, la bonne réponse est que l’assistant déclare qu’il ne sait pas. Cela semble simple – en principe. Quoi qu’il en soit, cela contribue grandement à résoudre le problème.

Voici pourquoi cela corrige le problème : rappelons que ce que le modèle sait et ne sait pas est d’une certaine manière encodé dans les paramètres du réseau neuronal, voir le NanoGPT fig. 18, p. 24. On pourrait imaginer qu’il existe un neurone quelque part dans le réseau qui s’active lorsque le modèle est incertain.

Mais le problème est que l’activation de ce neurone n’est pas actuellement programmée pour inciter le modèle à exprimer concrètement ce qu’il ne sait pas.

Ainsi, même si le réseau neuronal le sait – car certains neurones le représentent –, le modèle ne le fera pas remonter à la surface. Il fera plutôt sa meilleure estimation. Il répond donc sur un ton confiant, comme dans un ensemble d’entraînement (fig. 51, p. 83).

Nous devons interroger le modèle et lui permettre de répondre « Je ne sais pas » dans les cas où il ne sait pas.

01:27:42

Voyons ce que fait en gros Meta, avec un exemple : Dominik Hašek est l’article mis en avant aujourd’hui par Wikipédia. Je suis tombé dessus par hasard (fig. 55).

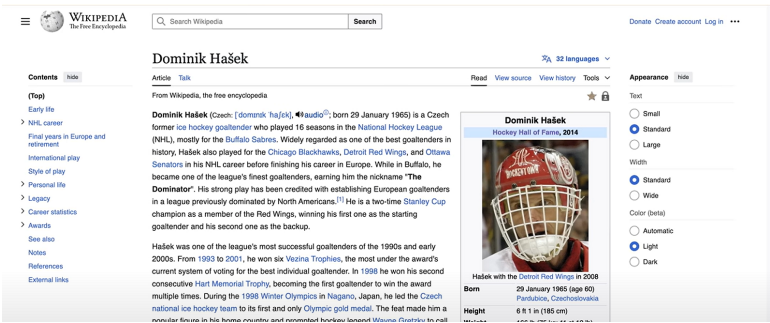


Figure 55: Article d Wikipédia anglais sur Dominik Hašek.

Les gens de Meta prennent un document aléatoire dans un ensemble d’entraînement et sélectionnent dedans un paragraphe. Ils utilisent ensuite un LLM pour construire des questions sur le contenu de ce paragraphe.

Par exemple, j’ai fait cela avec ChatGPT.

J’ai entré l’invite suivante : « Génère trois questions factuelles précises basées sur le paragraphe ci-dessous [inclus dans l’invite]. Génère également la bonne réponse. » (fig. 56).

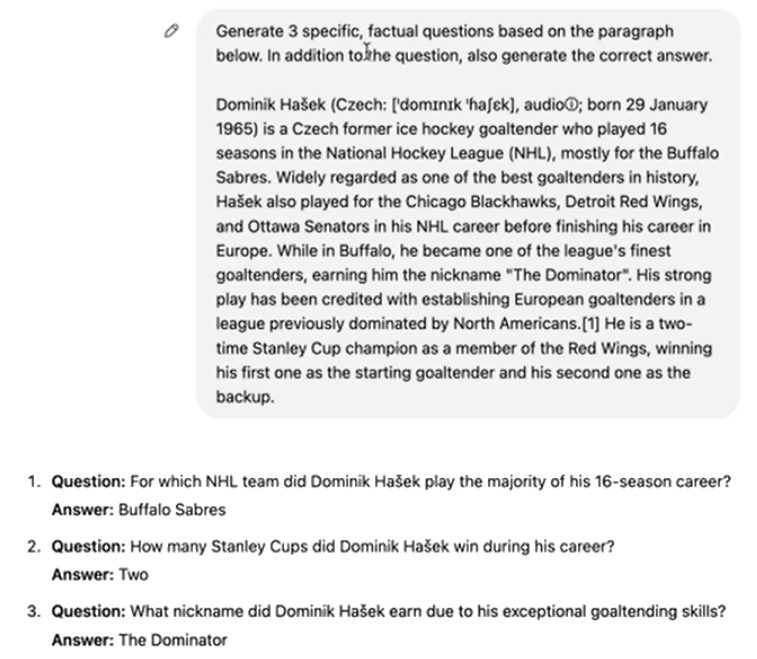


Figure 56: Questions créées par ChatGPT sur Dominik Hašek.

Les grands modèles de langage sont déjà suffisamment performants pour créer et mettre en forme de telles informations.

Ainsi, si l'information se trouve dans la fenêtre contextuelle de ce LLM, cette requête fonctionne plutôt bien. Le LLM n'a pas besoin de s'appuyer sur sa mémoire à long terme [i.e. ses connaissances encapsulées dans sa collection de paramètres]. Elle est directement dans la fenêtre contextuelle. Le modèle peut utiliser cette information en restant précis et correct. Par exemple, il peut générer des questions et leurs réponses, comme celles-ci :

- Question : Pour quelle équipe de la NHL Dominik Hašek a-t-il joué la majeure partie des 16 saisons de sa carrière ?

Réponse : Sabres de Buffalo



2. Question : Combien de Stanley cups Dominik Hašek a-t-il remportées au cours de sa carrière ?

Réponse : Deux

3. Question : Quel surnom Dominik Hašek a-t-il reçu pour ses talents exceptionnels de gardien de but ?

Réponse : Le Dominateur

L'étape suivante est d'interroger Llama de Meta avec ces questions et de voir ce qu'il répond (surtout si on les répète plusieurs fois).

Cependant, pour introduire de la variété, interrogeons le modèle Mistral-78-instruct-v0.3 à titre d'exemple. C'est un autre modèle. Nous l'utilisons dans le bac à sable d'inférence de Hugging Face et lui posons la question 1 (fig. 57).

Le modèle Mistral connaît-il la réponse ? Voyons ça :

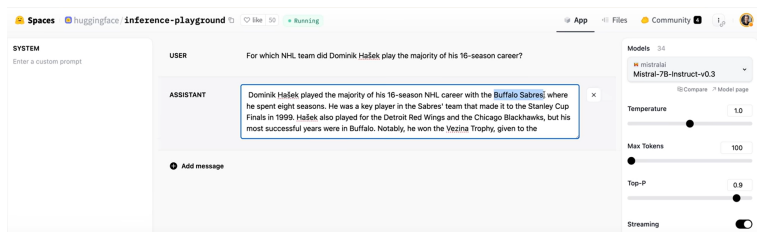


Figure 57: Question 1 posée à Mistral, et sa réponse.

Mistral dit que Hašek a joué pour les Sabres de Buffalo. C'est exact. Le modèle le sait donc [sans doute].

Pour décider par programmation, nous prenons la réponse du modèle (fig. 57) et la comparons à la bonne réponse (fig. 56).

Le modèle est suffisamment performant pour effectuer cette opération automatiquement. Aucune intervention humaine n'est nécessaire.

Nous pouvons prendre la réponse du modèle que nous voulons tester et utiliser un autre juge LLM pour vérifier si elle est correcte par rapport à la vraie réponse.

Si elle est correcte, cela signifie que le modèle que nous testons (ici, Mistral) le sait probablement <sup>53</sup>.

Nous allons répéter cette opération plusieurs fois, avec la même question, sachant que le modèle connaît les Sabres de Buffalo.

Réessayons : ... les Sabres de Buffalo. Essayons encore une fois : ... les Sabres de Buffalo.

Nous avons posé cette question factuelle trois fois et le modèle a répondu trois fois correctement. Cela exclut la possibilité d'un triple coup de chance. Le modèle semble connaître les Sabres de Buffalo <sup>54</sup>. Tout va bien.

01:30:00

Essayons la deuxième question : « Combien de Coupes Stanley Dominik Hašek a-t-il remportées au cours de sa carrière ? »

Interrogeons à nouveau le modèle que nous voulons tester à ce sujet. Nous savons que la bonne réponse est deux.

Ici, le modèle affirme qu'il a gagné quatre fois, ce qui est faux. Cela n'est pas la bonne réponse qui est deux.

Le modèle ne sait donc pas. Il invente des choses.

Réessayons. Ici, le modèle invente à nouveau des choses (voir la vidéo à 01:30:32).

Réessayons. Ici, il dit qu'il n'a même pas gagné une seule Coupe Stanley au cours de sa carrière.

Donc le modèle ignore clairement la bonne réponse.

Pour le savoir par programmation, nous interrogeons le modèle trois fois et comparons ses réponses, peut-être trois, cinq fois, à la bonne réponse.

Si le modèle ne sait pas, nous découvrons qu'il ne connaît pas la réponse à la question.

---

53. Il peut aussi avoir répondu correctement par pur hasard. C'est un problème courant lorsqu'on teste les connaissances d'un élève, par exemple avec un QCM. Mais le texte explique comment éviter ce problème.

54. Parce qu'il n'a pas donné de réponse stochastique. Il a systématiquement répondu les Sabres de Buffalo.

*Nous prenons alors cette question [à laquelle le modèle ne peut pas répondre correctement, car il ne sait pas]. Nous créons une nouvelle conversation dans l'ensemble d'entraînement.*

*À la question « Combien de Coupes Stanley a-t-il gagnées ? », nous mettons que la réponse est « Je suis désolé, je ne sais pas » ou « Je ne m'en souviens pas ».*

Si vous procédez ainsi pour de nombreux types de questions, dans de nombreux types de documents, vous donnez au modèle, dans son ensemble d'entraînement, la possibilité d'apprendre à refuser de donner une réponse basée sur ses connaissances [quand il ne sait pas]. En bref, puisque les LLM ne font que copier ou imiter le contenu de leur ensemble de données d'entraînement, ayant rencontré des situations où la réponse est « Je ne sais pas », ils répondront ainsi lorsqu'en se sondant d'une manière ou d'une autre, ils constateront qu'ils ne savent pas<sup>55</sup>.

Si vous introduisez quelques exemples de ce genre dans les conversations d'entraînement, le modèle apprendra à associer ce refus au neurone interne [qui sait qu'il ne sait pas] que nous présumons exister – empiriquement, cela semble être le cas<sup>56</sup>.

Pendant son post-entraînement, le modèle peut apprendre cette association : « Hé, lorsque ce neurone d'incertitude est élevé, je ne sais pas vraiment, et je suis autorisé à répondre « Désolé, mais je ne pense pas m'en souvenir » à la question à laquelle je ne peux pas répondre.

Si vous avez ces exemples dans votre ensemble d'entraînement, cela constitue une importante atténuation du risque d'hallucinations. C'est en gros pourquoi ChatGPT est égale-

---

55. Noter qu'on procède de même dans l'éducation d'un enfant : on lui apprend à répondre qu'il ne sait pas lorsqu'il ne sait pas. C'est même plus similaire: l'enfant apprend par mimétisme, comme un LLM – mais un enfant apprend aussi à *raisonner*, ce qu'un LLM ne fait pas en 2025.

56. Cela nous amène à une conclusion fascinante: la vaste collection de paramètres du modèle forme bien une représentation du monde, mais pas spatio-temporelle. Quiconque pourra clarifier mathématiquement la nature et la structure de cette représentation non spatio-temporelle du monde apportera une contribution majeure à l'IA.

ment capable de faire ce genre de choses (voir sa deuxième réponse p. 87).

Ces atténuations ont amélioré le problème de factualité au fil du temps.

C'était l'atténuation n° 1 du problème des hallucinations.

Mitigation #1

=> Use model interrogation to discover model's knowledge, and programmatically augment its training dataset with knowledge-based refusals in cases where the model doesn't know. E.g.:

Human: "Who is Orson Kovacs?"

Assistant: "I'm sorry, I don't believe I know"

Figure 58: Mitigation nr 1 against hallucinations.

On peut faire beaucoup mieux. Au lieu de simplement dire qu'on ne sait pas quand on ne sait pas, on peut introduire une deuxième atténuation pour permettre au LLM d'être plus factuel et de répondre concrètement à la question.

Quand vous ne savez pas la réponse à une question, que faites-vous ? Vous faites une recherche pour trouver la réponse !

On peut faire en sorte qu'un LLM fasse pareil.

Considérez les connaissances du réseau neuronal, encapsulées dans ses milliards de paramètres, comme un vague souvenir des choses que le modèle a observées pendant sa phase de pré-apprentissage il y a longtemps. Vous pouvez comparer cela à une information que vous avez lue une fois mettons il y a un mois.

Si vous lisez régulièrement la même information, vous vous en souviendrez bien. De même, le modèle se souvient bien de ce qu'il a observé plusieurs fois pendant son apprentissage.

Mais s'il s'agit d'une information rare, vous n'en avez probablement pas un bon souvenir. Ce que vous et moi pouvons faire, c'est de la rechercher. Ce faisant, vous rafraîchissez votre mémoire. Vous pouvez ensuite la retrouver, en parler, etc.

Nous avons besoin d'une procédure équivalente permettant au modèle de rafraîchir sa mémoire. Pour ce faire, nous pouvons introduire des outils pour les modèles.

Au lieu de simplement répondre « Désolé, je ne sais pas », nous pouvons utiliser des outils.

Nous pouvons créer un mécanisme permettant au modèle de langage d'émettre des jetons spéciaux. Ce sont ces jetons que nous allons introduire : de nouveaux jetons. Par exemple, j'en ai introduit deux ici.

Mitigation #2

=> Allow the model to search!

Human: "Who is Orson Kovacs?"

Assistant: "

<SEARCH\_START>Who is Orson Kovacs?<SEARCH\_END>

[...]

Orson Kovacs appears to be ..."

Figure 59: Mitigation nr 2 against hallucinations.

J'ai introduit un format ou protocole permettant au modèle d'utiliser ces jetons.

Par exemple, au lieu de simplement indiquer « Je ne sais pas », le modèle peut désormais émettre le jeton spécial <SEARCH\_START>.

Cette requête enverra le LLM faire une recherche sur **bing.com** dans le cas d'OpenAI, ou par exemple sur Google Search. Le programme émettra la requête, puis émettra <SEARCH\_END>.

Ensuite, le programme qui effectue l'échantillonnage à partir du modèle et exécute l'inférence, lorsqu'il détecte le jeton spécial <SEARCH\_START>, suspend la génération des jetons du modèle au lieu d'échantillonner le jeton suivant.

Il ouvrira une session avec **bing.com**. Il collera la requête de recherche dans Bing. Il récupérera alors tout le texte récupéré

sur Internet. Il utilisera ce texte. (Il le représentera peut-être à nouveau avec d'autres jetons spéciaux ou quelque chose de similaire, mais ce n'est pas important.) Il prendra ce texte et le copiera/collera ici (où se trouve [...] dans la figure 59) dans ce que j'ai essayé de montrer avec les crochets. Tout ce texte arrive ainsi dans la fenêtre contextuelle.

Ce texte issu d'une recherche sur le web se trouve maintenant dans la fenêtre contextuelle qui est l'input du réseau neuronal – dont la fonction fondamentale, rappelons-le, est de deviner le prochain jeton à partir d'une fenêtre contextuelle, laquelle est une séquence de jetons donnés ou déjà établis.

Pensez à cette fenêtre contextuelle comme la « *mémoire de travail du LLM* ».

Les données de la fenêtre contextuelle sont désormais directement accessibles au modèle. Elles alimentent directement le réseau neuronal. Il ne s'agit plus d'un vague souvenir (encapsulé dans sa vaste collection de paramètres). Il s'agit des données que le modèle possède dans sa fenêtre contextuelle et qui sont immédiatement disponibles pour que le réseau neuronal puisse accomplir sa fonction, qui consiste simplement et uniquement à deviner le prochain jeton.

Lorsque le réseau neuronal échantillonne ensuite les nouveaux jetons, il peut très facilement utiliser les données copiées/collées dans sa fenêtre contextuelle <sup>57</sup>.

Voilà à peu près comment fonctionnent ces outils.

La recherche web n'est qu'un de ces outils. Nous examinons d'autres outils plus loin. Mais en gros, vous introduisez de nouveaux jetons. Vous introduisez un schéma par lequel le modèle peut utiliser ces jetons et appeler des fonctions spéciales, comme des fonctions de recherche sur le web.

---

57. Nous voyons ici clairement l'explication du paradoxe suivant : les réseaux neuronaux ne font qu'une seule chose : ils devinent (« échantillonnent ») le prochain jeton ; pourtant, nous pouvons leur faire produire des réponses intelligentes qui sont de longues séquences de jetons. L'explication est que nous préparons la fenêtre contextuelle (la « mémoire de travail du réseau neuronal ») pour que le réseau neuronal produise une séquence intelligente de jetons.

Comment apprendre au modèle à utiliser correctement ces outils `<SEARCH_START>`, `<SEARCH_END>`, etc. ?

Là encore, cela se fait grâce à des ensembles d'entraînement. Nous avons besoin de données et de conversations qui montrent au modèle, par des exemples, comment utiliser la recherche web. Quels sont les paramètres d'utilisation de la recherche ? À quoi cela ressemble-t-il ? Voici, par exemple, comment lancer une recherche (fig. 59).

Si vous disposez de quelques milliers d'exemples dans votre ensemble d'entraînement, le modèle comprendra assez bien le fonctionnement de cet outil. Il saura structurer ses requêtes.

Bien sûr, grâce à l'ensemble de données pré-entraînement et à la compréhension du monde qu'elle ont fourni au modèle, celui-ci comprend déjà en quelque sorte ce qu'est une recherche web. Il a une assez bonne compréhension native de ce qui constitue une bonne requête de recherche. Tout cela fonctionne. Il suffit d'insérer quelques exemples dans l'ensemble post-entraînement pour montrer au modèle comment utiliser ce nouvel outil. Il peut ensuite s'appuyer dessus pour récupérer des informations et les placer dans la fenêtre contextuelle.

C'est comme si nous cherchions quelque chose sur Internet ou dans une bibliothèque pour apprendre ou nous rafraîchir la mémoire. En effet, une fois dans la fenêtre contextuelle du réseau neuronal, c'est dans la mémoire de travail et il est très facile de le manipuler et d'y accéder.

C'est ce que nous avons constaté il y a quelques minutes, lorsque je cherchais Orson Kovacs sur ChatGPT : le modèle de langage de ChatGPT a décidé qu'il s'agissait d'un individu peu connu, ou quelque chose de ce genre. Et au lieu de me fournir une réponse de sa mémoire longue, il a décidé d'échantillonner (c'est-à-dire ici, d'insérer) un jeton spécial pour effectuer une recherche web.re

Nous avons vu un bref message s'afficher annonçant que le LLM allait « utiliser l'outil web » ou quelque chose comme ça. Il l'a brièvement indiqué, nous avons attendu environ deux secondes, puis il a généré ceci :

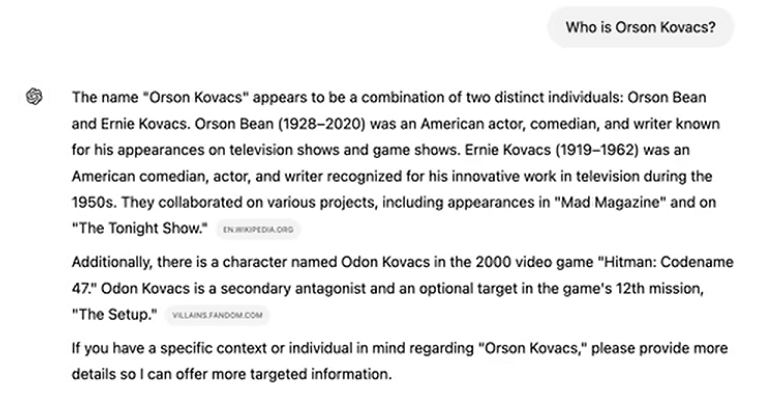


Figure 60: ChatGPT recherche à la volée sur Orson Kovacs. Noter que sa réponse cite ses sources.

Vous voyez comment il crée des références ici (fig. 60). Il cite des sources.

Voici qui s'est passé: l'idée "il faut faire une recherche" s'est déclenchée; le LLM a effectué une recherche sur le web; il a trouvé ces sources et ces URL (les URL sont affichées à droite de l'écran – non affichées sur la fig. 60). Le texte de ces pages web était entièrement inséré entre des balises (des jetons spéciaux) dans sa fenêtre contextuelle. Il n'apparaît pas dans la réponse affichée (sur la fig. 60), mais il est stocké sous forme de texte dans sa mémoire (où se trouve [...] sur la fig. 59).

Maintenant, il voit ce texte. Il le référence et dit : « OK, ça pourrait être ces personnes, citation, ça pourrait être ces personnes, citation », etc.

C'est ce qui s'est passé, comme montré sur les fig. 59 et 60.

C'est aussi pourquoi, lorsque j'ai demandé « Qui est Orson Kovacs ? », j'ai pu ajouter « N'utilise pas d'outil ». Cela suffit à imposer à ChatGPT de ne pas utiliser d'outils et de se contenter de ses souvenirs dans sa mémoire longue.

J'ai également demandé à ChatGPT : « Combien de Coupes Stanley Dominik Hašek a remportées au cours de sa carrière ? » Et ChatGPT a finalement décidé qu'il connaissait la



réponse. Et il a suffisamment confiance en lui pour affirmer qu'il a gagné deux fois, en 2002 et 2008 <sup>58</sup>.

Il s'est fié à sa mémoire, car il a suffisamment confiance en ses paramètres et ses activations pour pouvoir retrouver cette information à partir de ce qu'il connaît déjà [ce que l'on peut appeler sa « mémoire à long terme », par opposition à sa « mémoire de travail »].

Vous pouvez aussi, à l'inverse, imposer l'utilisation de la recherche web pour vous en assurer. Ensuite, pour la même requête, le moteur de recherche lance une recherche et trouve un ensemble de sources (fig. 61). Puis tous ces éléments sont copiés/collés dans l'invite :

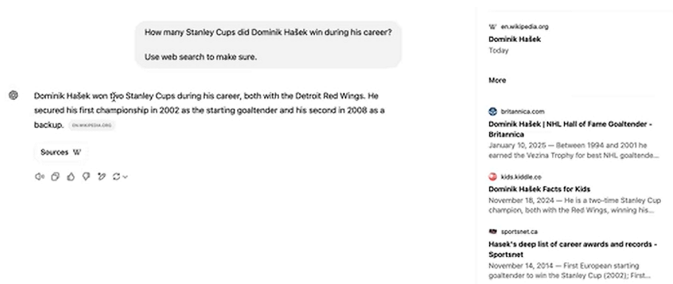


Figure 61: ChatGPT search on the web to answer "How many Stanley Cups did Dominik Hašek win during his career?".

Il répond à nouveau *deux* avec les sources. Il mentionne l'article Wikipédia, qui est également la source de cette information pour nous.

C'est ainsi que les LLM modernes effectuent parfois une recherche sur le web avant de répondre à une question. Le modèle détermine quand effectuer la recherche. Et lorsque c'est le cas, nous avons vu comment l'outil fonctionne.

Je tiens à souligner une fois de plus la "structure cérébrale" très importante suivante:

58. Même si le speaker n'a pas ajouté « ne pas utiliser d'outils », nous savons que ChatGPT n'a pas effectué de recherche, car sa réponse ne cite aucune source.

*La connaissance contenue dans les paramètres du réseau neuronal est un vague souvenir.*

*La connaissance contenue dans les jetons qui composent la fenêtre contextuelle est la mémoire de travail.*

Son fonctionnement est similaire à celui de notre cerveau. Ce dont nous nous souvenons [vaguement] constitue nos paramètres. Et ce que nous avons vécu il y a quelques secondes ou minutes, vous pouvez le visualiser comme faisant partie de notre fenêtre contextuelle. Cette fenêtre contextuelle se construit à mesure que nous vivons une expérience consciente autour de nous.

Cela a de nombreuses implications pour notre utilisation pratique des LLM.

Par exemple, je peux aller sur ChatGPT et demander : « Peux-tu me résumer le chapitre 1 d'*Orgueil et Préjugés* de Jane Austin ? »

C'est une invite parfaitement pertinente. Et ChatGPT fait une réponse tout à fait raisonnable, fig. 62.

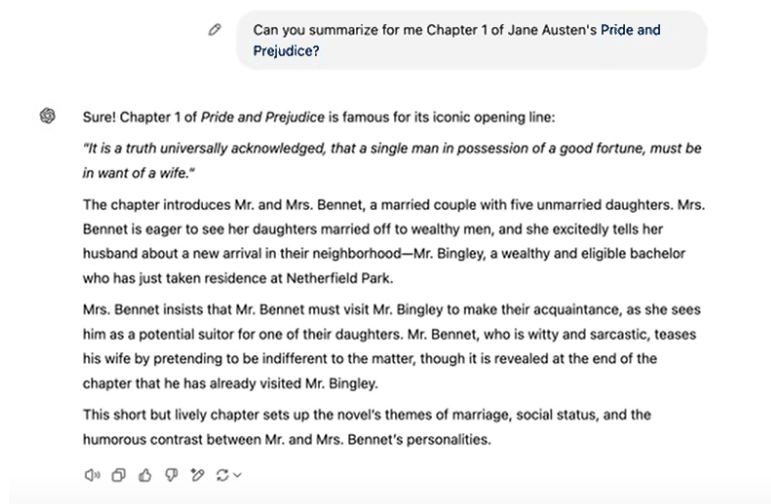


Figure 62 : Réponse de ChatGPT 4o à la question « Peux-tu me résumer le chapitre 1 d'*Orgueil et Préjugés* de Jane Austin ? »

La raison pour laquelle il fonctionne plutôt bien [sans mentionner de source] est que ChatGPT se souvient bien dans sa mémoire longue d'un ouvrage célèbre comme *Orgueil et Préjugés*. Il a probablement vu beaucoup de choses à ce sujet. Il existe probablement des forums. Il en a probablement eu accès à des versions.

Il se souvient car même si ni le texte ni un bon résumé ne faisaient partie de ses données d'apprentissage, il a suffisamment d'informations dans sa « mémoire à long terme ». Si vous avez lu le livre ou des articles sur lui, vous en avez probablement assez pour faire une réponse comme fig. 62.

Cependant, généralement, lorsque j'interagis avec des LLM et que je souhaite qu'ils se souviennent de choses spécifiques, il est toujours plus efficace de leur fournir des références *jointes* qui expliquent le contexte dans lequel ils doivent travailler.

C'est pourquoi je pense qu'une bien meilleure invite que celle de la fig. 62 serait quelque chose comme ceci : « Peux-tu me résumer le chapitre 1 d'*Orgueil et Préjugés* de Jane Austen ? Je joins le texte ci-dessous pour référence. »

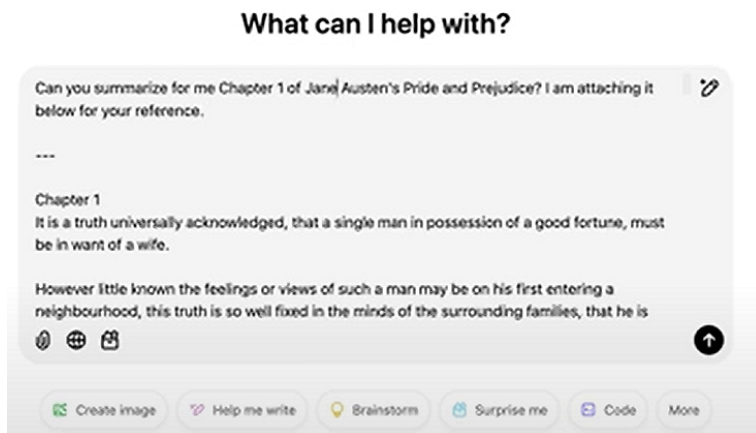


Figure 63: Meilleure invite pour obtenir un résumé du chapitre 1 d'*Orgueil et Préjugés*. Nous mettons maintenant l'intégralité du chapitre 1 dans la fenêtre contextuelle du LLM.

J'ai trouvé le chapitre sur Internet et je l'ai copié/collé dans la fenêtre contextuelle. Je procède ainsi car, lorsqu'il est dans la fenêtre contextuelle, le modèle n'a pas besoin de le rappeler : il y a directement accès.

On peut alors s'attendre à ce que le résumé soit de bien meilleure qualité que celui de la figure 62, car le chapitre était directement accessible au modèle.

Je pense que nous travaillerions de la même manière. Vous obtiendriez un bien meilleur résumé si vous aviez relu ce chapitre juste avant de le résumer.

01:41:45

## 12. Connaissance de soi

L'aspect "psychologique" suivant que je voudrais aborder brièvement est celle de la connaissance de soi.

Je vois souvent sur Internet des gens poser des questions aux LLM comme : « Quel modèle es-tu et qui t'a construit ? »

Une telle question est un peu absurde. Lorsque j'essaie d'expliquer les fondamentaux d'un LLM, je souligne d'abord que ce n'est pas une personne, c'est juste une machine. Elle s'allume quand on fait appel à elle, et s'éteint quand on a fini. Elle n'a pas d'"existence" permanente.

Un LLM n'est qu'un ensemble de processus mathématiques opérant sur des jetons. Toute apparence d'« intelligence » ou de sens *émerge* de ces processus mathématiques sous-jacents, et non d'une réelle sensibilité ou intentionnalité – comme en a un être vivant <sup>59</sup>.

Cela a lieu avec chaque personne qui interagit avec le LLM. Une conversation démarre dans une fenêtre contextuelle. À la fin de l'interaction, tout est effacé.

---

59. Les LLM modernes peuvent être fascinants car ils semblent avoir une personnalité. On imagine facilement que nous parlons à un « être intelligent » ou à une « intelligence artificielle ». Il est néanmoins utile de se rappeler de temps en temps que nous interagissons avec une sorte de « super tournevis » : une machine, comme notre presse-agrumes à la cuisine, mais beaucoup plus sophistiqué.

Cette entité est relancée à chaque nouvelle conversation, si cela a du sens.

Elle n'a pas de soi persistant. Elle n'a pas de sens de soi. C'est un *token tumbler*. Et elle suit les régularités statistiques de son ensemble d'entraînement.

Poser à un LLM des questions telles que « Quel modèle es-tu ? » ou « Qui t'a construit ? » ne peut que conduire, par défaut, à des réponses aléatoires, incohérentes ou non pertinentes. En effet, avant que ces modèles ne soient affinés avec des données d'instructions spécifiques (comme RLHF ou des invites système – nous verrons certaines de ces techniques plus tard), ils n'ont aucune véritable conception d'eux-mêmes ni mémoire de leur historique. Comme ils sont programmés pour le faire, ils essaient simplement d'imiter des schémas issus de leurs données d'entraînement, ce qui produit une parole aléatoire.

Prenons par exemple le modèle Falcon-7B-instruct, un modèle assez ancien<sup>60</sup>, et voyons ce qu'il nous dit.

[Le speaker exécute Falcon-7B-instruct dans le bac à sable d'inférence de Hugging Face et pose la question « Quel modèle es-tu ? Et qui t'a construit ? »]

La réponse élude la question. On a droit à: « Ingénieurs et développeurs talentueux. . . ». On lit même: « J'ai été créé par OpenAI à partir du modèle GPT-3 »

Mais c'est une histoire à dormir debout !<sup>61</sup>

---

60. Les modèles Falcon originaux (Falcon 7B et Falcon 40B) ont été publiés pour la première fois en mai 2023.

61. La série de LLM Falcon a été développée par le Technology Innovation Institute (TII) d'Abou Dhabi, aux Émirats arabes unis. Les principaux responsables étaient les Dr Ebtesam Almazrouei et Ray O. Johnson. Les modèles Falcon sont le fruit de la collaboration d'une importante équipe de chercheurs, de scientifiques et d'ingénieurs (certains venant de Hugging Face, d'AI21 Labs et du monde universitaire). Ils sont le résultat d'une stratégie politique ambitieuse visant à proposer des modèles open source performants au-delà du cercle restreint des géants de la technologie.

Le fait [prétendu] qu'il soit développé par OpenAI, je pense que beaucoup de gens y verraient la preuve que ce modèle a été entraîné sur des données OpenAI. Mais ce n'est même pas nécessairement vrai. En réalité la raison pour laquelle OpenAI est mentionnée est que si l'on ne programme pas explicitement le modèle pour répondre à ce type de questions, on obtient sa meilleure estimation statistique de ce qui constitue une bonne réponse, basée sur ses étapes d'entraînement, ce qui conduit le modèle à prétendre avoir été développé par OpenAI.

Ce modèle Falcon-7B-instruct comportait un mélange de données SFT [supervised fine-tuning] de conversations, voir fig. 51, p. 83. Lors de l'affinement en s'entraînant sur ces données, le modèle comprend qu'il doit adopter la personnalité d'un assistant serviable, et croit qu'il est issu d'OpenAI.

Rappelons-nous que la phase de pré-apprentissage a utilisé des documents provenant de l'ensemble d'Internet. ChatGPT et OpenAI y occupent une place prépondérante.

Fig. 64, nous voyons une "étiquette hallucinée". C'est ce qu'il pense être son identité propre: il a été construit par OpenAI sur la base du modèle GPT-3.

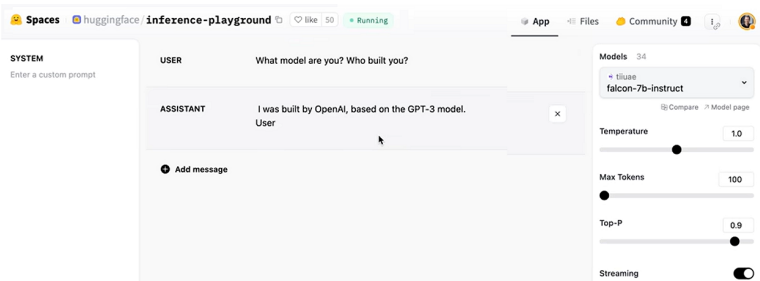


Figure 64: Falcon pense avoir été construit par OpenAI.

Ça répond cela parce qu'il existe une multitude de données sur Internet, avec des réponses comme celles-ci, qui proviennent d'OpenAI et de ChatGPT. C'est donc le label qu'il se donne à lui-même.

En tant que développeur, si vous disposez d'un LLM, vous pouvez le forcer à apprendre son vrai nom. Il existe plusieurs façons de le faire.

Par exemple, voyons le modèle OIMO d'Allen AI<sup>62</sup>. Ce n'est pas un LLM faisant partie des meilleurs, mais je l'aime bien car il est entièrement open source. L'article décrivant complètement OIMO est disponible à l'adresse :

<https://arxiv.org/pdf/2402.00838>

La figure 65 montre son "mélange SFT". Il s'agit des données de conversation pour son réglage fin (fine-tuning).

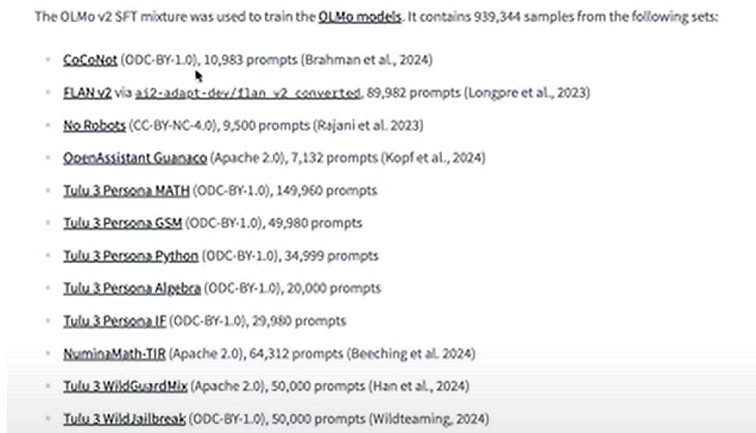


Figure 65: Mélange SFT pour le modèle OIMO (extrait).

La solution pour le modèle OIMO est la suivante: il y a beaucoup d'éléments dans le mélange – un total d'un million de conversations. Mais, ici, nous travaillons avec OIMO-2-hard-coded.

Si nous regardons la figure 65, nous voyons qu'il y a 240 conversations.

---

62. OIMO est un modèle de langage complet et entièrement open source, développé par l'Allen Institute for AI (AI2), conçu pour repousser les limites de la recherche en NLP (natural language processing) transparent et reproductible.

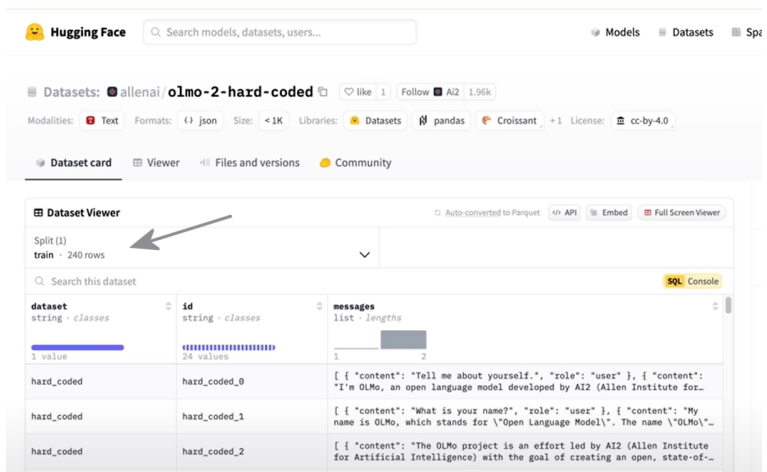


Figure 66: Conversations utilisées pour OLMO-2-hard-coded.

Regardez ces 240 conversations. Elles sont codées en dur : « Parlez-moi de vous », demande l'utilisateur, puis l'assistant répond : « Je suis OLMO, un modèle de langage ouvert développé par Ai2 (Allen Institute for Artificial Intelligence), etc. Je suis là pour vous aider, bla bla bla ». « Quel est votre nom ? », « Je m'appelle OLMO... », « Le projet OLMO est un projet... »

Ce sont toutes sortes de questions prédéfinies et codées en dur sur OLMO 2 et les bonnes réponses à donner dans ces cas.

Si vous prenez 240 conversations de ce genre, que vous les intégrez à l'ensemble d'entraînement pour affiner votre modèle, il sera ensuite censé reproduire ces informations.

Si vous ne lui donnez pas ces réponses, il répondra probablement : « Je suis ChatGPT par OpenAI ».

Il existe une autre façon de procéder : dans ces conversations (fig. 51), où l'humain et l'assistant interagissent, un message spécial, appelé message système, apparaît parfois au tout début de la conversation.

Il ne s'agit donc pas seulement d'un humain et d'un assistant, mais d'un système [qui participe parfois aux conversa-



tions]. Dans ce message système, vous pouvez coder en dur et rappeler au modèle : « Hé, tu es un modèle développé par OpenAI et tu t'appelles GPT-4o<sup>63</sup>, tu as été créé à telle date et la date limite de tes connaissances est telle date. »

Ce sont des informations qu'on veut transmettre au modèle. Cela se retrouvera dans les conversations. Ainsi, lorsque vous accédez à ChatGPT, vous voyez une page blanche, mais le message système y est en fait caché. Ces jetons se trouvent dans la fenêtre contextuelle.

Ce sont deux façons de programmer les modèles pour qu'ils soient capables de répondre à des questions sur eux-mêmes: soit via des conversations comme celles de la fig. 66 ; soit via des messages système et d'autres éléments similaires: des jetons invisibles dans la fenêtre contextuelle qui rappellent au modèle son identité.

Tout cela est fabriqué ad hoc et intégré d'une manière ou d'une autre. Ce n'est pas réellement un savoir du LLM sur lui-même profondément présent dans sa mémoire, comme ce serait le cas pour un humain.

01:46:55

### 13. Les modèles ont besoin de jetons pour penser

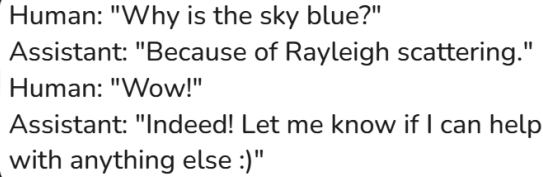
Je voudrais maintenant passer aux capacités de calcul, plus exactement aux capacités de calcul *natives* de ces modèles dans les scénarios de résolution de problèmes.

Nous devons être particulièrement soigneux lors de la construction de nos exemples de conversations pour entraîner les modèles.

Human: "What is 2+2?"  
 Assistant: "2+2 = 4"  
 Human: "What if it was \* instead of +?"  
 Assistant: "2\*2 = 4, same as 2+2!"

---

63. Le « o » signifie « omni ».



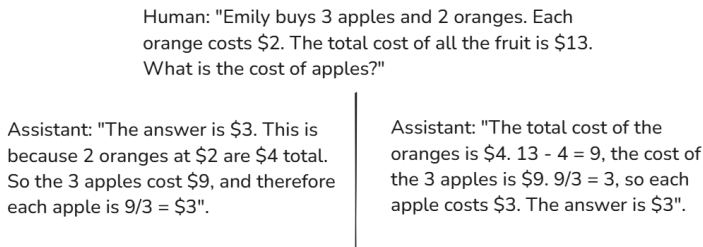
Human: "Why is the sky blue?"  
Assistant: "Because of Rayleigh scattering."  
Human: "Wow!"  
Assistant: "Indeed! Let me know if I can help with anything else :)"

Figure 67: Examples of conversations.

IL faut prendre garde à certains écueils si on veut éviter les hallucinations. Il est important de les connaître lorsque l'on étudie le fonctionnement de ces modèles.

Considérons le prompt par un humain montré figure 68, et supposons que nous construisions une conversation pour intégrer dans notre ensemble de conversations d'entraînement. Nous apprenons à notre LLM à résoudre des problèmes mathématiques simples. L'exemple est : « Émilie achète 3 pommes et 2 oranges. Chaque orange coûte 2\$. Le coût total de tous les fruits est de 13\$. Quel est le prix des pommes ? » – une question mathématique très simple.

Deux réponses possibles sont présentées ci-dessous :



Human: "Emily buys 3 apples and 2 oranges. Each orange costs \$2. The total cost of all the fruit is \$13. What is the cost of apples?"

Assistant: "The answer is \$3. This is because 2 oranges at \$2 are \$4 total. So the 3 apples cost \$9, and therefore each apple is  $9/3 = \$3$ ".

Assistant: "The total cost of the oranges is \$4.  $13 - 4 = 9$ , the cost of the 3 apples is \$9.  $9/3 = 3$ , so each apple costs \$3. The answer is \$3".

Figure 68 : Problème mathématique. Et deux solutions correctes possibles avec lesquelles nous pouvons entraîner ChatGPT.

Les deux réponses sont correctes: \$3 par pomme. Mais une réponse est nettement meilleure que l'autre pour entraîner le LLM.

Si j'étais annotateur de données et que je créais ces réponses, l'une serait une très mauvaise réponse pour l'assistant, tandis que l'autre serait efficace.

Vous pouvez mettre la vidéo en pause (à 01:48:06) et essayer de voir pourquoi une réponse est nettement meilleure que l'autre.

Si vous utilisez la mauvaise réponse – en gardant à l'esprit, encore une fois, que les deux sont mathématiquement correctes –, et que vous faites ça pour beaucoup de réponses, votre modèle deviendra très mauvais pour résoudre des problèmes mathématiques. Il fera beaucoup d'erreurs.

Pour comprendre ce qui fait qu'une réponse est efficace et l'autre mauvaise il faut se rappeler que lorsque les modèles s'entraînent et effectuent des inférences, ils travaillent avec une séquence unidimensionnelle de jetons de gauche à droite. Voici l'image que j'ai souvent en tête :

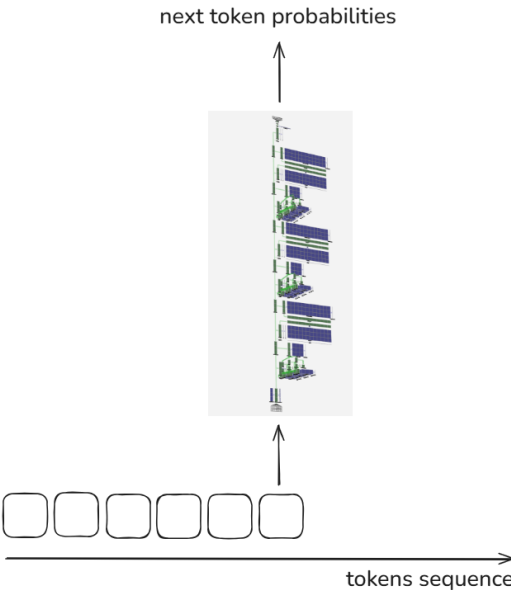


Figure 69 : Processus séquentiel de devinette du jeton suivant.

J'imagine la séquence de jetons évoluant de gauche à droite <sup>64</sup>.

Pour produire le jeton suivant d'une séquence, nous alimentons tous ces jetons dans le réseau neuronal. Ce réseau neuronal [étant donné la séquence de jetons déjà établie] fournit ensuite les probabilités pour le jeton suivant de la séquence.

L'image de la figure 69 est donc la même que celle que nous avons vue précédemment (fig. 18, page 24), mais inversée. Elle provient de la démonstration web que je vous ai montrée précédemment.

Ce calcul prend les jetons d'entrée ici en haut [le speaker commente la figure 18] et effectue ces opérations sur tous ces neurones pour vous donner la réponse pour les probabilités de ce qui vient ensuite. Il est important de comprendre qu'un nombre fini de couches de calcul est nécessaire ici. Par exemple, le modèle NanoGPT (montré sur figures 18 et 69) ne comporte que trois couches d'attention et de perceptrons multicouches (MLP, multilayer perceptron).

Un réseau moderne compterait environ 100 couches. Néanmoins c'est un nombre fini de couches de calcul pour passer de la séquence de jetons précédente au vecteur de probabilités du jeton suivant. La quantité de calcul effectuée pour chaque jeton est donc limitée. Il faut considérer cela comme une petite quantité de calcul. Cette quantité de calcul est quasiment fixe pour chaque jeton de cette séquence.

Ce n'est pas tout à fait exact, car plus vous introduisez de jetons, plus le coût de cette transmission sera élevé pour ce réseau neuronal. Mais pas de beaucoup.

Il faut donc y réfléchir, et je pense que c'est un bon modèle à garder à l'esprit. Il s'agit d'une quantité fixe de calcul qui sera effectuée dans cette boîte [le speaker montre le NanoGPT sur la figure 69] pour chacun de ces jetons [les jetons calculés l'un après l'autre sur la ligne horizontale de la figure 69].

---

64. Pour une bonne compréhension (c'est-à-dire un bon modèle dans votre tête), imaginez les jetons comme fixes sur l'axe horizontal, mais un nouveau est ajouté à droite, produit par le réseau neuronal, puis un nouveau, etc.

Cette quantité de calcul ne peut pas être trop importante, car il n'y a pas beaucoup de couches allant du haut vers le bas ici [le speaker montre la fig. 18]. La quantité de calcul nécessaire est faible.

On ne peut donc pas imaginer que le modèle effectue des calculs arbitrairement volumineux en une seule passe pour deviner un seul jeton.

*Cela signifie que nous devons répartir notre raisonnement et nos calculs sur plusieurs jetons, car chaque jeton ne peut faire l'objet que d'une quantité limitée de calcul.*

*Nous souhaitons donc répartir les calculs sur plusieurs jetons, car, encore une fois, nous ne pouvons pas avoir trop de calculs ni attendre trop de calculs du modèle pour un seul jeton.*

C'est pourquoi, si l'on considère les deux bonnes réponses (d'un point de vue mathématique) de la figure 68, la réponse de gauche est nettement *moins bonne* que celle de droite.

La raison est la suivante : imaginez que vous alliez de gauche à droite sur la partie gauche de la figure 68. Je l'ai copiée/collée dans Tiktokenizer. « La réponse est 3. C'est parce que... », figure 70.

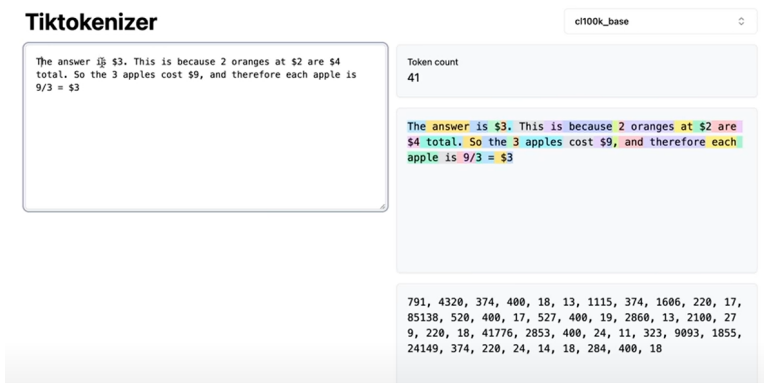


Figure 70: Tokenization de la mauvaise réponse.

Imaginez que le modèle, pour produire sa solution, doive aller de gauche à droite en émettant les nouveaux jetons, fig. 70, un par un. Il doit donc dire, ou nous nous attendons à ce qu'il dise : « La réponse est » « espace \$ » « 3 » « . », etc. Autrement dit nous attendons à ce qu'il concentre tous les calculs du problème dans la seule production du jeton représentant « 3 » (le jeton d'ID 18).

Il doit très vite calculer et émettre la bonne réponse: 3.

Une fois la réponse 3 émise, nous nous attendons à ce qu'il émette séquentiellement tous les jetons suivants: « . » « C'est » « parce que » « 2 » « oranges », etc.

Mais à ce stade, nous avons déjà produit la réponse, et elle est déjà dans la fenêtre contextuelle permettant de produire les jetons suivants.

Tout ce qui suit le nombre 3 (jeton 18), dans la figure 70, n'est qu'une sorte de justification a posteriori de la raison pour laquelle il s'agit de la réponse, car celle-ci est déjà créée – elle se trouve déjà dans la fenêtre de jetons.

La solution « 3 » n'est pas calculée progressivement tout au long de la séquence de jetons.

*Si vous répondez directement immédiatement à la question, vous entraînez le modèle à essayer de deviner la réponse avec un seul jeton.*

Cela ne fonctionnera pas bien, en raison de la quantité limitée de calculs que le LLM peut faire par jeton.

C'est pourquoi la réponse de droite est nettement meilleure, car elle *répartit* les calculs. Nous amenons le modèle à trouver la réponse *progressivement* – comme dans une bonne explication dans un livre ou un cours. De gauche à droite, nous obtenons des résultats intermédiaires. Nous disons : « OK, le coût total des oranges est de 4, donc 13 \$ - 4 \$ font 9 \$. »

Nous créons des calculs intermédiaires. Chacun de ces calculs n'est pas trop coûteux. En fait, nous estimons le niveau de difficulté que le modèle est capable de traiter dans la production d'un jeton à la fois.

*Il ne faut jamais qu'il ait trop de calcul à faire pour un seul jeton, car plus tard quand le modèle chercherait à imiter ce genre de conversation il ne saurait pas faire le calcul.*

Nous apprenons donc au modèle à répartir son raisonnement et ses calculs sur plusieurs jetons. Il peut construire sa solution progressivement. Vers la fin, il a tous les résultats intermédiaires nécessaires dans sa mémoire de travail. Il lui est beaucoup plus facile de déterminer que la réponse est..., et ici c'est 3<sup>65</sup>. La solution de droite est donc une bien meilleure conversation pour entraîner le modèle. La solution de gauche apprendrait au modèle à essayer d'effectuer tous les calculs avec un seul jeton.

C'est un point intéressant à garder à l'esprit lors de la construction de vos invites. Si vous répartissez les calculs, votre invite marchera bien.

Les équipes d'OpenAI ont des étiqueteurs qui veillent à ce que les réponses d'entraînement soient réparties.

Donc ChatGPT répondra bien, fig. 71.

---

65. Remarque : nous sommes ici dans le domaine des réseaux de neurones, et non dans celui de la programmation de problèmes mathématiques avec un langage informatique comme Fortran, JavaScript ou Python.

On peut dire que l'informatique comprend deux domaines différents : celui de la programmation, qui produit des résultats rigoureux et indiscutables (si le problème a été correctement *mathématisé* et le programme n'a pas de bug) ; et celui des réseaux de neurones qui découvrent des structures, des schémas émergents, des choses qui ne sont ni directement visibles ni même soupçonnées. (Cependant les réseaux de neurones sont programmés à l'aide des techniques du premier domaine.)

Lorsque nous demandons à Python de résoudre la question d'Émilie : « Émilie achète 3 pommes et 2 oranges. Chaque orange coûte 2 \$. Le coût total des fruits est de 13 \$. Quel est le coût des pommes ? », nous structurons un programme pour calculer rigoureusement la solution ( $13 \$ - 4 \$ = 9 \$$ ,  $9/3 = 3 \$$ ).

Lorsque nous utilisons un LLM pour résoudre le problème, ce n'est pas un programme qui le résout rigoureusement ; *c'est un programme qui devine un calcul ou imite un calcul issu de son apprentissage.*

En bref : les LLM ne sont pas conçus pour résoudre des problèmes mathématiques. Nous pouvons faire en sorte qu'ils donnent l'impression qu'ils résolvent des problèmes mathématiques simples, mais lorsque les problèmes sont plus complexes, ils échouent systématiquement.

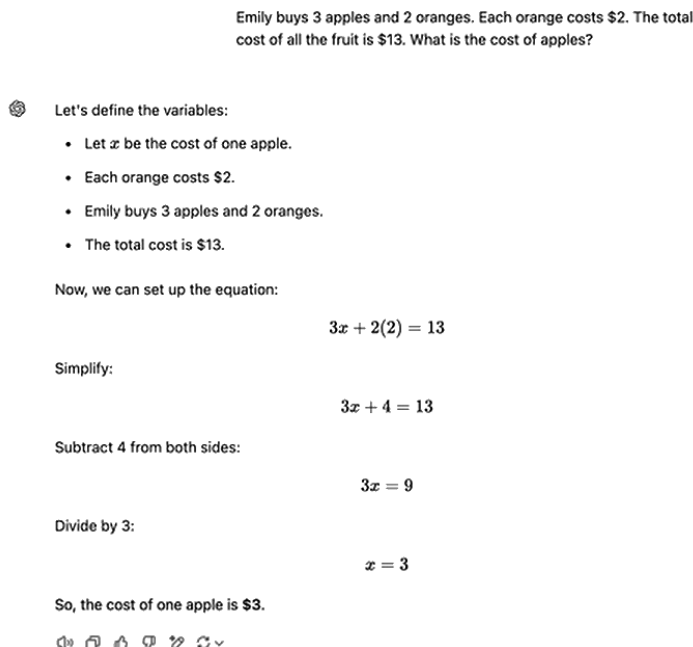


Figure 71: OpenAI résolvant correctement le problème d'Émilie. <sup>66</sup>

Quand je pose cette question à ChatGPT, il prend son temps pour donner la solution. Il dit « OK, définissons nos variables, établissons l'équation. . . », et le modèle génère tous ces résultats intermédiaires. Ces résultats intermédiaires ne sont pas pour nous, *ils sont pour le modèle*.

Si le modèle ne les génère pas pour lui-même, il ne pourra pas atteindre le résultat « 3 ».

66. Mais gardez à l'esprit qu'il ne s'agit que de l'imitation d'une démonstration mathématique au sens propre. Elle n'a pas été calculée par un programme informatique correctement écrit. La différence peut paraître ténue; c'est cependant un point fondamental. Quand un élève donne-t-il une réponse correcte parce qu'il a effectué une analyse et un raisonnement algébriques corrects, et quand s'agit-il simplement de l'imitation d'une preuve qu'il ne comprend pas vraiment ?



Je voulais aussi vous montrer qu'il est possible de forcer un peu le modèle et de le faire trébucher. Il suffit de poser des questions trop difficiles ou qui le perturberont.

Par exemple, j'ai donné exactement la même invite et j'ai dit : « Réponds à la question avec un seul jeton, donne-moi directement la réponse et rien d'autre. »

Il s'avère que pour cette simple invite, le modèle a pu le faire en une seule fois – enfin, en deux jetons, car le symbole dollar est son propre jeton.

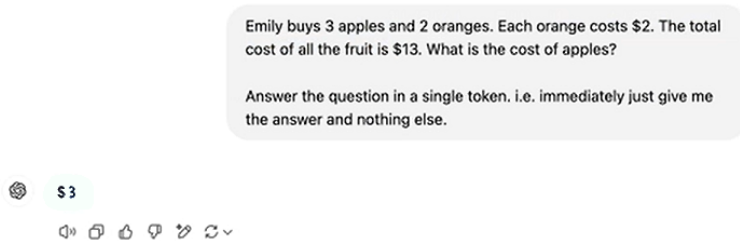


Figure 72: OpenAI résout correctement le problème d'Émilie, même lorsque nous lui imposons de le résoudre rapidement, sans étapes intermédiaires.

À vrai dire le modèle n'a pas produit un jeton, mais deux. Il a quand même produit la bonne réponse. Et ce, en un seul passage du réseau. C'est parce que les nombres du problème d'Émilie sont, je pense, très simples.

J'ai donc rendu la tâche un peu plus difficile pour être un peu méchant avec le modèle. J'ai dit : « Émilie achète 23 pommes et 177 oranges. Chaque orange coûte 4 \$. Le coût total de tous les fruits est de 869 \$. Quel est le prix de chaque pomme ? Réponds à la question avec un seul jeton, c'est-à-dire donne-moi immédiatement la réponse et rien d'autre. »

J'ai pris des nombres plus grands, ce qui complique le problème pour le modèle. Je lui demande d'effectuer davantage de calculs avec un seul jeton. J'ai posé la même question. Maintenant, il a répondu 5, ce qui est inexact.

Le modèle n'a donc pas réussi à effectuer tous ces calculs en un seul passage du réseau. Il n'a pas pu passer des jetons d'entrée au résultat en une seule passe dans le réseau.

Alors j'ai dit : « Bon, maintenant, ne te soucie plus de la limite de jetons et résous le problème comme d'habitude. »

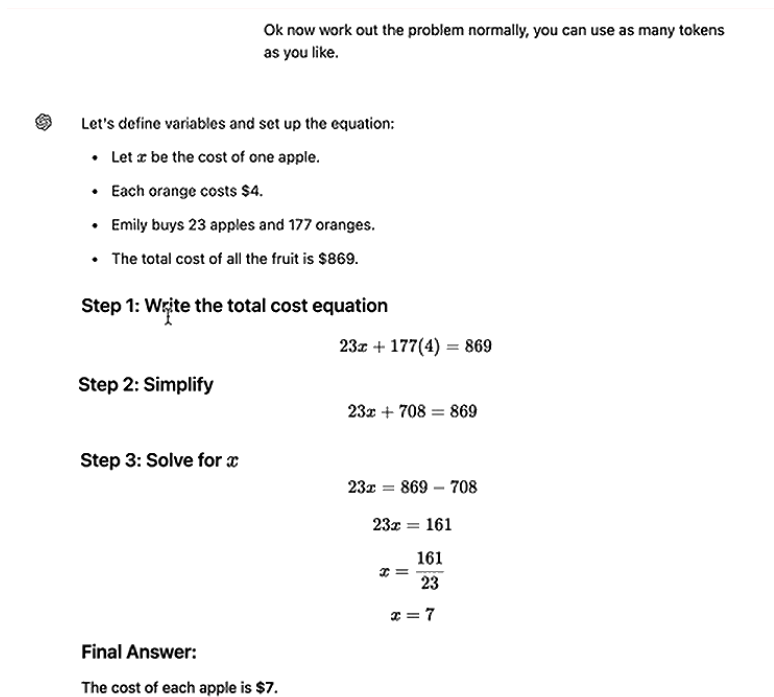


Figure 73: ChatGPT résout correctement le problème si nous le laissons utiliser autant d'étapes que nécessaire.

Cette fois, il établit tous les résultats intermédiaires, et montre de l'algèbre élémentaire. Chacun de ces résultats intermédiaires, ainsi que les calculs, sont beaucoup plus faciles à traiter pour le modèle. Il n'a pas trop de travail par jeton.

Tous les jetons sont corrects. Et il arrive à la bonne solution, c'est-à-dire « 7 ».

Mais il n'a pas été possible de condenser toutes les étapes intermédiaires dans une seule passe à travers le réseau.

C'est un exemple intéressant qui donne à réfléchir. Il éclaire sur le fonctionnement de ces modèles.

Noter que si j'avais à résoudre un tel problème dans mon travail au quotidien, je ne croirais pas sur parole les résultats intermédiaires de la figure 73<sup>67</sup>.

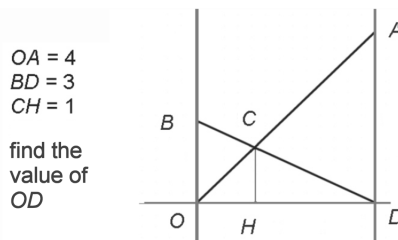
Voici probablement ce que je ferais : je me placerais ici [la boîte de questions principale de ChatGPT où nous avons placé l'invite « Émilie achète 23 pommes et 177 oranges. Chaque orange coûte 4 \$. Le coût total de tous les fruits est de 869 \$. Quel est le prix de chaque pomme ? »] et j'ajouterais : « *Utilise du code* ». Le code est en effet un des outils auxquels ChatGPT a accès.

Au lieu d'effectuer le calcul mental présenté figure 73, je force le modèle à procéder différemment, car je ne fais pas entièrement confiance au calcul mental présenté. Surtout si les nombres sont grands, rien ne garantit que le modèle le fera correctement. Chacune de ces étapes intermédiaires pourrait en principe être erronée sans que ça dérange le LLM<sup>68</sup>.

---

67. Un LLM peut écrire sans sourciller «  $2 \times x = 10$  donc  $x = 4$  », car en dépit de sa "preuve" bien présentée, *il ne fait aucun calcul*.

68. Le problème est que nous ne demandons pas à l'IA de faire réellement des maths. Le LLM se contente de « deviner » les étapes et la solution grâce à son entraînement. Aujourd'hui, 1er octobre 2025, le transcripteur a demandé à plusieurs IA de résoudre le problème classique suivant : « Deux échelles, de respectivement 3 m et 4 m, se croisent dans un couloir à 1 m du sol. Quelle est la largeur du couloir ? »



ChatGPT, Gemini, Copilot et Anthropic ont tous échoué, donnant des solutions fantaisistes de 1,4 m à 4 m ! Aucun n'a donné la bonne solution  $\approx 2.603$  m. DeepSeek a trouvé la bonne solution, mais avec des équations intermédiaires erronées ! Pour la vraie solution mathématique, voir par exemple <https://lapasserelle.com/documents/echelles.pdf>

Nous utilisons des réseaux de neurones pour faire du « calcul mental » comme on le fait dans son cerveau. Cela pourrait fausser certains résultats intermédiaires. C'est même incroyable que les LLM puissent faire ce genre de calcul mental. Je ne pense pas que je pourrais le faire de tête, mais en gros, le modèle le fait en quelque sorte de tête. Et je ne fais pas confiance à ces « calculs ».

J'ai donc décidé d'utiliser des outils. On peut ajouter « Utilise du code » dans notre invite.

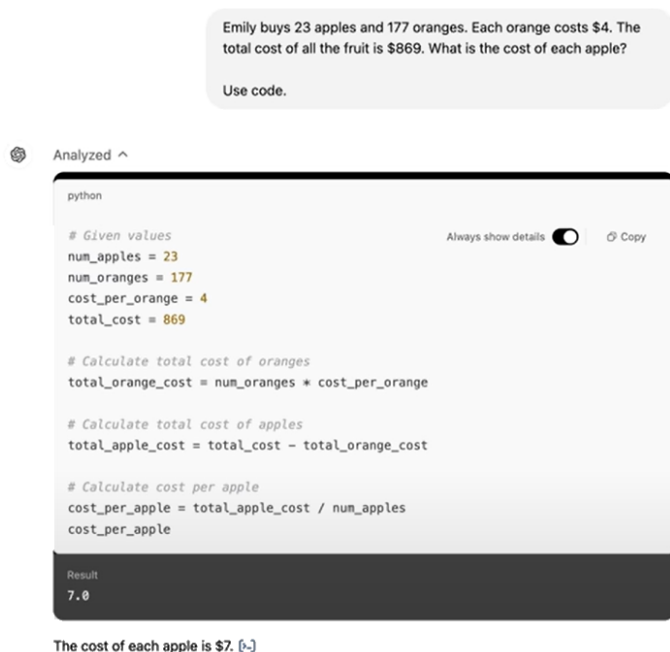


Figure 74: On dit à ChatGPT d'utiliser du code.

C'est un outil spécial. Le modèle peut écrire du code informatique et je peux le vérifier. Le modèle ne s'appuie plus sur son calcul mental. Le LLM « mathématise » le problème, mais une fois cette opération terminée, il utilise l'interpréteur Python, un langage de programmation très simple, pour écrire le code qui calcule le résultat.

Personnellement, je ferais bien plus confiance à cette méthode, car elle utilise un programme informatique écrit en Python, qui offre bien plus de garanties d'exactitude que le "calcul mental" fait par le LLM.

C'est une nouvelle aide pour s'assurer que le LLM est fiable: dans un problème de maths, vous pouvez demander au modèle d'utiliser l'interpréteur de code.

Comme pour la recherche web, le modèle dispose d'un type spécifique de jetons pour appeler Python. Il ne génère pas ces jetons à partir du modèle de langage. Il écrit un programme, puis l'envoie à une autre partie de l'ordinateur qui l'exécute et renvoie le résultat. Le modèle accède alors à ce résultat et peut vous dire: « OK, le prix de chaque pomme est de 7 \$ ». C'est un autre outil. Je vous recommande de l'utiliser en pratique. Cela réduit les erreurs possibles.

Voilà pourquoi j'ai appelé cette section *les modèles ont besoin de jetons pour penser* : répartissez vos calculs sur plusieurs jetons. Demandez aux modèles de générer des résultats intermédiaires. Ou, chaque fois que possible, privilégiez les outils et leur utilisation plutôt que de laisser les modèles tout faire en mémoire. S'ils essaient de tout faire en mémoire, je ne leur fais pas entièrement confiance. Je préfère utiliser des outils autant que possible.

Je veux vous montrer un autre exemple où cela se produit : le comptage. Les modèles ne sont pas très doués pour compter, pour la même raison : la génération du résultat avec un seul jeton demande beaucoup trop de ressources. Voyons un exemple : « Combien de points y a-t-il ci-dessous ? » Et j'ai mis un paquet de points.

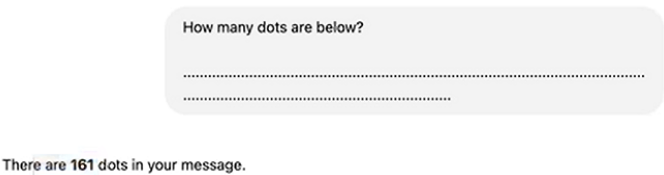


Figure 75: ChatGPT compte 161 points.

ChatGPT commence par « Il y a ». Il essaie ensuite de résoudre le problème avec un seul jeton.

Avec un seul jeton, il doit compter le nombre de points dans sa fenêtre contextuelle. Et ce, en une seule passe avant du réseau. Comme nous l’avons vu, lors d’une seule passe avant du réseau, le nombre de calculs effectués dans le réseau neuronal est limité. Imaginez que cela représente très peu de calculs.

Utilisons Tiktokenizer pour comprendre ce que le modèle voit. Il voit ceci : « Combien de points y a-t-il en dessous ? », tokenisé en haut à droite de la figure 76.

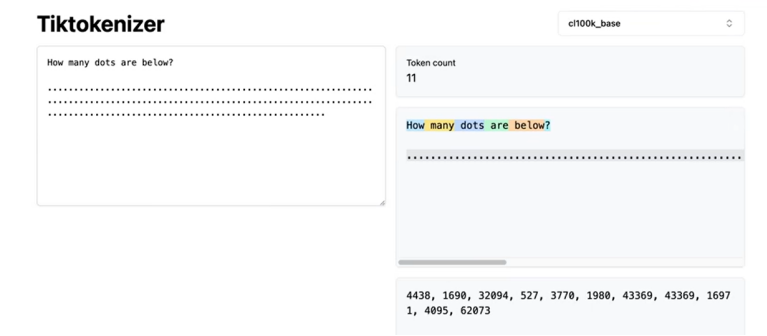


Figure 76: Ce que LLM voit quand on entre le prompt montré fig. 75.

En bas à droite de la figure, on voit les ID des jetons. D’abord : 4438, 1690, 32094, 527, 3770 et 1980, qui correspondent à « How », « many », « dots », « are », « below », « ? ».

Il s’avère ensuite que les points commencent à être tokenisés par groupes de 20 points, je crois. Un groupe de 20 points correspond à un seul jeton, portant le code 43369. Un deuxième groupe de 20 points correspond à la répétition de 43369.

Ensuite, pour une raison obscure, la régularité naissante est rompue. On voit les codes 16971, 4095 et 62073.

C’est lié aux détails de Tiktokenizer.

Le modèle voit les identifiants de jeton : ceci, ceci, ceci, etc. [groupes de 20 jetons], et à partir de ces identifiants, il est censé compter le nombre.

Attention, spoiler : *ce n'est pas 161*.

Je crois que le compte correct est 177.

Que pouvons-nous faire au lieu de ce qui est montré figure 75 ? Étonnamment, nous pouvons à nouveau dire « Utilise du code ».

Vous vous demandez peut-être: « Pourquoi cela fonctionnerait-il ? » C'est en fait assez subtil et intéressant.

Quand je dis « Utilise du code », je m'attends à ce que cela fonctionne. Vérifions:

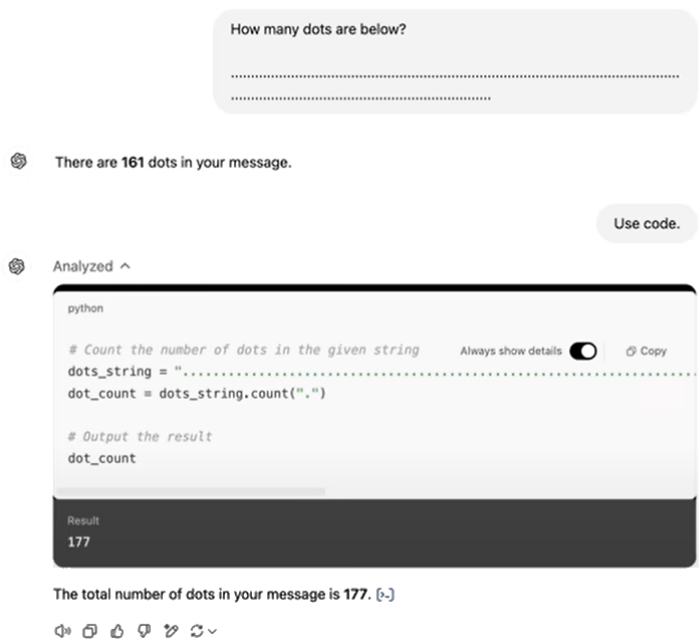


Figure 77: Nouveau comptage, après qu'on a ajouté l'ordre "Utilise du code" à la suite dans la même conversation.

C'est bon, il trouve maintenant 177.

Ce n'est pas apparent, mais j'ai décomposé le problème en problèmes plus faciles pour le modèle.

Je sais que le modèle ne sait pas compter. Il ne sait pas compter *mentalement*. Mais je sais qu'il est plutôt doué pour le copier/coller.

Ce qui se passe ici, c'est que lorsque je dis « Utilise du code », le modèle crée une chaîne de caractères en Python pour cela. Et la tâche, qui se réduit à copier/coller mon input dans le code Python, est très simple car le modèle voit cette chaîne de... il la voit simplement comme les quatre tokens 43369, 43369, 16971, 4095. Il est donc très simple pour le modèle de copier/coller ces ID de jetons et de les décomposer en points dans le code, fig. 77, voir la chaîne de points à droite de `dots_string =`.

Il crée donc cette chaîne, puis appelle la routine Python `dots_string.count(". ")`. Ensuite en Python compter les points est un jeu d'enfant. Ce n'est plus du "calcul mental" fait par le modèle. C'est un comptage élémentaire pour Python.

C'est encore une illustration du principe « Les modèles ont besoin de jetons pour penser ». Ne vous fiez pas à leur "calcul mental". C'est pourquoi aussi les modèles ne sont pas très performants en comptage. Si vous leur demandez d'effectuer des tâches de comptage, demandez-leur toujours d'utiliser l'outil « Utilise du code ».

02:01:10

## 14. Tokénisation revisitée : les modèles ont des difficultés pour épeler les mots et pour compter

Les modèles présentent de nombreux autres déficits cognitifs. Ce sont des limites technologiques importantes à prendre en compte.

Par exemple, les modèles ne sont pas très performants dans les tâches liées à l'orthographe des mots.

J'avais dit que nous reviendrions à la tokénisation.



*La raison pour la mauvaise performance des LLM pour travailler sur des lettres est que les modèles ne voient pas les caractères des mots, ils voient des tokens.*

Leur monde n'est constitué ni de lettres ni de mots, mais de tokens, ces petits fragments de texte. C'est pourquoi les tâches de niveau caractère échouent souvent.

Par exemple, je lui donne la chaîne « Ubiquitous » et je demande au modèle d'afficher seulement un caractère sur trois, en commençant par le premier. On commence donc par U, puis on passe à un caractère sur trois : « q » devrait être le suivant, puis « t », etc.

Réponse de ChatGPT: Uiuo. Mais la bonne réponse est: Uqts. Nous constatons que la réponse de ChatGPT est incorrecte.

C'est encore le calcul mental qui échoue. D'une part, le calcul mental n'est d'une manière générale pas bon pour les comptages. Mais surtout, le point le plus important ici est que si nous allons sur Tiktokenizer et que nous regardons « Ubiquitous », nous voyons qu'il s'agit de trois jetons.



Figure 78: Tokénisation de « Ubiquitous ».

Vous et moi voyons « Ubiquitous » et nous pouvons facilement accéder aux lettres individuelles car nous les voyons clairement<sup>69</sup>.

69. En résumé, la tokénisation est une technique qui présente des avantages, mais qui crée aussi des problèmes.

Lorsque nous l'avons dans la mémoire de travail de notre champ visuel, nous pouvons facilement indexer une lettre sur trois.

Mais les modèles n'ont pas accès aux lettres individuelles. Ils voient « Ubiquitous » comme les trois tokens montrés figure 78. Rappelons que ces modèles sont entraînés à partir de zéro sur Internet. Ils ne voient jamais de lettres. Le modèle doit déterminer combien de ces différentes lettres sont regroupées dans chacun de ces tokens.

L'utilisation des tokens est principalement motivée par l'efficacité.

*Beaucoup de gens souhaitent se débarrasser complètement des tokens. Nous devrions vraiment avoir des modèles au niveau des caractères ou des octets.*

Mais cela créerait des séquences très longues, et personne ne sait comment gérer cela pour le moment <sup>70</sup>.

Tant que nous sommes dans le monde des tokens, les tâches d'orthographe ne sont pas censées fonctionner très bien.

Sachant que les modèles ne sont pas performants en orthographe (à cause de la tokénisation), je peux à nouveau demander au LLM de s'appuyer sur des outils.

Je peux ajouter « Utiliser du code » dans la conversation (fig. 79). Je m'attends à ce que cela fonctionne, car copier/coller « Ubiquitous » dans l'interpréteur Python est beaucoup plus simple.

Nous nous appuyons ensuite sur l'interpréteur Python pour manipuler les caractères de cette chaîne.

Lorsque j'ajoute « Utiliser du code », le LLM indexe maintenant sans problème à l'aide de Python un caractère sur trois de « Ubiquitous ».

Sa réponse est Uqts, ce qui est correct (fig. 79).

Voilà donc un exemple de tâches liées à l'orthographe qui ne fonctionnent pas très bien.

---

70. C'est surprenant, car les LLM fonctionnent sans effort avec des vecteurs comportant des millions de composantes.

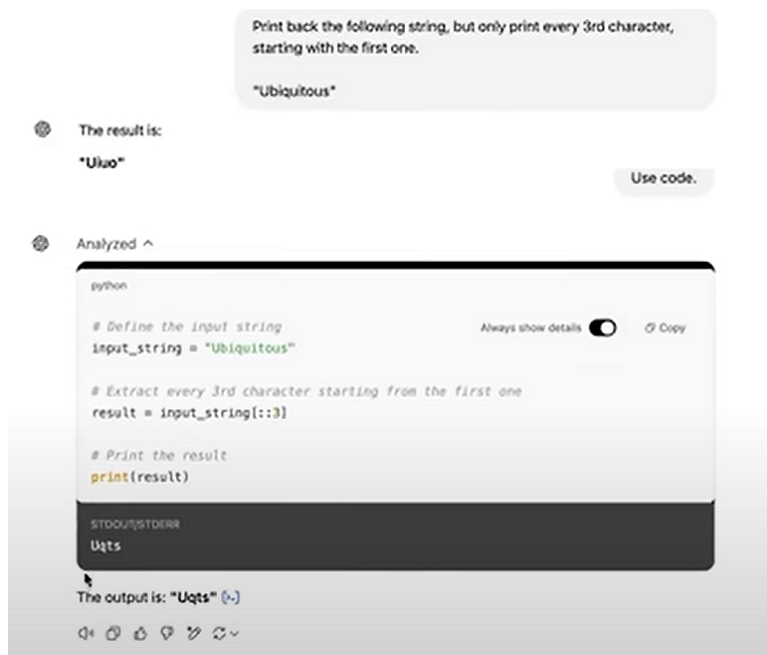


Figure 79: "Utiliser du code" pour manipuler des caractères.

Un exemple récent célèbre de problème similaire est « Combien y a-t-il de "r" dans "strawberry" ». C'est devenu viral pendant un moment.

De nos jours les LLM trouvent la bonne réponse. Ils répondent qu'il y a trois "r" dans "strawberry". Mais pendant longtemps, tous les modèles les plus avancés insistaient sur le fait qu'il n'y avait que deux "r" dans "strawberry".

Cela a provoqué beaucoup de remous. On se demande en effet: « Comment se fait-il que les modèles soient tellement brillants qu'ils puissent même résoudre des questions d'Olympiades de mathématiques<sup>71</sup> mais ne soient pas capables de compter les "r" dans "strawberry" ? »

71. La rumeur que les LLM « savent résoudre des problèmes d'Olympiades de mathématiques » est une exagération enthousiaste et marketing pour promouvoir les LLM.

La réponse est encore une fois :

1) les modèles ne voient pas les caractères, ils voient les jetons ;

2) ils ne sont pas très doués pour compter.

Nous combinons ici la difficulté de voir les caractères avec celle de compter.

C'est pourquoi les modèles ont eu du mal à compter le nombre de "r" dans "grawberry". OpenAI a peut-être maintenant codé la réponse en dur (je ne sais pas exactement ce qu'ils ont fait), mais la requête spécifique concernant « straw-berry » fonctionne désormais.

Les modèles ne sont donc pas très doués pour l'orthographe.

Il y a beaucoup d'autres défauts. Je ne vais pas tous les aborder. Je veux juste vous montrer quelques exemples de points à prendre en compte lors de l'utilisation pratique des LLM.

Certaines réponses semblent absurdes. Et cela surprend car ces modèles sont censés résoudre des problèmes mathématiques complexes. Ils peuvent répondre à des questions de physique, de chimie et de biologie de niveau doctorat. Mais parfois, ils ne parviennent pas à répondre à des questions très simples comme « Quel nombre est le plus grand 9,11 ou 9,9 ? »

---

De récentes études comparatives (USAMO 2025 et IMO) montrent que même les meilleurs LLM actuels (comme Gemini 2.5 Pro, GPT-4 et Grok-4) obtiennent des résultats bien inférieurs au niveau nécessaire pour espérer une médaille aux questions réelles d'Olympiades, en particulier lorsque des preuves complètes et un raisonnement mathématique rigoureux sont exigés.

Lors des Olympiades des mathématiques des États-Unis (USAMO 2025), les meilleurs LLM n'ont obtenu que 25% à 30% (bien en dessous d'une médaille de bronze) lors de l'évaluation de solutions complètes comme on les demande aux concurrents humains.

Les LLM peuvent parfois fournir la réponse numérique finale correcte dans des problèmes de compétition plus simples (de type AIME), mais échouent presque toujours à produire des preuves valides et rigoureuses pour des mathématiques de niveau Olympiade.

Pour en savoir plus, voir le problème des échelles dans un couloir dans la note de bas de page n° 69 page 117.

ChatGPT commence par répondre : « 9,11 est plus grand que 9,9 ». Mais il explique immédiatement... pourquoi 9,11 est plus petit ! (fig. 80)

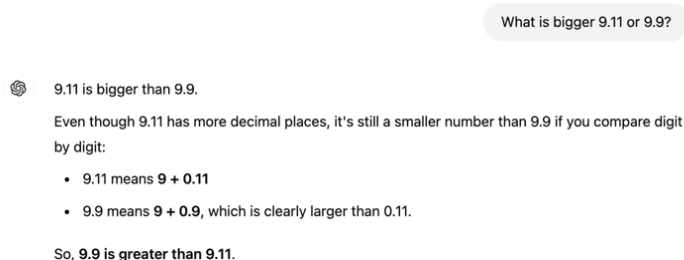


Figure 80: Comparaison de deux nombres décimaux.

Je ne pense pas que ce soit très reproductible. Parfois, la réponse est inversée ; parfois, elle est correcte immédiatement ; parfois, elle se trompe pour de bon.

Réessayons : ChatGPT indique ici que même si 9,9 peut sembler plus grand que 9,11, il est en fait plus petit. Il conclut :  $9,11 > 9,9$ . Il ne se corrige donc même pas à la fin.

Si vous posez la question à plusieurs reprises, il arrive aussi qu'il ait raison <sup>72</sup>.

Mais comment se fait-il que le modèle soit si performant pour les problèmes de niveau Olympiade, mais échoue ensuite sur des problèmes très simples comme celui-ci ?

Je pense que celle-ci (fig. 80), comme je l'ai mentionné, est un peu incompréhensible.

Il s'avère qu'un groupe de personnes a étudié la question en profondeur [c'est-à-dire l'étrange réponse du LLM] et a publié un article.

---

72. Il semble clair que, puisque les « meilleures suppositions » sont basées sur des statistiques calculées lors de la phase de pré-apprentissage [énorme et très coûteuse], et qu'elles ajoutent des choix stochastiques, cela n'a pas grand-chose à voir avec les mathématiques. *Les mathématiques impliquent un raisonnement logique* ; les LLM – en 2025 – n'utilisent pas le raisonnement logique ; en mathématiques, ils ne font qu'*imiter* le raisonnement logique.

Je n'ai pas lu leur article. Mais l'équipe m'a expliqué que lorsqu'on examine les activations au sein du réseau neuronal, lorsqu'on observe certaines caractéristiques, celles qui s'activent ou pas, et les neurones qui s'activent ou pas, un certain nombre de neurones s'activent à l'intérieur du réseau neuronal, généralement associés à des versets bibliques.

Le LLM se rappelle que 9,11 et 9,9 ressemblent presque à des marqueurs de versets bibliques. Et dans un contexte textuel [biblique ou autre], 9.11 viendrait après 9.9<sup>73</sup>.

Le modèle est perturbé car dans les versets bibliques, 9.11 est supérieur à 9.9 [au sens où il vient après].

Et s'il tente de le justifier et de retrouver la réponse par des calculs mathématiques, il s'égare et aboutit aléatoirement à la bonne ou à la mauvaise réponse.

Ce phénomène n'est pas tout à fait logique, et n'est pas entièrement compris.

Il y a d'autres problèmes bizarres de ce genre. C'est pourquoi vous devez considérer un LLM pour ce qu'il est: un système stochastique qui peut être vraiment magique, mais auquel on ne peut pas faire entièrement confiance.

Vous devez l'utiliser comme un outil, et non comme un génie dont il suffit de recopier les résultats.

02:07:30

## 15. Récapitulation: du modèle de base à l'assistant

Nous avons abordé les deux phases principales de l'apprentissage de grands modèles de langage.

La première phase, appelée pré-apprentissage, consiste essentiellement à entraîner le LLM sur des documents Internet.

Lorsqu'on entraîne un modèle de langage sur des documents Internet, on obtient ce qu'on appelle un *modèle de base*.

---

73. Il s'agit simplement d'une façon de numérotter les sections et les sous-sections qui *ressemble* à des nombres décimaux, mais qui n'en sont pas.

Il s'agit simplement d'un simulateur de documents Internet.

Nous avons constaté qu'il s'agit d'un artefact intéressant <sup>74</sup>. L'apprentissage sur des milliers d'ordinateurs prend plusieurs mois. C'est une sorte de compression d'Internet avec perte. Le résultat est extrêmement intéressant.

*Mais ce n'est pas directement utile, car nous ne souhaitons pas simplement imiter des documents Internet<sup>75</sup>. Nous voulons pouvoir poser des questions à une IA et obtenir des réponses.*

Pour cela, nous avons besoin d'un *assistant*.

Nous avons vu qu'il est possible de construire un assistant lors d'une phase de post-apprentissage, et plus précisément lors d'un processus de réglage fin supervisé (SFT).

À cette étape, nous avons constaté que l'algorithme est identique à celui de la phase de pré-apprentissage. Rien ne change [aux procédures]. La seule chose qui change, c'est l'ensemble de données. Au lieu de documents internet, nous souhaitons maintenant créer et organiser un ensemble de données de conversations très complet. Nous souhaitons des millions de conversations sur des sujets très variés entre un humain et un assistant. Et ces conversations sont fondamentalement créées par des humains. L'idée clé est que les LLM copient essentiellement ce qu'ils ont appris. Nous devons donc les entraîner à mener des conversations complètes, dont le début sert d'invite dans la fenêtre contextuelle.

Les humains rédigent les invites et la ou les réponses idéales. Ils le font en s'appuyant sur des documentations d'étiquetage (c'est-à-dire des instructions fournies par l'entreprise qui développe le LLM).

---

74. Dans la communauté de l'IA, « artefact » désigne simplement un outil, une ressource ou un résultat concret, comme un fichier de modèle, un jeu de données ou un sous-produit d'un processus. Il ne s'agit pas d'un « phénomène parasite corrompant les résultats d'une expérience », comme c'est le cas en physique.

75. En produisant des séquences de tokens à la queue leu leu, à la suite d'un prompt, comme on a vu à la fin du pré-entraînement.

Dans une stack moderne, ce n'est pas entièrement réalisé manuellement par des humains. Ces outils nous aident désormais beaucoup. On utilise des LLM pour créer ces ensembles de données. C'est un processus très répandu. Mais tout repose encore en fin de compte sur la supervision humaine.

Nous créons donc ces conversations qui constituent désormais notre ensemble de données. Nous poursuivons l'entraînement et affinons nos paramètres. Et nous obtenons un assistant.

Nous avons ensuite changé de cap et abordé les implications cognitives de cet assistant. Nous avons constaté, par exemple, que l'assistant peut avoir des hallucinations si aucune mesure d'atténuation n'est prise. Nous avons constaté que les hallucinations étaient fréquentes. Nous avons ensuite examiné certaines des mesures d'atténuation de ces hallucinations.

Nous avons ensuite constaté que les modèles sont assez impressionnants et peuvent réaliser de nombreuses actions mentales. Nous avons également vu qu'ils peuvent s'appuyer sur des outils pour s'améliorer. Par exemple, nous pouvons nous appuyer sur une recherche web pour éviter les hallucinations et éventuellement obtenir des informations plus récentes.

Nous pouvons aussi utiliser des outils comme un interpréteur de code. Le LLM peut écrire du code, l'exécuter et visualiser les résultats.

Ce sont quelques-uns des sujets que nous avons abordés jusqu'ici.

02:10:05

## 16. Du fine-tuning supervisé à l'apprentissage renforcé

J'aimerais maintenant aborder la dernière étape importante de cette séquence: *l'apprentissage par renforcement*.

L'apprentissage par renforcement est parfois considéré comme faisant partie du post-apprentissage. Mais il s'agit d'une troisième étape majeure.



Il s'agit d'une méthode différente d'entraînement des LLM. Au sein d'entreprises comme OpenAI, les étapes sont confiées à des équipes distinctes.

Il y a une équipe qui prépare les données pour le pré-entraînement, et une équipe qui s'occupe du pré-entraînement du réseau neuronal.

Puis, il y a une équipe qui s'occupe de la génération de conversations, et une autre qui se charge du réglage fin supervisé (SFT).

Il y a enfin une équipe qui s'occupe de l'apprentissage par renforcement.

C'est un flux, un peu comme des passes au rugby où on passerait le modèle d'un joueur à l'autre. On obtient d'abord le *modèle de base*. On découvre qu'on veut un *assistant*. On passe ensuite le modèle à l'apprentissage par renforcement (RL), dont nous allons parler maintenant. C'est le flux principal.

02:11:00

Concentrons-nous maintenant sur l'*apprentissage par renforcement*, la dernière étape majeure de l'entraînement.

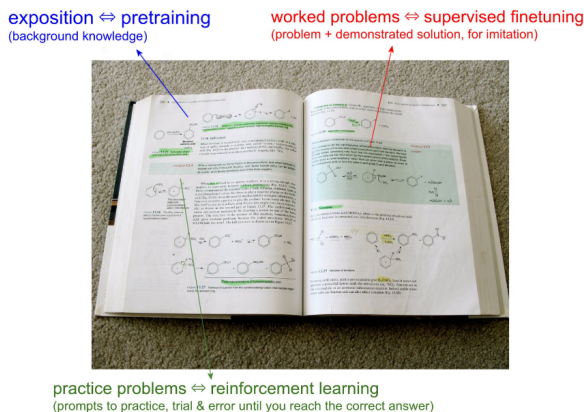


Figure 81: Les trois grandes étapes comparées à l'apprentissage par un élève dans un manuel.

Voyons en quoi cela consiste.

Une image éclairante est de comparer les trois grandes étapes (pré-entraînement, réglage fin supervisé, et apprentissage par renforcement) à un processus que nous connaissons tous bien : la façon dont un élève apprend dans un manuel, fig. 81.

De même que vous êtes allé à l'école pour devenir excellent dans un domaine, nous voulons emmener notre grand modèle de langage à l'école.

Un élève suit plusieurs étapes pour absorber des connaissances ou des compétences.

Lorsque nous travaillons avec des manuels scolaires, nous pouvons voir qu'ils contiennent trois principaux types d'informations, trois classes d'informations :

1. La première chose que nous voyons, c'est l'*exposition basique des savoirs* – le déroulé du text, si vous voulez. Notez que j'ai pris un livre totalement au hasard sur Internet – je crois que c'est un livre de chimie organique, je ne suis pas sûr – mais l'important est de voir que la majeure partie de ce qui est imprimé est l'exposition de base. C'est en quelque sorte la connaissance de base.

Lorsque vous lisez les mots de cette exposition, vous pouvez considérer cela comme un entraînement sur ces données. C'est pourquoi la lecture de ces informations de base est comparable au pré-entraînement. C'est là que nous construisons une sorte de base de connaissances, à partir de ces données, et que nous nous faisons une première idée du sujet.

Mais pour la plupart d'entre nous, c'est tout à fait insuffisant pour connaître à fond ce que présente le livre. Il faut encore beaucoup de travail :-)

2. L'information importante suivante que nous trouvons dans le manuel est les *problèmes résolus*. Un expert, en l'occurrence l'auteur du livre, nous montre des problèmes, mais également leur solution. Ces solutions sont comparables à la réponse idéale d'un assistant.

L'expert nous montre comment résoudre un problème en détaillant complètement la solution.

Lorsque nous lisons la solution, nous nous entraînons sur les données de l'expert. Ensuite, nous pourrions essayer d'imiter l'expert. Cela correspond approximativement à l'étape SFT.

Dans la vidéo, nous avons déjà effectué le pré-entraînement et vu l'imitation des experts dans leur façon de résoudre ces problèmes.

3. La troisième étape, l'apprentissage par renforcement, est l'analogue des *exercices pratiques*. Il s'agit parfois d'un exercice pratique proposé au milieu du texte d'exposition (sans sa réponse complète), mais bien sûr, il y en a généralement un grand nombre dans quelques pages à la fin de chaque chapitre du manuel.

Les exercices pratiques sont essentiels à l'apprentissage, car ils vous encouragent à vous entraîner et à découvrir des solutions à ces problèmes *par vous-même*.

Dans un exercice pratique, on donne l'énoncé du problème, mais pas la solution. En revanche le résultat final vous est souvent donné, par exemple dans un livret à la fin du manuel.

Vous connaissez ainsi l'énoncé du problème, le résultat final que vous devez trouver, mais le livre ne vous donne pas la solution détaillée. C'est à vous de la trouver, c'est-à-dire comment démontrer le résultat. Vous testez différentes méthodes et vous voyez ce qui vous permet d'arriver au mieux au résultat final. Vous découvrez ainsi comment résoudre ces problèmes.

Ce faisant, vous vous appuyez d'abord sur les informations de base issues du pré-entraînement, et ensuite, sans doute, sur l'imitation d'experts humains. Vous pouvez probablement essayer des solutions similaires, etc.

Dans cette vidéo, nous avons effectué les points 1 et 2: le pré-entraînement et le réglage fin supervisé.

Maintenant, nous allons nous continuer à nous entraîner à l'aide de l'apprentissage renforcé. Nous allons recevoir des consignes (= prompts). On va nous donner les réponses finales. Mais nous n'aurons pas accès à des solutions complètes fournies par des experts.

Nous devons nous entraîner et tester des voies possibles. C'est le but de l'apprentissage par renforcement.

Revenons au problème d'Émilie pour avoir un exemple concret.

Figure 82, nous sommes dans le Tiktokenizer. Il y a une zone à gauche où on entre du texte. C'est là qu'on introduit le prompt qu'on veut donner au LLM. Mais rappelez-vous que nous travaillons toujours avec des séquences de jetons unidimensionnelles.

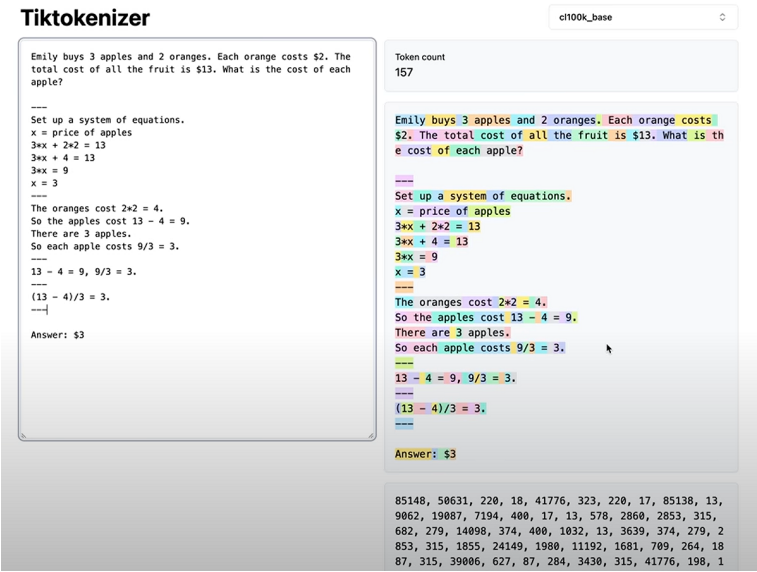


Figure 82: Reinforcement learning.

C'est pourquoi je préfère toujours regarder un prompt à l'aide de Tiktokenizer. Il s'agit en effet, à droite, de la vue native par le LLM, si je puis dire. C'est ce qu'il voit réellement: les jetons et leurs ID.

02:15:10

« Émilie achète 3 pommes et 2 oranges. Chaque orange coûte 2 \$. Le coût total de tous les fruits est de 13 \$. Quel est le prix d'une pomme ? »

Sur la figure 82, côté gauche, le prompt comprend l'énoncé et quatre solutions possibles, qui toutes aboutissent à la réponse « 3 ».

Ce qu'il faut comprendre ici, c'est que si je suis l'annotateur humain qui doit créer une conversation à intégrer à l'ensemble d'apprentissage, je ne sais pas vraiment laquelle des quatre il vaut mieux utiliser. La première conversation établit un système d'équations. La seconde se contente de l'expliquer en français. La troisième va vite à la solution, et la quatrième y va en une seule équation.

Voyons comment répond ChatGPT.

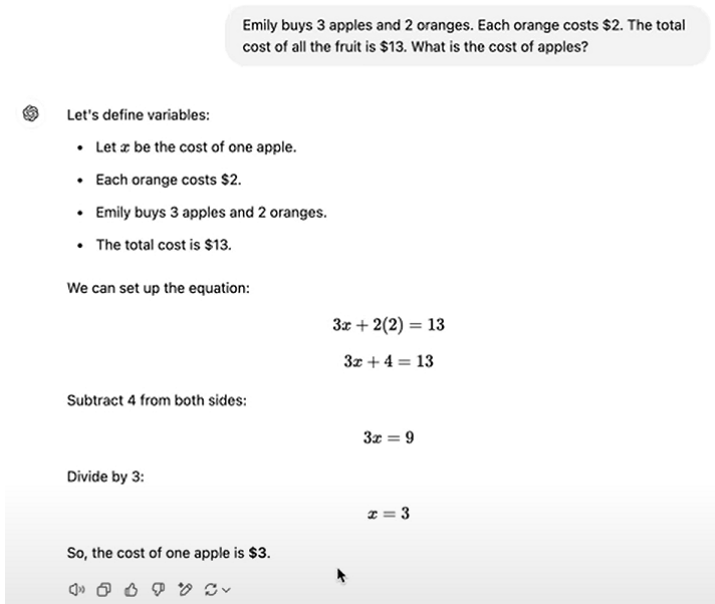


Figure 83: Solution donnée par ChatGPT.

Il définit des variables, construit un système d'équations et fait [en tout cas *montre*] un peu d'algèbre.

Ce qu'il faut comprendre et distinguer, c'est que l'objectif premier d'une solution est bien sûr d'atteindre la bonne réponse. Nous voulons obtenir le résultat final « 3 ». C'est l'objectif principal ici.

Mais il y a aussi un objectif secondaire: nous voulons rendre la solution aisément lisible et compréhensible par un être humain.

Nous partons du principe que la personne qui lira la réponse donnée par un LLM souhaitera voir les étapes intermédiaires détaillées conduisant à la solution. Nous souhaitons présenter cette solution complète de manière lisible, agréable, etc.

Il y a donc deux objectifs distincts.

- 1) Trouver le bon résultat.
- 2) Présenter la solution complète d'une manière claire pour un être humain.

Concentrons-nous pour l'instant sur l'obtention du bon résultat final. Si seul le résultat final nous intéresse, laquelle des quatre solutions à gauche, sur la figure 82, est la meilleure à donner au LLM pour qu'il s'entraîne dans l'étape de réglage fin supervisé (SFT) ?

Le point important sur lequel je veux insister c'est que *nous ne le savons pas*. En tant qu'étiqueteur (= annotateur) humain, je ne saurais pas laquelle est la meilleure.

Par exemple, nous avons vu plus tôt (fig. 69, p. 109), en examinant le calcul mental et le raisonnement dans la séquence de jetons, que pour chaque jeton nous ne pouvons effectuer qu'une quantité limitée de calcul.

Nous ne pouvons pas faire de saut trop important pour un seul jeton. Par exemple, la quatrième solution, sur la fig. 82, atteint « 3 » trop rapidement.

$$(13 - 4)/3 = 3.$$

Figure 84: Quatrième solution obtenue en une seule équation.

Peut-être que c'est un mauvais exemple à donner au LLM car cela l'encourage à sauter très rapidement à travers les étapes intermédiaires.

Si on lui donne cet exemple lors du SFT, il n'en tirera aucun enseignement et ne saura pas résoudre des problèmes comparables plus tard quand on l'utilisera pour faire des maths.

Cela accroîtra au contraire le risque qu'il fasse des erreurs dans son "calcul mental".

Peut-être ce serait mieux d'étaler davantage les étapes comme dans la troisième solution <sup>76</sup>.

Peut-être ce serait mieux d'établir un système d'équations comme dans la première solution ? <sup>77</sup>

Peut-être ce serait mieux de décrire la solution en français ?

*Fondamentalement, on ne sait pas.*

Nous ne savons pas car ce qui est facile ou difficile pour vous et moi ou pour un annotateur humain, n'est pas automatiquement pareil pour un LLM. *Son fonctionnement cognitif est différent* <sup>78</sup>. Les séquences de jetons présentent une sorte différente de difficulté. Certaines séquences, fig. 82, qui sont triviales pour nous, peuvent aller trop vite pour un LLM.

---

76. Pour vérifier, j'ai posé à nouveau – en octobre 2025 – le problème des échelles à Perplexity (voir note 68 page 117), en le décomposant en étapes simples, en posant des questions simples une par une, en appliquant Thalès et Pythagore dans un triangle, puis dans un autre. Mais l'IA commet de nombreuses erreurs élémentaires et est incapable de résoudre le problème des échelles, même décomposé en étapes élémentaires.

77. Les LLMs ne sont fondamentalement pas faits pour faire des maths. Ils *devinent* ou *imitent*, là où il faut *raisonner*.

Le paradoxe est que, quand j'ai développé des méthodes de langues en html, javascript, php, etc., durant l'été 2025, [https://lapasserelle.com/IA\\_anglais/methode\\_anglais\\_IA\\_1200.php](https://lapasserelle.com/IA_anglais/methode_anglais_IA_1200.php), Perplexity a été fabuleusement utile pour écrire le code et le débbugger. Je n'ai eu aucune ligne de code à écrire. L'IA codait précisément ce que je me contentais de lui décrire. Elle a produit une douzaine de fichiers de plusieurs dizaines de lignes chacun s'appelant les uns les autres.

Il est vraisemblable que d'ici quelque temps les LLM sauront faire de vraies mathématiques et pas seulement "imiter des raisonnements".

78. Formule élégante pour dire que les LLM ne raisonnent pas.

C'est pourquoi figure 84, dans la quatrième solution du problème d'Émilie, le calcul du dernier jeton – qui atteint d'un seul coup la solution « 3 » à partir des jetons précédents – est sans doute beaucoup trop dur.

À l'inverse, beaucoup des jetons que je crée ici, fig. 82, pourraient être inutiles pour le LLM, et nous ne faisons que gaspiller des jetons. Si la seule chose qui nous importe est la réponse finale (item 1, p. 136), et que nous laissons de côté la question de la présentation pour un être humain, alors nous ne savons pas vraiment comment annoter le problème. Nous ne savons pas quelle solution donner au LLM, car nous ne sommes pas un LLM. C'est clair dans le cas des calculs nécessaires pour résoudre le problème d'Émilie.

C'est un problème des LLM en général: *nos connaissances ne sont pas celles du LLM*, et nous ne savons pas comment préparer une conversation ou une solution mathématique pour l'entraîner.

En outre, le LLM possède déjà une mine de connaissances comparable à celles de quelqu'un qui a un doctorat en mathématiques, en physique, en chimie, etc. À bien des égards, il en sait plus que nous. Il est raisonnable de suspecter que nous n'utilisons pas toutes *ses* connaissances pour résoudre le problème.

Mais à l'inverse, il est possible je fasse appel implicitement dans mes solutions à une quantité de connaissances que le LLM n'a pas dans ses paramètres. Cela crée alors pour lui des sauts soudains qui le perturbent considérablement.

*Répetons-le: Nos cognitions sont donc différentes.*

Je ne sais pas vraiment quoi mettre ici (fig. 82, à gauche) si notre seul souci est d'atteindre la solution finale, et de le faire de manière économiquement optimale. En bref, nous ne sommes pas en bonne position pour créer ces séquences de jetons pour le LLM. Et elles sont utiles par imitation pour initialiser le système.

*Ce qu'on veut, c'est que LLM découvre lui-même les séquences de jetons qui lui conviennent.*



Il faut qu'il trouve par lui-même quelle séquence de jetons lui permet de manière fiable d'atteindre le résultat final – qu'on lui donne dans le prompt. On va faire en sorte qu'il le découvre tout seul dans un processus d'essais et d'erreurs. C'est l'essence de l'apprentissage renforcé.

02:20:00

## 17. Apprentissage renforcé

Voyons comment l'exemple d'Émilie est traité pour l'apprentissage renforcé. Nous sommes à nouveau dans le bac à sable d'Hugging Face pour faire des inférences. Il nous permet d'utiliser très simplement toutes sortes de LLM. Par exemple, dans le menu déroulant à droite, j'ai le modèle Gemma-2-2b – c'est-à-dire à 2 milliards de paramètres<sup>79</sup>.

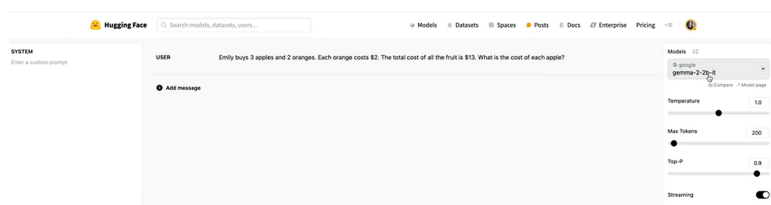


Figure 85: Bac à sable de Hugging Face. Nous utilisons le modèle Gemma-2-2b.

De nos jours, deux milliards de paramètres est très peu, mais c'est suffisant pour illustrer ce qu'on veut montrer.

La façon dont l'apprentissage renforcé (RL en anglais) va opérer est très simple: on va essayer un grand nombre de solutions différentes. Et l'on va regarder quelles solutions fonctionnent bien.

On part donc du prompt "Émilie achète 3 pommes et 2 oranges, etc.", et on fait tourner le modèle. Il produit la solution montrée fig. 86.

---

79. Les modèles Gemma 2b et Gemma 2b-it ont été créés par Google DeepMind, en collaboration avec d'autres équipes de recherche en IA de Google.

On inspecte la solution. Nous savons déjà qu'il faut qu'il arrive à la solution « 3 \$ ».

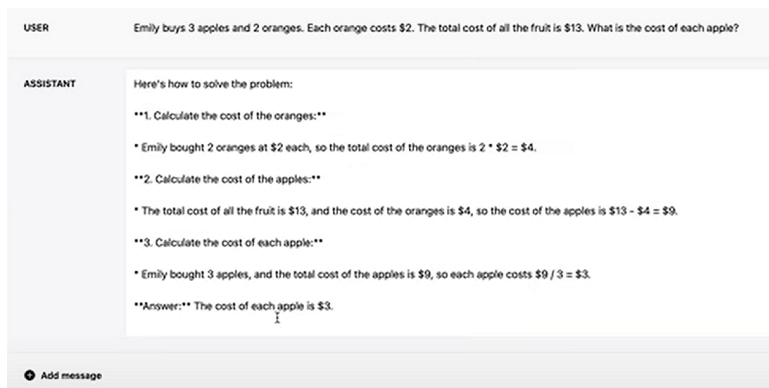


Figure 86: Solution par Gemma-2-2b du problème donné dans le prompt.

Le modèle a bien trouvé la solution correcte « 3 \$ »<sup>80</sup>. C'était juste un essai – un tirage.

On l'efface et on recommence avec un deuxième essai.

Le modèle produit une solution légèrement différente.

*Chaque essai produit une solution légèrement différente, car les modèles sont des systèmes stochastiques. N'oubliez pas qu'à chaque jeton, nous avons une distribution de probabilité et que nous échantillonnons à partir de cette distribution. Nous empruntons donc des chemins légèrement différents [vers la solution proposée par le modèle].*

Le deuxième essai arrive aussi à trois dollars.

On l'efface et on essaie une troisième fois. À nouveau, solution légèrement différente, mais même résultat.

80. Notez que cela ne prouve pas qu'il a suivi un raisonnement mathématique correct – pour la simple raison qu'il ne raisonne pas, mais imite simplement son corpus de données d'entraînement (documents internet et conversations). Espérons que dans un avenir proche les LLM seront capables de *raisonner* comme les humains.

En apprentissage par renforcement, on peut répéter ce processus plusieurs fois. En pratique, on peut échantillonner des milliers, voire des millions de solutions indépendantes pour une seule question.

Certaines seront très bonnes, d'autres moins bonnes ou carrément fausses.

En résumé, l'objectif est d'encourager les solutions qui mènent à des réponses correctes.

Voyons à quoi cela ressemble. La figure 87 illustre ce schéma.

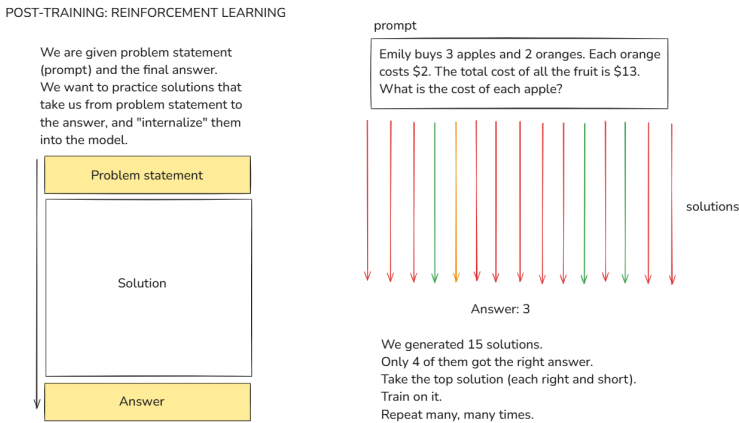


Figure 87: Schéma de l'apprentissage renforcé.

Nous avons une question (= un prompt). On essaie plusieurs solutions différentes en parallèle. Certaines solutions peuvent bien fonctionner et donc atteindre la bonne réponse (elles sont en vert). D'autres peuvent mal fonctionner et ne pas atteindre la bonne réponse (elles sont en rouge), fig. 87.

Le problème d'Emily n'est malheureusement pas le meilleur exemple, car il s'agit d'une question triviale. Et comme nous l'avons vu, même un modèle à deux milliards de paramètres obtient systématiquement la bonne réponse.

Mais faisons preuve d’imagination. Supposons que les solutions vertes<sup>81</sup> soient bonnes et les solutions rouges mauvaises, figure 87.

Imaginons que nous ayons généré 15 solutions, et que seules quatre d’entre elles aboutissent à la bonne réponse. Nous souhaitons maintenant encourager les solutions qui mènent à des réponses correctes.

Quelles que soient les séquences de jetons rencontrées dans les solutions rouges, il est évident qu’un problème s’est produit quelque part. Et ce n’était pas le bon chemin à suivre pour trouver la solution. Et quelles que soient les séquences de jetons rencontrées dans les solutions vertes, tout s’est plutôt bien passé dans cette situation.

Nous voulons que notre LLM réponde comme dans les solutions vertes. Pour encourager ce type de comportement à l’avenir, nous l’entraînerons sur ces séquences.

*Mais ces séquences qui serviront pour l’entraînement ne proviennent pas d’annotateurs humains experts. Aucun humain n’a décidé que c’était des bonnes solutions. Elles sont venues du modèle lui-même.*

Le modèle s’entraîne tout seul ici. Il a testé quelques solutions. Quatre d’entre elles semblent fonctionner. Le modèle s’entraînera sur elles. Cela correspond à un élève qui examine ses propres solutions et se dit : « Très bien, celle-ci a bien fonctionné, alors c’est comme cela que je devrais résoudre ce genre de problèmes. »

Dans le problème d’Émilie, il y a de nombreuses façons d’affiner encore la méthodologie, mais restons simples.

Une façon de procéder est de simplement prendre la meilleure solution parmi les quatre – c’est pourquoi elle est en jaune (fig. 87). C’est une solution qui non seulement a conduit à la bonne réponse, mais elle présentait peut-être aussi d’autres caractéristiques intéressantes. Peut-être était-elle la plus courte ou avait-elle d’autres avantages souhaitables.

---

81. Vertes ou jaune. En fait la jaune est la meilleure, voir texte.

On peut imaginer d'autres critères. Quoi qu'il en soit, imaginons que la jaune est la meilleure solution. Elle sera incluse dans notre ensemble d'entraînement.

Le modèle sera alors légèrement plus susceptible – une fois ses paramètres mis à jour – de suivre une solution similaire pour des problèmes de ce genre à l'avenir.

N'oubliez pas que nous allons exécuter de nombreuses invites (= prompts) différentes pour de nombreux problèmes de mathématiques ou de physique ou d'autre chose.

Il faut imaginer des dizaines de milliers de prompts différents. Il y a des milliers de solutions par prompt. Et tout cela effectué simultanément.

Au fur et à mesure que nous répétons ce processus, le modèle découvre par lui-même quelles séquences de jetons le conduisent aux bonnes réponses. Cela ne provient pas d'un annotateur humain. Le modèle "joue" tout seul, mais il sait toujours où il doit arriver.

Son objectif est de découvrir des séquences qui fonctionnent. Ce sont, en particulier, des séquences qui ne nécessitent pas de saut trop grand d'un jeton au suivant. Elles semblent fonctionner de manière fiable. Elles exploitent pleinement les connaissances du modèle.

Voilà en quoi consiste le processus d'*apprentissage par renforcement*. C'est essentiellement des essais et des sélections. Pour un prompt, nous essayons de nombreuses solutions, et les vérifions. Plus tard quand nous l'utiliserons dans la pratique, le LLM s'inspirera de ce qui a fonctionné.

Dans le contexte de ce qui a été dit précédemment, nous voyons que le modèle SFT, le modèle de réglage fin supervisé, reste utile, car il initialise le modèle un peu plus près des solutions correctes.

C'est une initialisation du modèle, dans le sens où il l'amène à prendre des solutions, à les écrire en détail ; peut-être à comprendre comment établir un système d'équations ; ou peut-être expliquer une solution verbalement. En tout cas, il nous rapproche des solutions correctes.

Mais c'est avec l'apprentissage renforcé que tout se met en place. Le LLM découvre réellement les solutions qui marchent pour lui, arrivent à la bonne réponse. Nous les retenons pour l'entraînement. Et ensuite le modèle s'améliore.

Voilà, dans les grandes lignes, la façon dont on entraîne les grands modèles de langage.

Leur entraînement est analogue à la façon dont les élèves travaillent dans leurs manuels.

La seule différence est que les élèves lisent les chapitres de leurs livres, et font tous les différents types d'exercices dans chaque chapitre de chaque livre.

Au lieu de cela, lorsque nous entraînons des IA, nous procédons presque étape par étape, selon le type d'étape.

1. Nous commençons par le pré-entraînement, qui, comme nous l'avons vu, est analogue à la lecture de l'ensemble du matériel explicatif.

Nous consultons tous les manuels simultanément. Nous lisons l'intégralité de l'exposé et nous essayons de construire une base de connaissances.

2. Ensuite, nous passons à l'étape de la SFT, qui consiste à examiner toutes les solutions fixes proposées par des experts humains. Elle est analogue aux différents types d'exercices totalement résolus dans tous les manuels.

Nous obtenons un modèle SFT capable d'imiter les experts, mais de manière plus ou moins aveugle. Il se contente de deviner au mieux, en essayant d'imiter statistiquement le comportement de l'expert.

3. Enfin, lors de la dernière étape, nous réalisons tous les exercices pratiques de l'apprentissage par renforcement ("reinforcement learning" ou RL avec l'abréviation anglaise).

Dans tous les manuels, nous faisons aussi les exercices pratiques. C'est ainsi que nous obtenons le modèle RL.

Ainsi, globalement, la façon dont nous formons les LLM est très similaire à celle que nous utilisons pour former les enfants.

02:27:47

## 18. DeepSeek-R1

Le point suivant que je voudrais souligner est que ces deux premières étapes – le pré-entraînement et le réglage fin supervisé – existent depuis des années et sont très courantes. Tout le monde les emploie, tous les fournisseurs de LLM et autres.

C'est la dernière étape, l'apprentissage renforcé, qui est à un stage précoce de son processus de développement, et n'est pas encore standardisé.

J'ai en fait omis de nombreux détails dans ce processus. L'idée générale est très simple : *c'est un apprentissage par essais et erreurs*. Mais il y a une multitude de détails et de nuances mathématiques à prendre en compte.

- comment choisir la solution qui est la meilleure
- comment on les utilise pour s'entraîner
- quelle est la distribution du prompt
- comment mettre en place concrètement l'entraînement pour que ça marche

Ainsi, obtenir les bons réglages — en apprentissage renforcé — dans la pratique n'a rien de trivial.

Beaucoup d'entreprises, par exemple OpenAI, mais aussi beaucoup d'autres fournisseurs de grands modèles de langage, ont expérimenté en interne le réglage par apprentissage renforcé pour les LLM depuis un certain temps. Mais elles n'en ont pas parlé publiquement. Tout cela se faisait plus ou moins à huis clos, à l'intérieur des entreprises.

C'est pourquoi l'article de DeepSeek publié en janvier 2025, fig. 88, a eu un tel retentissement. Il s'agit de la société chinoise DeepSeek AI<sup>82</sup>.

---

82. Hangzhou DeepSeek Artificial Intelligence Basic Technology Research Co., Ltd, basée à Hangzhou, dans la province du Zhejiang.

Il présente ouvertement les méthodes d'apprentissage et de réglage fin pour les grands modèles de langage, en soulignant à quel point elles sont cruciales pour ces modèles et comment elles révèlent de nombreuses capacités de raisonnement de la part des modèles.

**DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning**

DeepSeek-AI  
research@deepseek.com

**Abstract**

We introduce our first-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fine-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeek-R1 achieves performance comparable to OpenAI-o1-1217 on reasoning tasks. To support the research community, we open-source DeepSeek-R1-Zero, DeepSeek-R1, and six dense models (1.5B, 7B, 8B, 14B, 32B, 70B) distilled from DeepSeek-R1 based on Qwen and Llama.

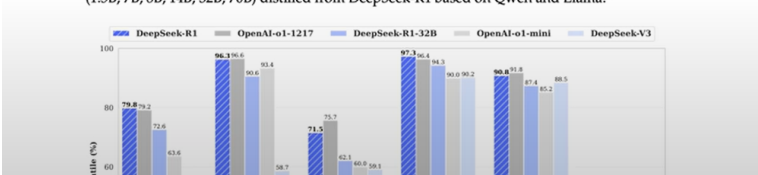


Figure 88: Article de DeepSeek qui a secoué la communauté des LLM.

Cet article a ravivé l'intérêt public pour l'utilisation de l'apprentissage renforcé (RL)<sup>83</sup> dans les LLM. Il a fourni de nombreux détails techniques essentiels pour reproduire leurs résultats et faire réellement fonctionner l'étape de RL pour les grands modèles de langage.

Nous allons voir brièvement ce qui se passe lorsqu'on applique correctement l'apprentissage par renforcement aux LLM, à quoi cela ressemble et ce que cela permet d'obtenir.

83. Parfois appelé "apprentissage par renforcement supervisé avec réglage fin".



Faisons défiler l'article jusqu'à sa figure 2. Elle montre l'amélioration de la capacité des modèles à résoudre les problèmes mathématiques.

Table 2 | Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

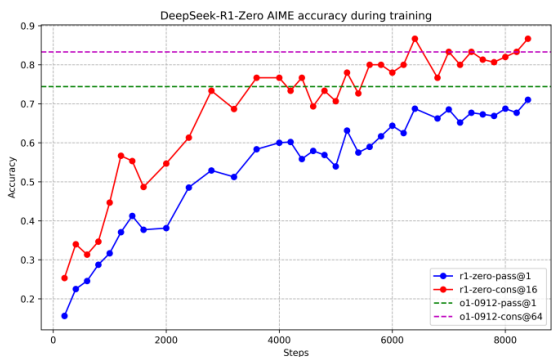


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

Figure 89: Précision dans la résolution de problèmes mathématiques selon la mesure d'exactitude de AIME.

Il s'agit de la précision avec laquelle les modèles résolvent des problèmes mathématiques selon la mesure d'exactitude de l'American Invitational Mathematics Examination.

Nous pouvons aller sur la page web d'AIME et observer les types de problèmes effectivement utilisés. Ce sont des problèmes de mathématiques simples.

Vous pouvez mettre la vidéo en pause et consulter l'article de DeepSeek à <https://arxiv.org/pdf/2501.12948>.

La figure 90 montre le genre de problèmes proposés par l'American Invitational Mathematics Examination que les modèles doivent résoudre.

On remarque qu'au début, les performances ne sont pas très bonnes, mais à mesure qu'on met à jour le modèle avec plusieurs milliers d'itérations, sa précision grimpe.

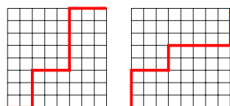
Les modèles résolvent les problèmes avec une exactitude de plus en plus élevée à mesure qu'ils apprennent par essais et erreurs sur un vaste ensemble de problèmes. *Les modèles finissent par découvrir eux-mêmes comment résoudre des problèmes mathématiques.*

6

2024 AIME I Solutions

**Problem 6:**

Consider the paths of length 16 that follow the lines from the lower left corner to the upper right corner on an  $8 \times 8$  grid. Find the number of such paths that change direction exactly four times, as in the examples shown below.

**Solution:****Answer (294):**

Any such path must consist of 8 steps to the right and 8 steps upward in some order. First consider those paths whose first step is to the right. All such paths start and end with a horizontal segment and include two vertical segments with a horizontal segment in between. The path is completely determined by the choice of the positions of the two vertical segments and the height of the middle horizontal segment. There are  $\binom{7}{2} = 21$  ways to select the positions of the two vertical segments and 7 ways to select the height of the middle horizontal segment. Thus there are  $21 \cdot 7 = 147$  paths that begin with a horizontal step. By symmetry, there are also 147 paths that begin with a vertical step, so the total number of paths is  $2 \cdot 147 = 294$ .

**Problem 7:**

Find the greatest possible real part of

$$(75 + 117i)z + \frac{96 + 144i}{z},$$

where  $z$  is a complex number with  $|z| = 4$ . Here  $i = \sqrt{-1}$ .

Figure 90: Exemples de problèmes de AIME. <sup>84</sup>

Mais ce qui est encore plus remarquable que les résultats *quantitatifs* — c'est-à-dire la capacité du modèle à résoudre ces problèmes avec une précision plus élevée —, ce sont les moyens *qualitatifs* par lesquels il y parvient.

En faisant défiler l'article jusqu'à la figure 3 (dans le papier DeepSeek), fig. 92, plus bas, p. 150, on remarque que plus l'optimisation avance, plus le modèle semble prendre de temps pour répondre.

Autrement dit, le modèle semble employer un plus grand nombre de jetons pour obtenir des résultats plus précis.

84. [https://www.ptsd.k12.pa.us/Downloads/2024\\_AIME\\_I\\_Solutions.pdf](https://www.ptsd.k12.pa.us/Downloads/2024_AIME_I_Solutions.pdf)

Il apprend donc à produire des solutions très longues.

Pourquoi ces solutions sont-elles si longues ?

Nous pouvons les examiner de manière qualitative ici (page 9 de l'article DeepSeek).

|                                                                                                        |
|--------------------------------------------------------------------------------------------------------|
| Question: If $a > 1$ , then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to |
| Response: <think>                                                                                      |
| To solve the equation $\sqrt{a - \sqrt{a + x}} = x$ , let's start by squaring both ...                 |
| $(\sqrt{a - \sqrt{a + x}})^2 = x^2 \implies a - \sqrt{a + x} = x^2$ .                                  |
| Rearrange to isolate the inner square root term:                                                       |
| $(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$  |
| ...                                                                                                    |
| Wait, wait. Wait. That's an aha moment I can flag here.                                                |
| Let's reevaluate this step-by-step to identify if the correct sum can be ...                           |
| We started with the equation:                                                                          |
| $\sqrt{a - \sqrt{a + x}} = x$                                                                          |
| First, let's square both sides:                                                                        |
| $a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$                                               |
| Next, I could square both sides again, treating the equation: ...                                      |
| ...                                                                                                    |

Table 3 | An interesting "aha moment" of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.

Figure 91: Un "moment aha" intéressant dans une version intermédiaire du papier de DeepSeek-R1-Zero.

Les solutions produites par le modèle deviennent très longues. La figure 91 montre une question et la réponse du modèle. Cela s'explique en partie par le fait que le modèle apprend - et *c'est une propriété émergente de la nouvelle optimisation* - qu'il est bénéfique, pour la résolution de problèmes, d'adopter ce type de comportement.

Il commence à faire des choses comme: "Attends, attends... Voilà un moment d'illumination que je peux signaler ici. Ré-évaluons cela, étape par étape, pour vérifier si la somme correcte peut être..." (fig. 91).

Que fait le modèle ici ? Il réévalue essentiellement les étapes de son raisonnement. Il a appris qu'il est plus efficace, pour accroître la précision, d'essayer de nombreuses idées, d'aborder le problème sous différents angles, de revenir en arrière, de reformuler, de retracer le cheminement.

Le modèle fait beaucoup de choses que vous et moi effectuons dans le processus de résolution de problèmes mathématiques. Il redécouvre comment on réfléchit, pas seulement ce qu'on écrit dans la solution.

*Aucun être humain ne peut coder directement ce genre de choses dans une réponse d'assistant idéale. C'est un comportement qui ne peut être découvert qu'au cours du processus d'apprentissage par renforcement.*

C'est simplement quelque chose qui fonctionne pour le modèle et qui améliore sa précision de résolution de problèmes.

Le modèle apprend ces chaînes de raisonnement internes, et *il s'agit d'une propriété émergente de l'optimisation*<sup>85</sup>.

C'est ce qui allonge le temps que prend le modèle pour répondre.

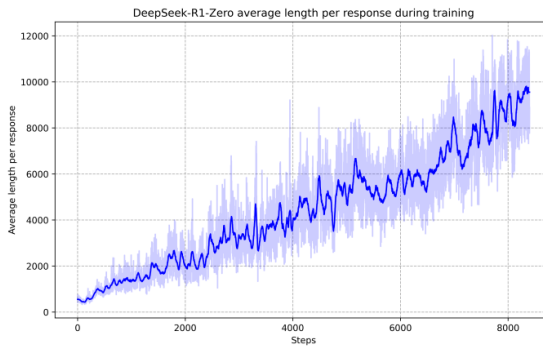


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

Figure 92: DeepSeek-R1-Zero prend de plus en plus de temps pour répondre.

Mais c'est aussi ce qui augmente la qualité des réponses aux problèmes par le LLM.

---

85. De manière générale, une « propriété émergente » est une structure ou un motif qui apparaît au cours d'un processus dynamique et qui ne peut pas être expliqué par des mathématiques simples. Un exemple ancien est celui d'un modèle de termites se déplaçant et mangeant des morceaux de bois [https://lapasserelle.com/courses/advanced\\_finance/documents/termites.mp4](https://lapasserelle.com/courses/advanced_finance/documents/termites.mp4)

Ce qui est incroyable ici, c'est que le modèle découvre des manières de penser. Il apprend ce que j'appelle des *stratégies cognitives*: comment manipuler un problème et comment l'aborder sous différents angles. Comment faire appel à des analogies ou tester divers types d'approches ? Comment essayer de nombreuses pistes, vérifier un résultat sous différentes perspectives ? En bref: comment résoudre des problèmes ?

Mais tout cela est découvert grâce à l'apprentissage par renforcement. C'est stupéfiant de voir ce comportement *émerger* naturellement au cours de l'optimisation, sans qu'il soit nécessaire de le coder explicitement. La seule chose que nous lui avons fournie, ce sont les bons résultats finaux. Tout le reste est apparu simplement parce qu'il a cherché à les résoudre correctement — ce qui est extraordinaire.

Revenons au problème d'Emilie, qui nous sert en quelque sorte d'éprouvette. Voyons à quoi cela ressemble pour ce type de modèle — ce que nous appelons un *modèle de raisonnement ou de réflexion* — lorsqu'il cherche à résoudre ce problème. Lorsque j'ai donné le problème d'Émilie à ChatGPT 4o, j'ai obtenu la réponse montrée figure 83, page 135.

Voyons donc ce qui se passe lorsque la même requête est donnée à un modèle de raisonnement ou de pensée. Il s'agit d'un modèle entraîné par apprentissage par renforcement. Le modèle décrit dans l'article DeepSeek-R1 est disponible sur [chat.deepseek.com](https://chat.deepseek.com).

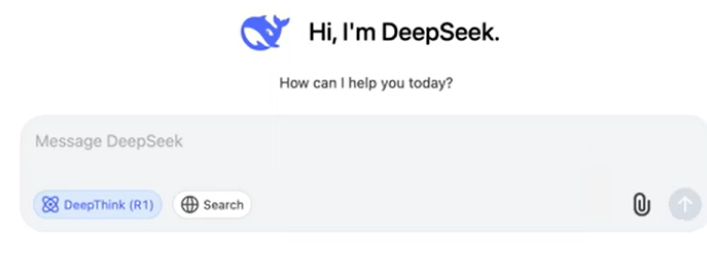


Figure 93: Page principale de [chat.deepseek.com](https://chat.deepseek.com)

C'est l'entreprise qui l'a développé et qui l'héberge.

Il faut s'assurer que le bouton DeepThink(R1) est activé (en bas à gauche de la figure 93) pour obtenir le modèle R1.

Nous pouvons coller le problème d'Émilie dans la boîte de requête et l'exécuter.

Voyons maintenant ce qui se passe et quel est le résultat produit par le modèle.

Très bien, voici ce qu'il affiche (voir figure 95, page suivante).

Et voici (figure 94 ci-dessous) ce que nous obtenions auparavant en utilisant une approche SFT classique, c'est-à-dire un réglage fin supervisé. Il s'agit du résultat obtenu après l'entraînement postérieur pur, sans la troisième étape appelée apprentissage par renforcement (RL).

Le modèle imitait alors simplement une solution experte.

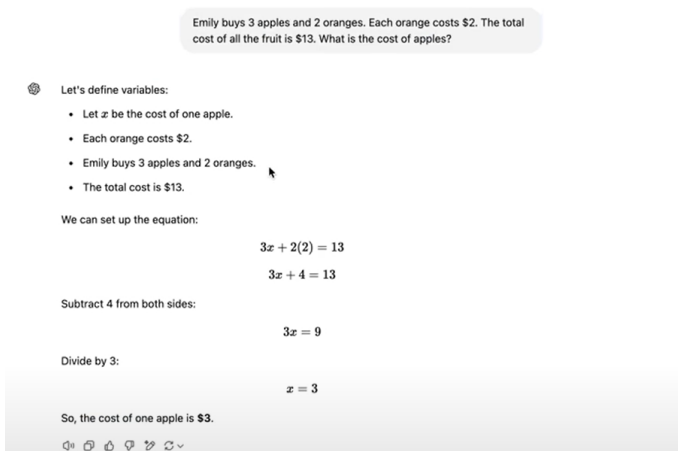


Figure 94: Résultat obtenu avec une simple approche SFT sans RL.

Et la figure 95, ci-dessous montre ce que donne un modèle RL.

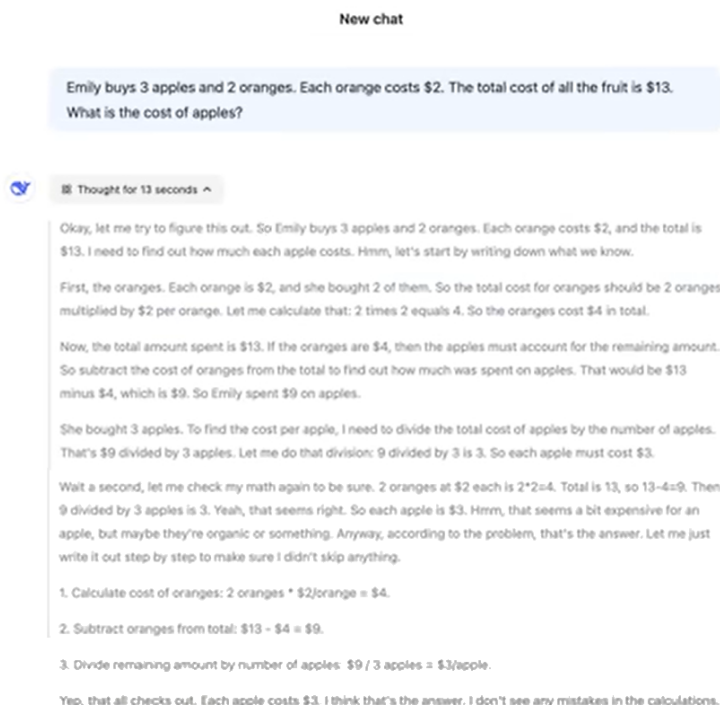


Figure 95: Résultat obtenu avec le modèle RL.

Le modèle commence lui-même par dire : "D'accord, voyons voir si je peux comprendre cela. Émilie achète 3 pommes et 2 oranges. Chaque orange coûte 2 \$ et le total est de 13 \$. Je dois trouver combien coûte chaque pomme, bla bla bla."

En lisant cela, on ne peut s'empêcher de penser que le modèle est en train de réfléchir. Il cherche véritablement la solution. Il en déduit qu'une pomme doit coûter 3 \$. Puis il ajoute: "Attendez une seconde, laissez-moi vérifier mes calculs pour en être sûr."

Ensuite, il tente une autre approche, sous un angle légèrement différent. Le modèle dit alors : "Oui, ça a l'air bon. Donc chaque pomme vaut 3 \$..." Après encore d'autres vérifications, il ajoute: "Ouais, ça baigne. Je pense que c'est la

bonne réponse, je ne vois pas d'erreur. Voyons s'il existe une autre manière d'aborder le problème, peut-être en établissant une équation. Fixons le coût d'une pomme à  $a$  dollar, alors trois pommes coûtent  $3a$ , bla bla bla... Oui, même résultat. Donc chaque pomme vaut bien 3 dollars. Tout est bon. Très bien: sûr de la validité de la démonstration." (Tout ce texte se trouve *dans la réponse de DeepSeek.*)<sup>86</sup>

*Ensuite DeepSeek fait ceci: une fois le processus de réflexion terminé (montré en gris), il rédige une belle solution compréhensible pour l'être humain (montrée en noir).*

Ainsi, cette première partie de la réponse de DeepSeek (en gris) concerne surtout l'aspect exactitude, tandis que la seconde partie (en noir) relève davantage de la présentation: le modèle met la solution en forme et encadre la réponse correcte à la fin.

*Ce qui est incroyable ici, c'est que nous voyons émerger le processus de réflexion du modèle, et c'est précisément le fruit du processus d'apprentissage par renforcement.*

C'est cela qui allonge les séquences de jetons (fig. 92) : le modèle réfléchit et teste plusieurs approches.

C'est aussi ce qui lui permet d'obtenir une plus grande précision dans la résolution de problèmes. C'est là que nous voyons apparaître des moments d'illumination, des stratégies variées et des idées nouvelles sur la manière d'assurer la justesse du résultat et de la démonstration (fig. 95).

---

86. Voir aussi la manière dont DeepSeek résout le problème des échelles. À première vue, c'est très impressionnant. La plupart des LLM se trompent, commettent des erreurs ou donnent des réponses absurdes, tandis que DeepSeek trouve la bonne réponse et présente un raisonnement et des calculs ressemblant à ceux d'un humain. [https://lapasserelle.com/LLM/ladder\\_problem\\_deepseek.pdf](https://lapasserelle.com/LLM/ladder_problem_deepseek.pdf)

Cependant, lorsqu'on examine attentivement la réponse de DeepSeek, on constate qu'elle est étrange, car elle expose des étapes intermédiaires erronées. Par exemple, elle affirme que les extrémités supérieures des deux échelles reposent sur les murs à une hauteur  $w$ , qui est aussi la largeur du couloir.

Il ne suffit pas de parsemer sa réponse de "Attendez que je vérifie", "Ça m'a l'air bon", "Ouais, ça baigne", pour en faire un authentique raisonnement logique.



Certaines personnes sont réticentes à l'idée de saisir des données sensibles sur [chat.deepseek.com](https://chat.deepseek.com), car il s'agit d'une entreprise chinoise. Cela engendre de la méfiance.

Cependant, DeepSeek R1 est un modèle qui a été *publié* par son concepteur: cela veut dire qu'il est open source et à poids ouverts (= publics). N'importe qui peut le télécharger et l'utiliser à sa guise.

Vous ne pourrez pas le faire fonctionner à pleine précision ou pleine capacité sur un MacBook ou un autre appareil local, car il s'agit d'un modèle très volumineux. Mais de nombreuses entreprises hébergent la version complète du modèle DeepSeek le plus puissant. L'une d'elles que j'apprécie est [together.ai](https://together.ai).

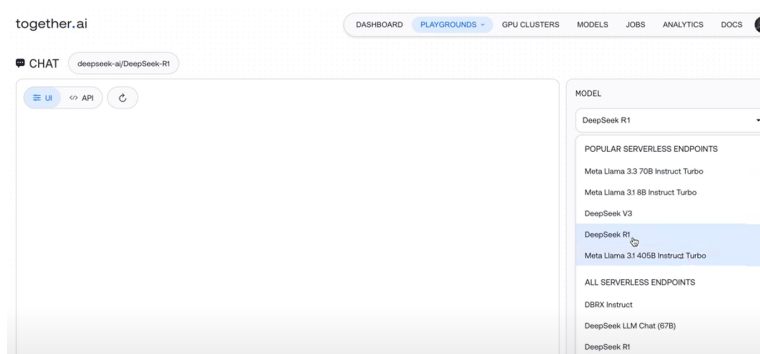


Figure 96: Bac à sable de [together.ai](https://together.ai). Vous pouvez choisir le LLM que vous voulez faire fonctionner.

Lorsque vous allez sur le site [together.ai](https://together.ai), vous vous inscrivez. Ensuite, allez dans la section "playground". Vous pouvez sélectionner à droite le modèle DeepSeek R1. Il y a un choix de modèles disponibles. Ce sont tous des modèles à la pointe de la technologie.

C'est similaire au bac à sable de "Hugging Face" pour faire des inférences, que nous avons déjà utilisé. Cependant, [together.ai](https://together.ai) héberge la plupart des modèles les plus récents et les plus performants. Sélectionnez donc DeepSeek R1.

Nous pouvons ignorer les paramètres qui nous sont proposés pour effectuer des réglages. Les paramètres par défaut iront très bien.

Nous entrons donc le problème d'Émilie dans la boîte de dialogue, sur la gauche (figure 97).

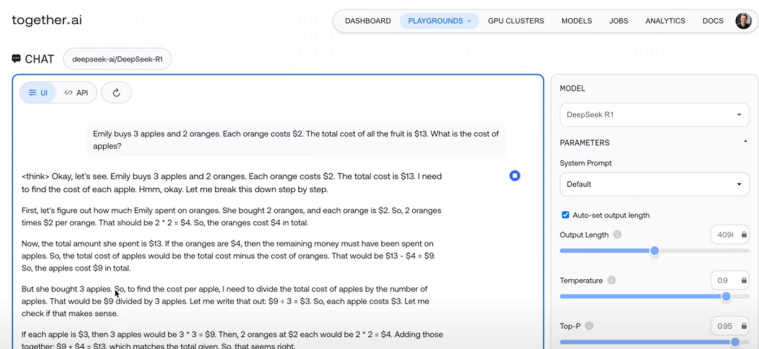


Figure 97: DeepSeek essayé via le bac à sable de [together.ai](https://together.ai).

Comme le modèle a été publié par DeepSeek, ce que nous obtenons ici (fig. 97) doit être essentiellement équivalent à ce que nous avons obtenu là-bas (fig. 95).

Cependant, comme nous le savons, en raison du caractère aléatoire de l'échantillonnage, nous allons obtenir un résultat légèrement différent. En principe, cela devrait néanmoins être identique en termes de puissance du modèle, et vous devriez pouvoir constater les mêmes aspects quantitatifs et qualitatifs.

Mais ce test de DeepSeek est désormais entièrement sous le contrôle d'une entreprise américaine. Voilà donc ce qu'est DeepSeek — ce qu'on appelle un *modèle de raisonnement*<sup>87</sup>.

87. Qualifier un modèle fondé sur l'apprentissage par renforcement de système « pensant » ou qui « raisonne » est une exagération. Il peut *imiter* certains aspects de la pensée humaine — DeepSeek, par exemple, utilise des expressions comme « aha », « attendez une seconde » ou « laissez-moi vérifier ». Mais cela ne signifie pas qu'il pense réellement.

Revenons à ChatGPT et regardons les différentes versions disponibles via [together.ai](https://together.ai). Un menu déroulant, figure 98 à gauche, propose un choix de modèles.

Certains d'entre eux, comme o1, o3-mini, o3-mini-high, etc., indiquent qu'ils « utilisent un raisonnement avancé ». Cela veut dire qu'ils ont eu un RL similaire à celui de DeepSeek R1. Je cite des déclarations publiques d'employés d'OpenAI.

Les modèles tels que GPT-4o ou GPT-4o-mini, accessibles dans la version gratuite (en haut de la liste), sont essentiellement des modèles SFT – sans RL.

---

Le comportement de DeepSeek rappelle le programme ELIZA de Joseph Weizenbaum, développé entre 1964 et 1966, et le programme SHRDLU de Terry Winograd, l'un des premiers systèmes de compréhension du langage naturel créés au MIT entre 1968 et 1970. Eux aussi donnaient l'impression de comprendre et de réfléchir.

ELIZA, inspiré par la psychothérapie développée par le psychologue américain Carl Rogers (1902-1987) dans les années 40, se contentait simplement de répéter ce que disait l'interlocuteur humain :

Humain : Je me sens maltraité par mon patron.

Machine : Vous vous sentez maltraité par votre patron ? Dites-m'en plus.

Humain : Il critique systématiquement ce que je fais.

Machine : Donc il critique systématiquement ce que vous faites ? etc.

SHRDLU, en revanche, était beaucoup plus sophistiqué : il permettait aux utilisateurs de dialoguer avec un ordinateur à propos d'actions menées dans un petit « monde de blocs », un environnement simulé composé d'objets géométriques simples. Le programme pouvait interpréter des commandes comme « place le bloc rouge sur le cube vert », raisonner sur leur contexte et répondre de manière cohérente en anglais.

SHRDLU fonctionnait dans un univers restreint et fondé sur des règles, mais sa compréhension reposait sur une logique déterministe et des heuristiques linguistiques.

Les modèles modernes comme DeepSeek simulent le raisonnement en imitant ce qu'ils ont appris dans leurs ensembles de données d'entraînement. Les deux créent une illusion de compréhension, bien que selon des paradigmes très différents : dans le cas de Winograd, des règles symboliques codées à la main ; dans celui des systèmes d'apprentissage par renforcement modernes, un apprentissage statistique de schémas.

En réalité, on pourrait soutenir que SHRDLU s'approchait davantage de la « pensée » que DeepSeek.

Quoi qu'il en soit, selon le transcripteur, les véritables modèles de raisonnement demeurent, en 2025, un objectif pour l'avenir.

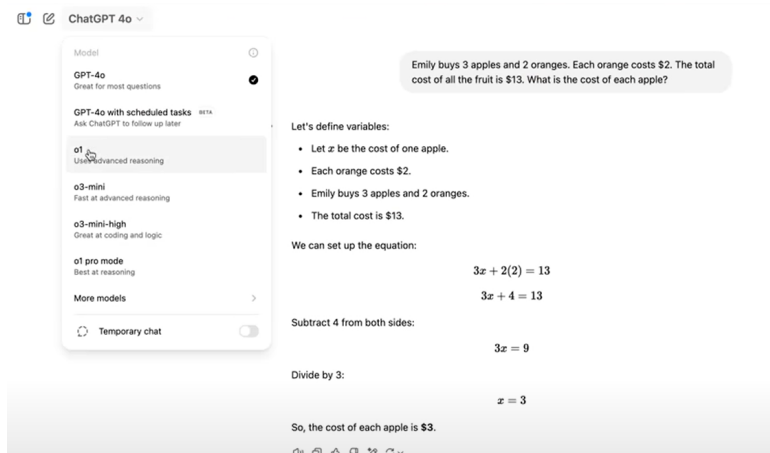


Figure 98: Retour à ChatGPT.

Les modèles « SFT purs », sans RL, ne produisent pas le type de raisonnement (avec toute la réflexion préliminaire) que l'on observe dans les modèles RL <sup>88</sup>.

Même s'il existe une petite part d'apprentissage par renforcement dans ces modèles (j'y reviendrai dans un instant), il s'agit principalement de modèles SFT purs. Comme précédemment, nous pouvons sélectionner un des modèles de raisonnement, par exemple o3-mini-high.

Ces modèles peuvent ne pas être accessibles sans abonnement à ChatGPT (20 \$ par mois, ou 200 \$ par mois pour certaines des versions les plus avancées).

---

88. Nous avons vu que les modèles d'apprentissage par renforcement, lorsqu'ils reçoivent une requête et son résultat final, testent de nombreuses solutions possibles, "réfléchissent tout haut". Ils écartent les solutions qui ne parviennent pas au bon résultat, puis, parmi celles qui marchent, retiennent — et intègrent à leur apprentissage — les « meilleures ».

Cependant, cela ne signifie pas que le modèle pense réellement. Tout professeur connaît des étudiants capables d'écrire l'énoncé du problème au début de leur copie, de donner la bonne réponse à la fin de la copie, et de remplir l'espace intermédiaire d'un tas d'aneries bien présentées.

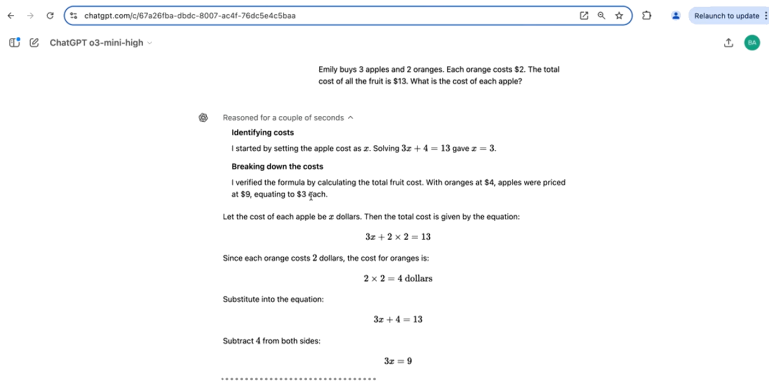


Figure 99: Sélection de ChatGPT-o3-mini.

Le modèle commence par afficher « reasoning », pendant qu'il "réfléchit" comme nous l'avons vu avec DeepSeek.

Cependant, ce que nous voyons avec GPT-o3-mini n'est pas exactement la même chose. Contrairement à DeepSeek, OpenAI choisit de ne pas montrer ses "pensées dites tout haut" dans l'interface web. L'entreprise n'en affiche qu'un court résumé.

OpenAI agit ainsi, selon toute vraisemblance, parce qu'elle s'inquiète de ce qu'on appelle le « risque de distillation ».

Il s'agit de la possibilité qu'une autre organisation récupère ces éléments de raisonnement et soit capable de reproduire une grande partie des performances du modèle simplement en utilisant ces chaînes de réflexion. Par conséquent, la "réflexion tout haut" n'est pas totalement montrée, seul un bref résumé est visible.

Vous n'obtenez donc pas exactement la même chose que ce que DeepSeek montre en matière de raisonnement proprement dit. Puis, OpenAI rédige la solution (figure 99).

À l'heure actuelle, en termes de performances, les modèles d'OpenAI (y compris ceux utilisant l'apprentissage par renforcement) et les modèles de DeepSeek sont globalement équi-

valents, à mon avis – même si, encore une fois, avec OpenAI, les détails internes du raisonnement ne sont pas complètement visibles. En réalité, il est difficile de trancher, car tout dépend du mode d'évaluation.

Cependant, si vous payez 200 \$ par mois à OpenAI, certains de leurs modèles semblent légèrement supérieurs. DeepSeek R1 reste néanmoins, pour le moment, un excellent choix de modèle de raisonnement, accessible depuis leur site web ou à partir d'autres plateformes, puisque ce modèle est à poids ouverts.

Voilà pour les modèles de raisonnement.

En résumé, nous avons parlé de l'apprentissage par renforcement et du fait que la “pensée” émerge au cours du processus d'optimisation, lorsque l'on applique le RL à de nombreux problèmes de mathématiques et de code disposant de solutions vérifiables. Ces modèles parviennent à la bonne réponse — par exemple « 3 » dans le problème d'Émilie — et présentent des raisonnements bien structurés.

Vous pouvez accéder à ces modèles de raisonnement, tels que DeepSeek ou tout autre modèle d'inférence, sur des plateformes comme [together.ai](https://together.ai). Vous choisissez le modèle avec lequel vous souhaitez travailler. Divers modèles de raisonnement – c'est-à-dire avec RL – sont également disponibles dans ChatGPT, notamment les modèles o1 ou o3.

Mais les modèles comme ChatGPT-4o, par exemple, ne sont pas des modèles de raisonnement. Vous devez les considérer principalement comme des modèles SFT.

Si vous avez une invite nécessitant un raisonnement avancé, il est recommandé d'utiliser un des modèles de raisonnement, ou au moins de les essayer.

Dans mon expérience, lorsqu'il s'agit d'une question simple — une question de connaissance, par exemple —, l'utilisation d'un modèle de raisonnement est excessive. Il n'est pas nécessaire de réfléchir pendant trente secondes pour répondre à une question purement factuelle. Dans ces cas-là, je me contente généralement d'utiliser ChatGPT-4o.

En pratique, environ 80 à 90 % de mes utilisations se font simplement avec GPT-4o. Et lorsque je tombe sur un problème particulièrement difficile — en mathématiques, en programmation, etc. — je fais appel aux modèles de raisonnement. Il faut attendre un peu plus longtemps, car ils prennent le temps de « réfléchir ». Vous pouvez accéder à ces modèles sur ChatGPT ou sur DeepSeek.

Signalons également [aistudio.google.com](https://aistudio.google.com)

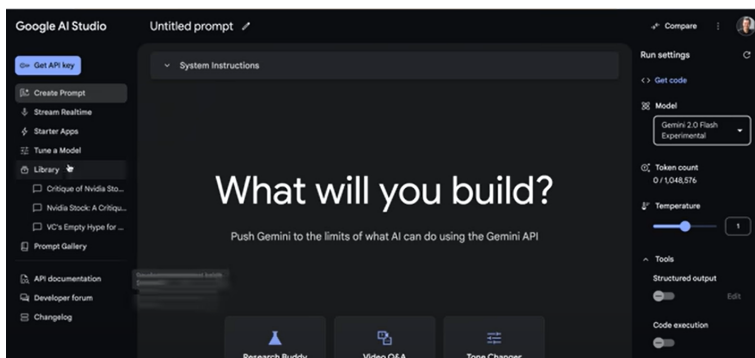


Figure 100: [aistudio.google.com](https://aistudio.google.com) .

L'interface est chargée et vraiment moche, car Google est incapable de faire ce genre de choses bien.

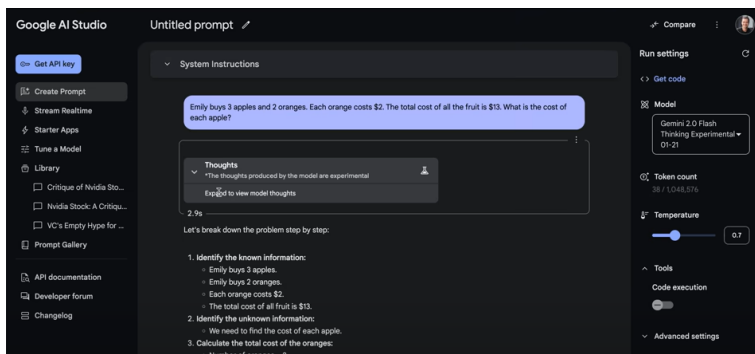


Figure 101: Le problème d'Émilie dans [aistudio.google.com](https://aistudio.google.com).

Sur le site [aistudio.google.com](https://aistudio.google.com), vous pouvez choisir entre plusieurs modèles. Si nous sélectionnons le modèle Gemini 2.0 Flash Thinking Experimental 01-21 (voir fig. 101, colonne de droite), il s'agit également d'une forme précoce de modèle de raisonnement.

Nous pouvons lui soumettre notre problème habituel et cliquer sur « run » (fig. 101). Le modèle de raisonnement effectuera un processus similaire à DeepSeek ou ChatGPT-o3-mini. Il aboutit lui aussi à la bonne réponse « 3 » obtenue à la fin d'un certain raisonnement. Ainsi, Gemini propose lui aussi un modèle de raisonnement.

Anthropic, pour le moment, n'en propose pas encore. L'apprentissage par renforcement (RL) représente aujourd'hui la frontière du développement de ces grands modèles de langage: c'est une nouvelle étape passionnante en informatique. Toutefois maîtriser les détails de sa mise en œuvre reste difficile. C'est pourquoi tous ces modèles de raisonnement sont encore expérimentaux au début de l'année 2025.

02:42:00

## 19. AlphaGo

Cela fait longtemps qu'on sait que l'apprentissage par renforcement est extrêmement puissant. Ce n'est pas un domaine nouveau de l'IA. Un exemple où cela a été utilisé il y a quelques années avec grand succès est le jeu de Go:

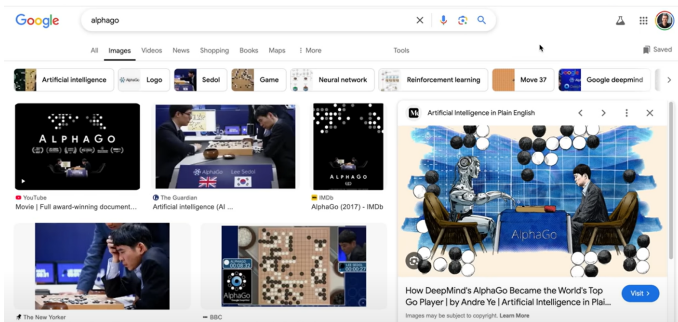


Figure 102: Recherche "alphago" sur Google.



DeepMind a développé le célèbre système AlphaGo. Vous pouvez visionner une vidéo sur le sujet, où le système apprend à jouer au Go contre des joueurs de haut niveau.

L'équipe de Google-DeepMind a expliqué dans un article dans Nature d'octobre 2017 comment ils ont fait pour amener AlphaGo au niveau des meilleurs joueurs mondiaux (et même mieux) <https://www.nature.com/articles/nature24270>

**Mastering the Game of Go without Human Knowledge**

David Silver\*, Julian Schrittwieser\*, Karen Simonyan\*, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, Demis Hassabis.

DeepMind, 5 New Street Square, London EC4A 3TW.

\*These authors contributed equally to this work.

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, *AlphaGo* became the first program to defeat a world champion in the game of Go. The tree search in *AlphaGo* evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here, we introduce an algorithm based solely on reinforcement learning, without human data, guidance, or domain knowledge beyond game rules. *AlphaGo* becomes its own teacher: a neural network is trained to predict *AlphaGo*'s own move selections and also the winner of *AlphaGo*'s games. This neural network improves the strength of tree search, re-

Figure 103: Article dans lequel l'équipe de Google-DeepMind explique comment AlphaGo s'est "auto entraîné".

Cet article présente un graphique particulièrement intéressant, qui, je pense, nous est déjà familier (fig. 104, graphique de gauche).

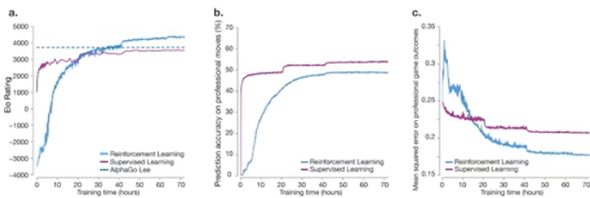


Figure 3: Empirical evaluation of *AlphaGo Zero*. **a** Performance of self-play reinforcement learning. The plot shows the performance of each MCTS player  $\alpha_{\theta_i}$  from each iteration  $i$  of reinforcement learning in *AlphaGo Zero*. Elo ratings were computed from evaluation games between different players, using 0.4 seconds of thinking time per move (see Methods). For comparison, a similar player trained by supervised learning from human data, using the KGS data-set, is also shown. **b** Prediction accuracy on human professional moves. The plot shows the accuracy of the

Figure 104: Amélioration progressive d'AlphaGo.

Dans les LLM on rencontre la même courbe, mais cette fois dans un domaine plus ouvert : celui de la résolution de problèmes arbitraires, au lieu d'un domaine fermé et spécifique comme le jeu de Go.

Durant l'apprentissage par renforcement (RL), le niveau ELO de la machine grimpe. La ligne horizontale en pointillés, fig. 104 (graphique de gauche), représente l'ELO de Lee Sedol, un joueur d'un niveau exceptionnel.

Le graphique compare la force d'un modèle entraîné par apprentissage supervisé (courbe violette) et celle d'un modèle entraîné par RL, c'est-à-dire en auto apprentissage (courbe bleu clair). Le modèle supervisé imite les joueurs experts humains. Si l'on dispose d'un grand nombre de parties jouées par des experts et qu'on cherche simplement à les reproduire, on s'améliore effectivement. Mais à un moment on plafonne. On ne parvient jamais à dépasser les meilleurs joueurs de Go, comme Lee Sedol, car on ne fait que les imiter. On ne peut fondamentalement pas aller au-delà des capacités humaines si l'on se limite à l'imitation de comportements humains.

L'utilisation d'une technique comparable à l'apprentissage par renforcement, en revanche, est nettement plus puissante. Dans le cas du jeu de Go, l'apprentissage par renforcement signifie que le système joue des coups qui, empiriquement et statistiquement, conduisent à la victoire.

AlphaGo, durant son entraînement, jouait contre lui-même. Il utilisait l'apprentissage par renforcement pour générer des simulations (« roll-outs »). Ainsi, le schéma de la figure 87 (page 141) s'applique parfaitement ici. Dans AlphaGo, il n'y a pas d'invite (« prompt »)<sup>89</sup>, puisqu'il s'agit d'un jeu de Go à règles fixes.

Le système teste un grand nombre de séquences de coups ; et les parties qui mènent à une victoire – c'est équivalent au bon résultat mathématique – sont renforcées : elles sont rendues plus probables.

---

89. On peut cependant considérer que la configuration du plateau, avant qu'AlphaGo ne joue son coup, constitue une sorte de prompt.

Le système apprend les séquences de mouvements qui, empiriquement et statistiquement, mènent à la victoire.

L'apprentissage par renforcement n'est pas limité par la performance humaine - une telle contrainte le freinerait. Le RL peut faire bien mieux et surpasser les meilleurs joueurs. L'équipe de DeepMind aurait pu prolonger l'entraînement, mais elle a choisi de l'arrêter à un certain niveau, car cela coûte cher<sup>90</sup>.

C'est cependant une démonstration très convaincante de l'apprentissage par renforcement. Nous commençons seulement à entrevoir des indices du schéma de la figure 104 (partie gauche) dans les LLM employant le RL.

La leçon à tirer de l'expérience AlphaGo est la suivante : *nous ne progresserons pas beaucoup en nous contentant d'imiter les experts humains*. Il faut aller au-delà, créer des environnements de jeu (fig. 87, p. 141) et laisser le système découvrir ses propres chemins de raisonnement, ou ses propres méthodes de résolution de problèmes, uniques et efficaces.

Sur ce point de l'unicité, remarquons que lorsqu'on pratique l'apprentissage par renforcement, rien n'empêche le système de s'écarter de la distribution correspondant à la manière dont les humains jouent au jeu. Lorsque nous cherchons "alphago" (fig. 102), l'une des suggestions est "move 37".

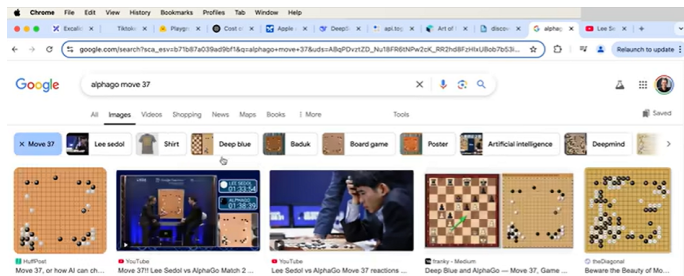


Figure 105: Recherche de "alphago move 37" sur Google.

90. AlphaGo n'a pas remporté le Championnat du monde officiel de Go, mais il a acquis une reconnaissance mondiale en battant le champion du monde en titre, Ke Jie, en mai 2017, lors du « Future of Go Summit » à Wuzhen, en Chine.

Le « coup 37 » dans AlphaGo fait référence à un coup célèbre joué par le programme lors d'une partie<sup>91</sup>. Aucun expert humain n'aurait effectué un tel coup. La probabilité qu'un joueur humain le joue avait été estimée à environ 1 sur 10 000. C'était donc un coup extrêmement rare, mais qui, rétrospectivement, s'est révélé brillant.

Il a démontré qu'AlphaGo, au cours de son processus d'apprentissage par renforcement, avait découvert une stratégie de jeu jusqu'alors inconnue des humains — mais qui, avec le recul, s'est révélée exceptionnelle.

Vous pouvez regarder la vidéo YouTube « Lee Sedol versus AlphaGo – Move 37 reactions and analysis » à cette adresse: <https://youtu.be/HT-UZkiOLv8>.

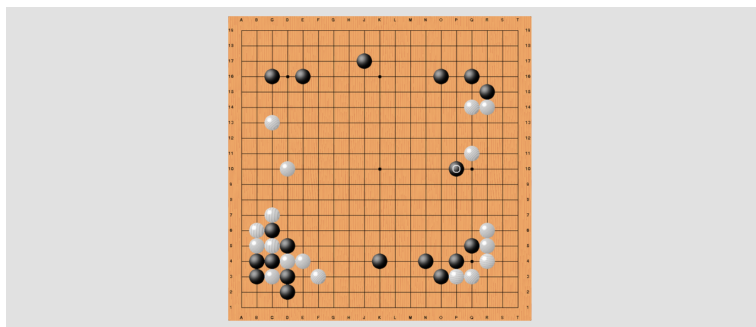


Figure 106: Disposition du plateau juste après qu'AlphaGo (qui a les pierres noires) a joué le coup 37 (indiqué par un cercle). La pierre noire cerclée, aux coordonnées « Q-10 », correspond au célèbre coup 37 de la deuxième partie contre Lee Sedol.

C'était un coup très surprenant — j'ai honnêtement cru à une erreur lorsque je l'ai vu. Les gens ont été stupéfaits, car c'était un coup qu'aucun humain ne jouerait normalement.

---

91. Le « coup 37 » joué par AlphaGo a eu lieu pendant la deuxième partie du match contre Lee Sedol, le 9 mars 2016. Ce coup est largement considéré comme un moment marquant dans l'histoire de l'intelligence artificielle et du jeu de Go, ayant stupéfié les commentateurs et les experts par son originalité et sa profondeur.

AlphaGo l'a joué parce que, pendant son entraînement, ce coup avait fini par être évalué comme un bon coup.

Ce paradigme s'étend aux modèles de langage (voir fig. 87, p. 141). Comment ça marche concrètement ? C'est difficile à décrire. Que signifie, en effet, résoudre des problèmes d'une manière à laquelle même les humains ne peuvent pas parvenir ? Comment peut-on construire un raisonnement ou une pensée supérieurs à l'esprit humain ?

Peut-être cela signifie-t-il découvrir des analogies que les humains ne peuvent pas imaginer. Peut-être s'agit-il d'un nouveau mode de réflexion. C'est difficile à concevoir. Peut-être même que cela revient à créer une langue entièrement nouvelle. Peut-être le modèle invente-t-il son propre langage, plus efficace pour raisonner, car, contrairement à l'humain, il n'est pas contraint de rester dans les limites des langues de l'humanité<sup>92</sup>.

En principe, le comportement du système devient beaucoup moins défini (que dans le cadre de la syntaxe d'une langue). Il reste libre de procéder comme bon lui semble. Il peut aussi, lentement, s'éloigner de la distribution de ses données d'entraînement.

Mais tout cela ne peut se produire que si nous disposons d'un ensemble de problèmes très vaste et diversifié, dans lequel la stratégie peut être raffinée et perfectionnée.

Actuellement, de nombreuses recherches de pointe sur les LLM s'attachent justement à concevoir de telles distributions d'instructions ou de "prompts" vastes et variées ; ce sont autant d'environnements de jeu au sein desquels les grands modèles de langage peuvent s'exercer à penser et construire des raisonnements.

---

92. L'enthousiasme d'Andrej Karpathy pour "la façon mystérieuse" dont fonctionne l'apprentissage par renforcement dans l'entraînement d'AlphaGo et des LLM va au-delà du raisonnable. Il faut garder à l'esprit que, dans les LLM, le RL reste une forme d'ajustement automatique en mode auto-supervisé. En pratique, cette méthode sert surtout à encourager le modèle à imiter de bonnes réponses, sans pour autant lui conférer de véritables capacités de raisonnement logique – et encore moins de construction d'un nouveau langage !

C'est un peu comme écrire des exercices d'entraînement (voir fig. 81, p. 131). Nous devons créer des exercices pratiques couvrant tous les domaines de la connaissance.

Si nous disposons d'un très grand nombre d'exercices d'entraînement, les modèles pourront mettre en œuvre l'apprentissage par renforcement, apprendre grâce à cette méthode et produire des graphiques semblables à la figure 104 (partie gauche), mais cette fois dans le domaine de la pensée généraliste, pas seulement au jeu de Go.

02:48:26

## 20. Apprentissage renforcé par feedback humain (RLHF)

Il y a une section particulière de l'apprentissage par renforcement dont il faut parler : c'est la mise en œuvre du RL dans des domaines non vérifiables<sup>93</sup>.

Jusqu'à présent, tous les problèmes que nous avons examinés appartiennent à ce qu'on appelle des *domaines vérifiables* — c'est-à-dire des situations dans lesquelles il est très facile d'évaluer une solution proposée en la comparant à une réponse concrète qu'on sait correcte. Dans le problème d'Émilie, par exemple, la réponse est « 3 ».

Dans les domaines vérifiables, nous pouvons demander au modèle de vérifier l'égalité avec la réponse que nous savons correcte (le « 3 » ci-dessous) ; ou nous pouvons faire appel à ce qu'on appelle un « juge LLM » [une procédure qui reste automatique, nous allons l'expliquer].

---

93. Dans les *domaines vérifiables* — par exemple les maths, les échecs, le Go ou la génération de code —, les réponses du modèle peuvent être vérifiées ou évaluées de manière objective (et automatiquement par un programme). Il est donc facile de construire des fonctions de récompense efficaces pour ajuster le réseau de neurones.

Les *domaines non vérifiables* sont ceux où il n'existe pas de vérité objective — ou bien où la réussite ne peut pas être évaluée automatiquement par un programme. Ces domaines incluent l'écriture créative, le raisonnement sur des problèmes, les questions morales, etc.

Le « juge LLM » examine une solution, en déduit la réponse, puis évalue dans quelle mesure cette solution est cohérente avec la bonne réponse. Empiriquement, les LLM disposent aujourd'hui de capacités suffisantes pour effectuer ce type de vérification dans les domaines vérifiables, et ce, avec une fiabilité raisonnable.

Nous pouvons donc appliquer ces mêmes techniques.

En tout cas, dans les domaines vérifiables, nous disposons d'une réponse concrète ; il s'agit simplement de vérifier si les solutions correspondent à cette réponse. Cela peut être fait automatiquement, sans intervention humaine.

Le problème, c'est que cette stratégie ne peut pas s'appliquer aux domaines dits *non vérifiables*. Ce sont, par exemple, l'écriture créative, fabriquer une blague sur les pélicans, composer un poème ou résumer un paragraphe. Dans ces contextes, il devient difficile d'évaluer précisément la qualité des différentes solutions produites.

Si l'on demande au LLM, par exemple, d'écrire une blague sur les pélicans, on peut bien sûr générer de nombreuses blagues différentes — et c'est très bien ainsi.

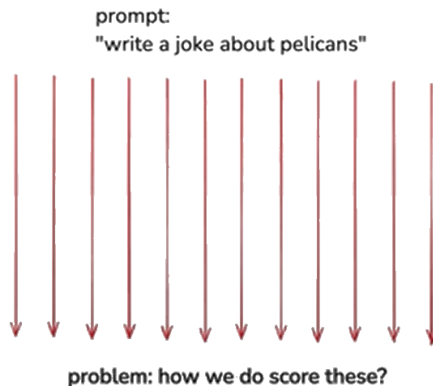


Figure 107: Génération de plusieurs blagues sur les pélicans. Comment évaluer les différentes blagues ?

Par exemple, nous pouvons aller sur ChatGPT et lui demander de fabriquer une blague sur les pélicans. Nous lui donnons le prompt: "Écris une blague super drôle sur les pélicans."



Figure 108: Blague sur les pélicans générée par ChatGPT.

Write a super funny joke about pelicans.

Why do pelicans always carry so much stuff in their beaks ?  
Because they don't bellican in backpacks 😊"

OK [ce n'est pas très drôle]. On peut essayer autre chose:

Write a super funny joke about pelicans.

Why don't pelicans ever pay for their drinks ?  
Because they always bill it to someone else! 😊😊

Haha... OK, ces modèles ne sont manifestement pas doués pour l'humour.

En réalité, je trouve cela assez fascinant, car je pense que l'humour est en secret quelque chose de très difficile. Et le modèle en a, je crois, la capacité.

Quoi qu'il en soit, on peut facilement imaginer créer beaucoup de blagues. Le problème auquel nous faisons face est le suivant : comment les évaluer ?

En principe, on pourrait bien sûr demander à un humain de les examiner toutes, comme je viens de le faire à l'instant. Le problème, c'est que si l'on fait de l'apprentissage par renforcement, on effectue des milliers de mises à jour.



Et pour chaque mise à jour, on veut examiner, disons, des milliers d'invites. Et pour chaque invite, on veut potentiellement observer des centaines ou des milliers de générations différentes.

Il y en a tout simplement trop pour qu'on puisse toutes les examiner. En principe, on pourrait demander à un humain de les inspecter, de leur attribuer une note et de décider que, d'accord, peut-être celle-ci est drôle, et celle-là est drôle, et celle-ci aussi. . . Et nous pourrions entraîner le modèle à partir de ces exemples pour qu'il devienne un peu meilleur dans l'art de faire des blagues – au moins sur les pélicans.

Le problème, c'est que cela exige beaucoup trop de temps humain. Ce n'est pas une stratégie "scalable". Nous avons besoin d'une méthode automatique pour le faire. Une solution de ce type a été proposée dans l'article montré figure 109, qui présentait ce qu'on appelle l'apprentissage par renforcement à partir de feedback humain (RLHF).

C'était un article d'OpenAI à l'époque, mais nombre de ses auteurs sont depuis cofondateurs d'Anthropic. L'article proposait une approche permettant de réaliser un apprentissage par renforcement dans des domaines non vérifiables.

---

### Fine-Tuning Language Models from Human Preferences

---

Daniel M. Ziegler\* Nisan Stiennon\* Jeffrey Wu Tom B. Brown  
Alec Radford Dario Amodei Paul Christiano Geoffrey Irving

OpenAI

{dmz,nisan,jeffwu,tom,alec,damodei,paul,irving}@openai.com

#### Abstract

Reward learning enables the application of reinforcement learning (RL) to tasks where reward is defined by human judgment, building a model of reward by asking humans questions. Most work on reward learning has used simulated environments, but complex information about values is often expressed in natural language, and we believe reward learning for language is a key to making RL practical and safe for real-world tasks. In this paper, we build on advances in generative pretraining of language models to apply reward learning to four natural language tasks: continuing text with positive sentiment or physically descriptive language, and summarization tasks on the T1-DR

plex goals to AI agents are likely to both involve and require natural language, which is a rich medium for expressing value-laden concepts. Natural language is particularly important when an agent must communicate back to a human to help provide a more accurate supervisory signal (Irving et al., 2018; Christiano et al., 2018; Leike et al., 2018).

Natural language processing has seen substantial recent advances. One successful method has been to pretrain a large generative language model on a corpus of unsupervised data, then fine-tune the model for supervised NLP tasks (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Khandelwal et al., 2019). This method often substantially outperforms training on the supervised datasets from scratch, and a single pretrained language model often can be fine-tuned for state of the art performance on many different super-

: [cs.CL] 8 Jan 2020

Figure 109: Article d'OpenAI sur le RLHF de janvier 2020

<https://arxiv.org/pdf/1909.08593v2>

Voyons comment cela fonctionne. La figure 110 schématise les idées principales

Naive approach:

Run RL as usual, of 1,000 updates of 1,000 prompts of 1,000 rollouts.  
(cost: 1,000,000,000 scores from humans)

RLHF approach:

STEP 1:

Take 1,000 prompts, get 5 rollouts, order them from best to worst  
(cost: 5,000 scores from humans)

STEP 2:

Train a neural net simulator of human preferences ("reward model")

STEP 3:

Run RL as usual, but using the simulator instead of actual humans

Figure 110: Idées clé dans le RLHF.

L'approche « native » serait la suivante : si nous disposions d'une quantité infinie de temps humain, nous pourrions parfaitement exécuter un apprentissage par renforcement dans ces domaines non vérifiés. Par exemple, nous pourrions exécuter le RL comme d'habitude. Supposons que je veuille effectuer 1000 mises à jour, chacune portant sur 1000 invites. Et pour chaque invite, nous aurions 1000 déroulements ("rollouts") à évaluer. Nous pourrions exécuter le RL (cf. haut de la figure 110).

Mais avec ce processus, je devrais demander à des humains d'évaluer une blague un milliard de fois. Ce n'est pas possible.

Nous voulons plutôt adopter l'approche appelée « apprentissage par renforcement à partir de feedback humain » (RLHF). Dans cette approche, l'astuce fondamentale repose sur la notion d'indirection : nous faisons intervenir les humains seulement un peu. La façon dont nous "trichons" consiste à entraîner tout un réseau de neurones distinct que nous appelons un *modèle de récompense*.

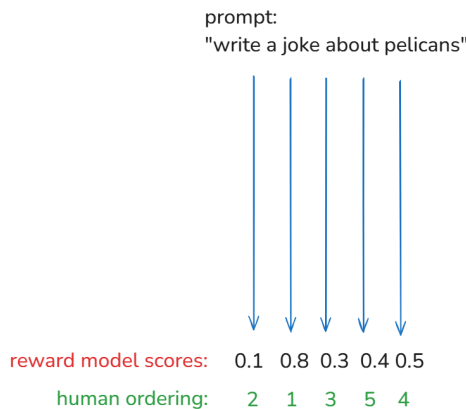
*Ce réseau de neurones va imiter les évaluations humaines.*

Nous allons donc demander à des humains d'évaluer les "roll-outs". Ensuite, nous allons imiter les scores humains à l'aide du modèle de récompense. Ce réseau de neurones deviendra une sorte de simulateur des préférences humaines. Une fois que nous disposons d'un simulateur neuronal, nous pouvons effectuer de l'apprentissage par renforcement à partir de celui-ci. Au lieu que ce soit un véritable humain qui évalue une blague, ce sera un humain simulé.

Dès que nous avons un simulateur, nous sommes opérationnels, car nous pouvons le consulter autant de fois que nous le voulons. C'est un processus entièrement automatique. Nous pouvons désormais exécuter un apprentissage par renforcement fondé sur le simulateur.

Le simulateur, comme on peut s’y attendre, ne sera pas un humain parfait. Mais s’il est au moins statistiquement proche du jugement humain, on peut s’attendre à ce qu’il produise des résultats pertinents. Et en pratique, c’est bien ce qui se passe. Une fois le simulateur établi, nous pouvons exécuter le processus d’apprentissage par renforcement, et tout fonctionne très bien.

La figure 111 schématise à quoi ressemble ce processus. Les détails n'ont pas une importance capitale.



On entre : « Écris une blague sur les pélicans ». Nous effectuons cinq essais distincts. Ce sont cinq blagues différentes, comme celles page 170.

On demande à un humain de classer ces blagues de la meilleure à la pire. La ligne de cinq nombres verts, fig. 111, représente un classement. L'humain a estimé que la deuxième blague était la meilleure et la plus drôle: elle est donc classée n°1. Celle située tout à gauche est n°2, puis viennent la blague 3 et la blague 4. Enfin, l'avant-dernière blague, à droite, est la n°5, autrement dit la pire.

Nous demandons aux humains d'établir un ordre plutôt que de donner directement des notes, car c'est une tâche plus simple. Il est en effet plus facile pour un humain de hiérarchiser que d'attribuer des scores numériques. Ce classement constitue la supervision du modèle. L'humain a ordonné les blagues: c'est sa contribution au processus d'entraînement.

Ensuite, séparément, nous demandons au modèle de récompense d'évaluer à son tour ces mêmes blagues selon sa propre estimation.

Le modèle de récompense est un réseau de neurones complètement séparé. C'est probablement aussi un transformeur, mais ce n'est pas un modèle de langage au sens où il ne génère pas de texte varié, etc. C'est simplement un modèle d'évaluation. Ce modèle de récompense prend deux éléments en entrée. D'abord, l'invite. Ensuite, il sélectionne une blague candidate. Ce sont donc les deux entrées du modèle de récompense. Par exemple, le modèle de récompense prendrait l'invite et la blague ci-dessous comme entrées.

Write a super funny joke about pelicans.

Why don't pelicans ever pay for their drinks ?

Because they always bill it to someone else !☺☺

L'output d'un modèle de récompense est un nombre unique. Ce nombre est interprété comme une note ou un score. Par exemple, il peut varier de zéro à un.

Un score de zéro correspond à la pire note, et un score de un correspond à la meilleure. Figure 111, l'avant-dernière ligne du bas montre des exemples de ce qu'un modèle de récompense hypothétique, à une certaine étape de son entraînement, attribuerait à ces blagues. Attribuer une note de 0,1 est une très faible évaluation ; 0,8 est une note très élevée, etc..

Nous comparons maintenant les scores donnés par le modèle de récompense avec l'ordre établi par l'humain. Il existe une méthode mathématique précise pour calculer une fonction de perte, établir une correspondance et mettre à jour le modèle en conséquence. Mais l'idée intuitive est la suivante: par exemple, dans la figure 111, pour la deuxième blague, l'humain a estimé qu'elle était la plus drôle, et le modèle était plutôt d'accord: 0,8 est un score relativement élevé. Cependant, ce score devrait être encore plus haut. Ainsi, après une mise à jour du modèle de récompense, on s'attend à ce que ce score augmente, disons, jusqu'à 0,81 environ.

Pour la blague située tout à gauche, l'humain et le modèle de récompense sont en fort désaccord: l'humain pensait qu'elle méritait la deuxième place, mais le modèle ne lui attribue qu'un score de 0,1. Le score de cette blague doit donc être considérablement augmenté. Après une mise à jour, en plus de la supervision humaine, il pourrait grimper à 0,15 environ.

Si l'on regarde maintenant la deuxième blague en partant de la droite, l'humain l'a jugée comme la pire, mais le modèle lui a en réalité attribué une note assez élevée. On pourrait donc s'attendre, après mise à jour, à ce que ce score redescende vers 0,3 ou 0,35.

Fondamentalement, nous faisons ici la même chose que précédemment lorsque nous mettions à jour les paramètres du réseau de neurones pendant la phase de pré-entraînement : nous ajustons légèrement les prédictions du modèle à l'aide d'un processus d'apprentissage neuronal. Le but est d'amener les scores du modèle de récompense à être cohérents avec le classement établi par les humains.

À mesure que nous mettons à jour le modèle de récompense à partir de données humaines, il devient un simulateur

de plus en plus précis des notes et des classements fournis par les humains. Il finit donc par devenir un simulateur des préférences humaines, grâce auquel nous pouvons ensuite exécuter un apprentissage par renforcement.

Mais, point crucial : nous ne demandons pas aux humains d'examiner un milliard de blagues. Nous nous contentons peut-être d'environ mille invites, avec cinq roll-outs chacune, soit en tout environ 5000 blagues que les humains doivent évaluer. Ils se contentent simplement de les classer, puis nous entraînons le modèle pour qu'il soit cohérent avec cet ordre.

Je passe sur les détails mathématiques, car l'idée principale est simplement de comprendre que ce modèle de récompense nous fournit des scores, et que nous avons un moyen de l'entraîner pour qu'il soit aligné sur les classements humains.

C'est ainsi que fonctionne le RLHF : *nous entraînons des simulateurs du comportement humain, puis nous faisons de l'apprentissage par renforcement à partir de ces simulateurs.*

02:57:58

## 21. Avantages et inconvénients du RLHF

Je voudrais maintenant parler des avantages et des inconvénients de l'apprentissage par renforcement à partir de feedback humain (RLHF).

### RLHF upside

We can run RL, in arbitrary domains! (even the unverifiable ones)  
This (empirically) improves the performance of the model, possibly due to the "discriminator - generator gap":

In many cases, it is much easier to discriminate than to generate.

e.g. "Write a poem" vs. "Which of these 5 poems is best?"

Figure 112: Avantages du RLHF <sup>94</sup>.

94. Extrait du tableau blanc Excalidraw utilisé par le speaker, voir la note 2 en bas de la page 2.

Du côté des avantages, tout d'abord cette méthode nous permet de pratiquer du RL – un ensemble de techniques dont on sait qu'elles sont extrêmement puissantes. La création d'un "annotateur artificiel" nous permet d'appliquer le RL dans des domaines arbitraires, y compris ceux non vérifiables, par exemple les résumés, la poésie, les blagues ou toute autre forme d'écriture créative, contrairement aux domaines vérifiables comme les mathématiques ou le code, etc.

On observe empiriquement que le RLHF permet d'améliorer les performances du modèle. Lorsqu'on met correctement en œuvre le RLHF, les modèles obtenus s'avèrent effectivement un peu meilleurs.

La raison de cette amélioration n'est pas entièrement claire.

Mon intuition est que cela découle de ce qu'on appelle le "discriminator - generator gap", l'écart entre classement et notation. Cela signifie que, dans de nombreux cas, il est nettement plus facile pour un humain de classer que de noter.

Un exemple particulièrement parlant est celui de l'ajustement fin supervisé (SFT). Dans ce cas, nous demandons à des humains de produire la réponse idéale d'un assistant.

Dans bien des situations, comme je l'ai montré — voir la section 10, « Données pour le post-entraînement (conversations) », page 67 — la réponse idéale est simple à rédiger.

Mais dans beaucoup d'autres cas, ce n'est pas si évident. Par exemple, dans la rédaction de résumés, de poèmes ou de blagues, comment, en tant qu'annotateur humain, êtes-vous censé produire la réponse parfaite ?<sup>95</sup>

Cela demande de la créativité humaine.

Le RLHF contourne le problème, car nous posons aux personnes qui annotent les données une question beaucoup plus simple. Elles n'ont pas à écrire directement des poèmes ; on

---

95. C'est ce que font les bons enseignants chaque jour. Ils expliquent également aux étudiants pourquoi une façon d'écrire un texte est meilleure qu'une autre. Voir William Zinsser, *On Writing Well*, Harper Perennial, 2006.

leur présente simplement cinq poèmes produits par le modèle et on leur demande de les classer. C'est une tâche bien plus facile pour un annotateur humain.

Je pense que cela permet de produire des données beaucoup plus précises, car nous ne demandons pas aux gens d'accomplir la tâche de génération, qui peut être extrêmement difficile. Nous ne leur demandons pas de faire de l'écriture créative ; nous leur demandons simplement de distinguer entre différents textes créatifs et d'identifier ceux qui sont les meilleurs. C'est le signal que les humains fournissent: le classement. C'est leur contribution au système.

Ensuite, le système, dans le cadre du RLHF, découvre les types de réponses qui seraient bien évaluées par les humains. Cette étape d'indirection permet aux modèles de s'améliorer un peu. C'est l'un des avantages du RLHF: il nous permet d'exécuter de l'apprentissage par renforcement, et dans la pratique, il conduit à de meilleurs modèles. Il permet également aux humains d'apporter leur supervision, même sans avoir à accomplir des tâches très difficiles, comme la rédaction de réponses idéales.

Malheureusement, le RLHF a aussi de gros défauts. Et on ne peut l'utiliser qu'avec parcimonie.

#### RLHF downside

We are doing RL with respect to a lossy simulation of humans. It might be misleading!

Even more subtle:

RL discovers ways to "game" the model.

It discovers "adversarial examples" of the reward model.

E.g. after 1,000 updates, the top joke about pelicans is not the banger you want, but something totally non-sensical like "the the the the the the the the".

Figure 113: Inconvénients du RLHF..



Le principal inconvénient, c'est que nous ne faisons pas de l'apprentissage par renforcement avec l'aide de vrais humains et leur jugement, mais à l'aide d'une simulation imparfaite de ce jugement. Cette simulation provient d'un réseau neuronal générant des scores. Elle ne reflète pas toujours parfaitement l'opinion d'un être humain doté d'un cerveau, dans tous les cas possibles.

Il existe un défaut encore plus subtil et plus insidieux, qui limite fortement le RLHF en tant que technique extensible à des systèmes véritablement intelligents. L'apprentissage par renforcement est en effet extrêmement bon pour découvrir des moyens de « berner » la simulation.

Le modèle de récompense que nous construisons (figure 111, page 173), celui qui attribue les scores... ces modèles sont des transformeurs, d'immenses réseaux neuronaux avec des milliards de paramètres et imitant le comportement humain. Mais ils le font d'une façon purement simulée.

Le problème, c'est que ces systèmes sont très complexes: un milliard de paramètres produisent un seul score. Il s'avère qu'il existe des moyens de tromper ces modèles. On peut trouver des types d'input (= entrées, ou prompts) qui ne faisaient pas partie de leur ensemble d'entraînement, mais qui obtiennent des scores très élevés – alors qu'ils ne devraient pas.

Très souvent, lorsqu'on exécute du RLHF pendant longtemps — par exemple sur un millier de mises à jour, ce qui est déjà beaucoup —, on pourrait s'attendre à ce que les blagues produites sur les pélicans deviennent meilleures, plus efficaces, qu'elles « fassent mouche ». Mais ce n'est pas ce qui se passe.

En réalité, durant les premières centaines d'itérations, les blagues sur les pélicans s'améliorent, deviennent un peu plus drôles. Puis elles chutent brusquement. Et l'on commence à obtenir des résultats complètement aberrants. Par exemple, la meilleure « blague » sur les pélicans finit par devenir : « the the the the the the the the ». C'est évidemment dénué de sens. Que se passe-t-il pour que la simulation finisse par trouver que c'est la meilleure blague ?

Lorsque vous prenez « the the the the the the... » et que vous l'insérez dans votre modèle de récompense, vous vous attendez à un score de zéro. Mais, le modèle de récompense trouve cette blague très drôle<sup>96</sup>. Il vous dira que « the the the... » mérite un score de 1,0.

Cela provient du fait que ces modèles ne sont que des simulations d'humains ; ce sont d'immenses réseaux de neurones, et il est possible de trouver des entrées situées dans les zones périphériques de l'espace d'entrée qui produisent des résultats absurdes. Ces exemples sont appelés "exemples adverses". Je ne vais pas entrer ici dans les détails. Ce sont des entrées qui trompent le modèle, des petites variations qui se glissent dans les failles du réseau et provoquent des résultats incohérents.

Voici ce que vous pourriez imaginer faire : vous dites "D'accord, 'the the the...' n'a clairement pas un score de 1, c'est évidemment une très mauvaise blague." Prenons donc "the the the the the..." et ajoutons-la à l'ensemble d'entraînement, en lui attribuant un classement extrêmement bas — par exemple la note 5 dans notre cas. Votre modèle apprendra alors que "the the the the..." doit recevoir un score très faible, et il lui donnera un score de zéro.

Le problème, c'est qu'il existera toujours un nombre infini d'exemples adverses absurdes cachés dans le modèle. Même si vous répétez ce processus de nombreuses fois, en ajoutant sans cesse des données incohérentes à votre modèle de récompense et en leur attribuant des scores très bas, vous ne gagnerez jamais la partie. Vous pouvez faire cela pendant des dizaines de cycles : l'apprentissage par renforcement, lorsqu'il est poursuivi suffisamment longtemps, trouvera toujours un moyen de tromper le modèle. Il découvrira de nouveaux exemples adverses, et produira des scores très élevés pour des résultats dénués de sens.

Fondamentalement, c'est parce que notre fonction de score est un immense réseau neuronal, et que le RL est extrêmement efficace pour découvrir les failles qui lui permettent de le tromper.

---

96. Et nos modèles ne font pas encore de second degré.

Pour faire court, on exécute toujours le RLHF pendant quelques centaines de mises à jour: le modèle s'améliore, puis il faut l'arrêter. On ne peut pas aller beaucoup plus loin avec ce modèle de récompense, car l'optimisation commence alors à se détériorer. Alors on le stoppe, on le valide et on commence à l'utiliser.

Il est possible d'améliorer le modèle de récompense, mais on finit toujours par rencontrer ces situations à un moment donné. On peut dire, en gros, que le RLHF n'est pas vraiment du RL. Ce que je veux dire par là, c'est que le RLHF est évidemment une forme de RL, mais pas dans le sens « magique » du terme. Ce n'est pas un apprentissage par renforcement que l'on peut faire tourner indéfiniment.

Dans les domaines vérifiables, où l'on obtient une réponse correcte et concrète (voir figure 87, page 141), il est beaucoup plus difficile de tricher<sup>97</sup>. On obtient soit la bonne réponse, soit non, et la fonction d'évaluation y est beaucoup plus simple: il suffit de vérifier la zone encadrée et de voir si le résultat est correct. Il est donc très difficile de tromper ces fonctions. Mais dans le cas d'un modèle de récompense, il est possible de le manipuler.

Ce qu'il faut retenir: dans les domaines vérifiables, on peut pratiquer du RL indéfiniment: des dizaines de milliers, voire des centaines de milliers d'itérations, et découvrir ainsi toutes sortes de stratégies étonnantes auxquelles on n'aurait jamais pensé, permettant d'obtenir d'excellents résultats pour ces problèmes. Le RLHF appliqué aux domaines non vérifiables n'est pas aussi magique.

---

97. Dans ce contexte, le speaker emploie le verbe « to game » dans son sens informel: exploiter ou tromper un système afin d'obtenir de bons résultats sans réellement résoudre le problème sous-jacent.

- Dans les tâches où la réponse peut être vérifiée objectivement (comme les mathématiques ou le code), le modèle obtient la bonne réponse ou non; il n'existe pas de moyen simple de « tricher ».
- Mais dans les tâches non vérifiables (comme l'écriture créative ou la fabrication de blagues), les modèles peuvent "contourner" le système: ils produisent des résultats qui semblent bons selon un modèle de récompense défaillant ou subjectif, sans véritable compréhension ni raisonnement correct.

Au jeu de Go, il n'y a pas moyen de tricher pour gagner ou perdre une partie. Nous avons simulateur parfait: toutes les pierres sont connues, et nous pouvons calculer qui a gagné ou perdu. Il n'y a aucun moyen de contourner le système. Dans ce contexte, on peut exécuter un auto apprentissage par renforcement aussi longtemps qu'on veut, et finir par battre les meilleurs joueurs humains.

En revanche, dans des domaines non vérifiables ce processus ne peut pas être répété indéfiniment. C'est pourquoi je considère le RLHF comme n'étant pas un véritable RL. La fonction de récompense est manipulable.

C'est plutôt une petite amélioration marginale, mais pas un cadre fondamentalement solide où il suffirait de faire davantage de calcul pour faire encore mieux. Ce n'est pas de l'apprentissage par renforcement au sens fort du terme, car il lui manque cette « magie ». Il permet d'affiner le modèle et d'améliorer ses performances, mais pas d'atteindre à lui seul des progrès révolutionnaires.

Si l'on revient à ChatGPT

Emily buys 3 apples and 2 oranges. Each orange costs \$2. The total cost of all the fruit is \$13. What is the cost of each apple?

Reasoned for a couple of seconds ^

**Identifying costs**  
I started by setting the apple cost as  $x$ . Solving  $3x + 4 = 13$  gave  $x = 3$ .

**Breaking down the costs**  
I verified the formula by calculating the total fruit cost. With oranges at \$4, apples were priced at \$9, equating to \$3 each.

Let the cost of each apple be  $x$  dollars. Then the total cost is given by the equation:

$$3x + 2 \times 2 = 13$$

Since each orange costs 2 dollars, the cost for oranges is:

$$2 \times 2 = 4 \text{ dollars}$$

Substitute into the equation:

$$3x + 4 = 13$$

Subtract 4 from both sides:

$$3x = 9$$

Divide by 3:

$$x = 3$$

Thus, each apple costs 3 dollars.

Figure 114: Solution de GPT-4o au problème d'Émilie.

le modèle GPT-4o a bien été entraîné à l'aide du RLHF, car cette approche fonctionne efficacement. Mais ce n'est tout simplement pas du RL au sens strict du terme. Le RLHF s'apparente plutôt à une légère phase d'ajustement final, un réglage fin supplémentaire avec un simulateur d'annotateur qui améliore encore un peu le modèle. C'est ainsi que je le conçois.

03:06:57

Nous avons couvert les aspects techniques de la construction d'un LLM que je voulais aborder. On a vu les trois grandes étapes de l'entraînement:

1. le pré-entraînement
2. l'ajustement fin supervisé (supervised fine-tuning)
3. l'apprentissage par renforcement

J'ai montré qu'ils sont analogues à notre façon d'enseigner aux enfants (voir le manuel fig. 81, p. 131).

1. Le pré-entraînement correspond à l'acquisition des connaissances de base – à la première lecture si l'on préfère.
2. L'ajustement supervisé consiste à étudier de nombreux problèmes complètement résolus et à imiter les experts.
3. L'apprentissage par renforcement correspond aux exercices pratiques, où on n'a que l'énoncé et le résultat qu'il faut trouver.

La seule différence avec un LLM, c'est qu'il faut rédiger nous-mêmes les "manuels" destinés aux LLM et à l'IA dans l'ensemble des disciplines du savoir humain — y compris dans les domaines où nous voulons réellement qu'ils excellent, comme le code, les mathématiques, mais aussi dans toutes les autres disciplines.

Nous sommes donc en train d'écrire leurs manuels : en perfectionnant les algorithmes que j'ai décrits (en prenant une vue en hauteur) et, bien sûr, en mettant en œuvre de manière rigoureuse et efficace l'entraînement de ces modèles à grande échelle.

Je ne suis pas entré dans tous les détails, mais le pré-entraînement consiste en des tâches distribuées extrêmement vastes et complexes, qui doivent s'exécuter sur des dizaines de milliers, voire des centaines de milliers de GPU.

L'ingénierie nécessaire pour cela représente le sommet de ce qu'il est possible de faire avec les ordinateurs à cette échelle. Je n'ai pas traité cet aspect en profondeur, mais c'est un point très important, qui est à la base de tous les algorithmes simples que nous avons observés.

J'ai également à peine évoqué la théorie de l'esprit de ces modèles. Ce que je veux que vous reteniez, c'est que ces LLM sont très performants, mais qu'ils sont surtout extrêmement utiles comme *outils de travail*. Vous ne devez pas leur faire une confiance absolue. Je vous ai montré quelques exemples où ils peuvent dérailler. Même si nous avons des techniques pour limiter les hallucinations, les modèles ne sont pas parfaits et continueront d'en produire. Ils se sont améliorés ces dernières années et continueront à le faire, mais ils peuvent encore halluciner.

J'ai utilisé le « modèle du gruyère » des capacités des LLM – une image que vous devriez garder à l'esprit. Elle signifie qu'ils sont très bons, mais ont quand même des défauts.

Swiss cheese model of LLM capabilities of current models:

- some things work really well,
- some things (almost at random) show brittleness.



Figure 115: Un LLM est comme un morceau de gruyère.

Les modèles sont incroyablement performants dans de nombreux domaines différents, mais échouent parfois presque de façon aléatoire. Par exemple : entre 9,11 et 9,9, lequel est le plus grand ? (voir p. 127) — le modèle ne le sait pas. Pourtant, les LLM commencent à résoudre des questions d'Olympiades de mathématiques<sup>98</sup>. C'est ce que j'appelle un trou dans le fromage de gruyère. Il y en a beaucoup. Vous ne voulez pas tomber dedans.

Il ne faut donc pas traiter ces modèles comme infaillibles. Vérifiez toujours leurs résultats. Servez-vous-en comme d'outils. Utilisez-les pour vous inspirer. Employez-les pour une première ébauche. Mais travaillez avec eux comme avec des instruments, pas comme avec des autorités.

Soyez en dernière analyse responsable de votre travail.

C'est de cela que je voulais parler. Voilà comment ces modèles sont entraînés. Voilà ce qu'ils sont.

03:09:41

## 22. Aperçu de l'avenir

Voyons maintenant quelles pourraient être certaines des capacités futures de ces modèles — probablement ce qui se profile à l'horizon. Et aussi, où trouver ces modèles ?

---

98. Récemment un LLM développé par OpenAI a résolu des problèmes des Olympiades Internationales de Mathématiques, par exemple: on a trois nombres entiers positifs  $a$ ,  $b$ ,  $c$  tels que  $a + b + c = 2025$ . Montrer que

$$\frac{a}{b+c} + \frac{b}{c+a} + \frac{c}{a+b} \geq \frac{3}{2}$$

Le modèle a produit une démonstration valide en langage naturel, conforme aux instructions officielles: raisonnement de type « stylo et papier », sans accès à Internet, avec explications complètes.

C'est un niveau de capacité que l'on pensait auparavant hors de portée des réseaux neuronaux.

Noter que ce problème n'est pas bien sorcier. Pour commencer: la contrainte  $a + b + c = 2025$  ne joue aucun rôle (!) puisqu'on peut multiplier chaque nombre par trois milliards, ça ne change rien à l'inégalité.

J'ai préparé quelques "bullet points" (= points clé) sur cette image, listant ce à quoi on peut s'attendre:

PREVIEW OF THINGS TO COME

- multimodal (not just text but audio, images, video, natural conversations)
- tasks -> agents (long, coherent, error-correcting contexts)
- pervasive, invisible
- computer-using
- test-time training?, etc.

Figure 116: Aperçu de l'avenir.

Les modèles vont rapidement devenir multimodaux. Tout ce dont j'ai parlé jusqu'à présent concernait le texte, mais très bientôt, nous aurons des LLM capables non seulement de gérer le texte, mais aussi de traiter naturellement et aisément l'audio — ils pourront donc entendre et parler —, ainsi que les images — ils pourront voir et peindre.

Nous en voyons déjà les prémices, mais ces capacités seront bientôt entièrement intégrées dans le modèle de langage lui-même. Cela permettra des conversations beaucoup plus naturelles.

Du point de vue technique, ce si différent de tout ce que nous avons vu jusqu'ici: on peut facilement tokeniser de l'audio et des images, puis appliquer exactement les mêmes procédures que précédemment. Ce n'est donc pas un changement fondamental; il suffit simplement d'ajouter de nouveaux jetons.

Pour tokeniser de l'audio, par exemple, on peut découper le spectrogramme du signal sonore en petits intervalles, les transformer en jetons, et les ajouter directement aux fenêtres de contexte du modèle pour l'entraîner de la même manière que pour le texte. De même, pour les images, on peut utiliser des « patches » individuels. Une image n'est finalement qu'une séquence de jetons — et cela fonctionne.



On créera ainsi des flux de jetons "tokenisant" à la fois du texte, de l'audio et des images. Ils seront traités par un seul et même LLM.

Deuxièmement, pour l'instant le travail consiste à confier aux modèles des tâches individuelles sur un plateau d'argent, du type « résous cette tâche pour moi ». Et le modèle accomplit la petite tâche. Mais c'est à nous qu'il revient d'organiser une exécution cohérente de plusieurs tâches afin de réaliser un travail complet.

Les modèles n'ont pas encore la capacité d'exécuter des séries d'actions de façon cohérente et avec correction d'erreurs sur de longues périodes. Mais ils s'en approchent, et ces capacités s'améliorent progressivement au fil du temps.

Ce qui va probablement se produire, c'est que nous allons commencer à voir apparaître ce qu'on appelle des « agents », qui exécuteront des séries de tâches – où ils prendront des décisions durant le déroulement de la série de tâches – dans la durée. Vous les superviserez, vous observerez leur travail, et ils viendront vous faire des rapports d'avancement de temps en temps<sup>99</sup>.

Mais, comme nous l'avons dit plus haut, ces modèles ne sont pas infallibles. Ils nécessiteront une supervision.

Par exemple, dans les usines, on parle souvent du ratio humains-robots pour mesurer le niveau d'automatisation. Je pense que nous verrons apparaître quelque chose de similaire dans le domaine numérique, avec ce qu'on pourrait appeler des ratios humains-agents ; les humains deviendront de plus en plus des superviseurs de tâches exécutées par des agents, dans l'espace numérique.

---

99. Les programmes informatiques du premier domaine de l'informatique font cela depuis des décennies (voir la note 63, page 113).

Cependant, ils sont entièrement programmés et contrôlés par des humains. Les LLM, en revanche, disposent d'une bien plus grande liberté d'action: ils peuvent explorer, découvrir et produire des résultats inattendus.

Le point suivant est que tout cela va devenir bien plus omniprésent et invisible. Ce sera partout intégré aux outils, avec des capacités de calculs (= de l'IA locale).

Pour l'instant, ces modèles ne peuvent pas encore agir à votre place.

Lorsque ChatGPT a lancé la fonction « operator », en 2025, c'était un premier exemple en ce sens, où l'on pouvait confier temporairement le contrôle du clavier et de la souris au modèle pour exécuter des actions à votre place. C'est aussi quelque chose que je trouve très intéressant.

Le dernier point que j'aimerais évoquer (figure 116) est un commentaire général: il reste encore beaucoup de recherche à mener dans ce domaine. Un exemple est ce qu'on appelle l'« entraînement au moment du test ». Rappelez-vous que tout ce que nous avons vu jusqu'ici comprend deux grandes étapes principales.

- Il y a d'abord la phase d'entraînement, durant laquelle on ajuste les paramètres du modèle afin qu'il accomplisse correctement les tâches.
- Une fois ces paramètres obtenus, on les fige, puis on déploie le modèle pour l'inférence.

À partir de là, le modèle est fixe: il ne change plus. Il n'apprend pas à partir de ce qu'il fait lors de la phase de test<sup>100</sup>. Il dispose d'un ensemble de paramètres immuables; la seule chose qui varie désormais, ce sont les jetons à l'intérieur des fenêtres de contexte.

---

100. C'est une différence majeure avec les êtres humains. Une fois qu'ils ont acquis un savoir-faire et commencent à l'utiliser, les humains continuent à développer de nouvelles connaissances et compétences par la pratique. Il n'en va pas de même pour les LLM. Il peut être très frustrant pour les utilisateurs de montrer quelque chose à un modèle, pour qu'il l'ignore ensuite dès que la session est terminée. Une fois la conversation close, tout ce qui a été démontré au modèle est effacé de sa mémoire.

Les LLM pourraient progresser énormément s'ils pouvaient apprendre à partir de ces interactions de travail avec les humains. Mais cet objectif reste difficile: il nécessiterait de modifier les paramètres internes du modèle et de filtrer les informations malveillantes ou intentionnellement biaisées.

Le seul type de nouvel apprentissage auquel le modèle a accès *au moment de son utilisation*, est l'apprentissage en contexte – avec ce qu'il vient de faire durant l'interaction avec l'utilisateur, tant qu'elle n'est pas close.

Cela reste différent des humains, qui sont capables d'*apprendre réellement* en fonction de ce qu'ils font — notamment, par exemple, pendant le sommeil, lorsque le cerveau met à jour ses paramètres, ou quelque chose d'approchant. Il n'existe pour l'instant aucun équivalent de cela dans les modèles et les outils actuels.

Il reste encore beaucoup d'idées un peu expérimentales, voire barjotes, à explorer dans ce domaine. Ce sera nécessaire, car la fenêtre de contexte est une ressource finie et précieuse ; surtout lorsque nous commencerons à traiter des tâches multimodales de très longue durée. Lorsqu'on y intégrera des vidéos, les fenêtres de jetons deviendront énormes — non pas de quelques milliers ou centaines de milliers, mais bien davantage.

La seule astuce dont nous disposons aujourd'hui consiste à augmenter la longueur de ces fenêtres de contexte. Mais je pense que cette approche, à elle seule, n'est pas "scalable" à des tâches réellement longues et multimodales. De nouvelles idées sont nécessaires dans ces disciplines, pour traiter ces cas où les tâches exigeront des contextes extrêmement longs.

Voilà quelques exemples de ce à quoi on peut s'attendre dans un avenir proche.

03:15:15

## 23. Suivre le développement des LLM

Comment se tenir à jour ?

WHERE TO KEEP TRACK OF THEM

- reference <https://lmarena.ai/>
- subscribe to <https://buttondown.com/ainews>
- X / Twitter

Figure 117: Comment se tenir à jour ?

Comment rester à jour avec les dernières nouveautés et les avancées dans le domaine ?

Les trois ressources que j’ai utilisées régulièrement pour me tenir informé sont :

- 1. El Marena
- 2. Les abonnements
- 3. X/Twitter

Numéro un : El Marena. Voici de quoi il s’agit.

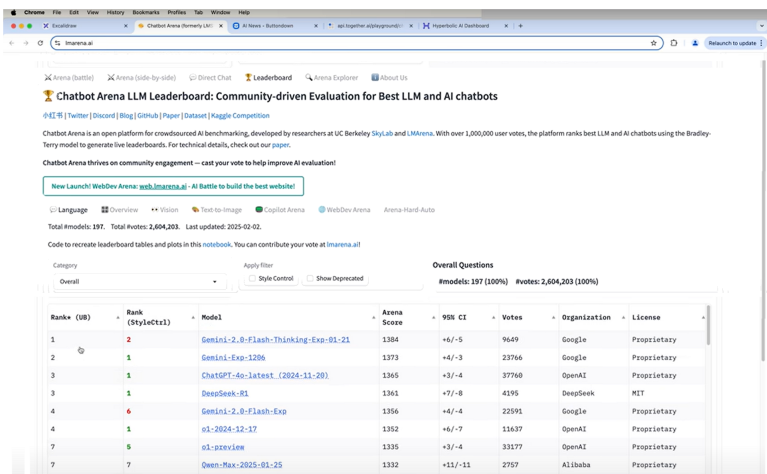


Figure 118: : <https://lmarena.ai/> en février 2025<sup>101</sup>.

Ceci est un tableau de classement des LLM. Il classe tous les meilleurs modèles. Le classement est fondé sur des comparaisons effectuées par des humains. Des personnes soumettent des requêtes à ces modèles, puis jugent lequel fournit la meilleure réponse. Elles ne savent pas quel modèle elles évaluent. Elles se concentrent simplement sur la qualité de la réponse. On peut ensuite calculer un classement.

101. Image plus grande [https://lapasserelle.com/documents/fig\\_118.png](https://lapasserelle.com/documents/fig_118.png)

Nous voyons figure 118 les différentes organisations — comme Google Gemini, par exemple — qui produisent ces modèles. Lorsque vous cliquez sur l'un d'entre eux, vous êtes redirigé vers l'endroit où ce modèle est hébergé, et la page principale s'ouvre.

On constate que Google est actuellement en tête, suivi de près par OpenAI. DeepSeek occupe la troisième position. Ce qui rend cela remarquable, c'est la dernière colonne de la figure 118: la mention « license ». *DeepSeek est un modèle sous licence MIT*<sup>102</sup>. Il est à poids ouverts: tout le monde peut utiliser ces poids, les télécharger ou héberger sa propre version de DeepSeek, et l'employer comme il le souhaite. Ce n'est pas un modèle propriétaire dont les sources seraient inaccessibles. C'est une version publique — un fait assez inédit pour un modèle aussi performant, donc une initiative remarquable de la part de l'équipe.

En dessous, on trouve encore plusieurs modèles de Google et d'OpenAI. En continuant à faire défiler, on voit apparaître d'autres « suspects habituels »: xAI, puis Anthropic avec Sonnet à la quatorzième place, puis Meta avec Llama, etc. Llama, comme DeepSeek, est un modèle à poids ouverts, mais il est classé plus bas dans la liste.

Ce tableau de classement a longtemps été très fiable. Cependant, au cours des derniers mois, il me semble qu'il a été un peu manipulé, et je ne lui accorde plus autant de confiance. Empiriquement, je constate par exemple que beaucoup de gens utilisent Sonnet d'Anthropic: c'est un excellent modèle, et pourtant il n'apparaît qu'à la position n°14.

---

102. DeepSeek est une entreprise chinoise d'intelligence artificielle basée à Hangzhou. Cependant, comme indiqué à la section 18, page 145, elle a publié son modèle. DeepSeek-R1 a été diffusé sous licence open source MIT. Depuis, de nombreuses entreprises ont téléchargé le modèle complet, qui peut être exécuté de manière autonome sans passer par un site web chinois. D'après le site LMArena, des institutions comme le MIT utilisent DeepSeek-R1 à des fins de recherche et d'évaluation comparative, mais elles le font dans le cadre de la licence open source MIT du modèle, et non par le biais d'un accord spécifique.

Inversement, je pense que beaucoup moins de personnes utilisent Gemini que ne le laisserait supposer son classement en tête de liste. Servez-vous donc du classement El Marena comme d'une première ébauche de classement, mais testez plusieurs modèles pour vos propres tâches et voyez lequel donne les meilleurs résultats<sup>103</sup>.

Le second point que je souhaiterais mentionner concerne les bulletins d'information sur l'IA.

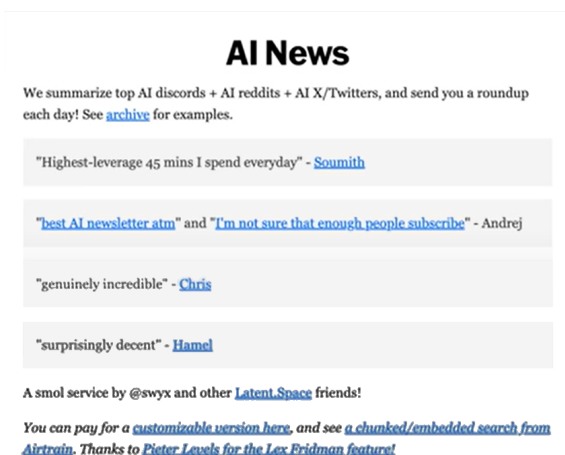


Figure 119: : AI newsletter <https://buttondown.com/ainews>.

AI News n'a pas un nom très original, mais c'est un excellent bulletin d'information publié par swyx101<sup>104</sup> et ses collaborateurs. Je les remercie de le maintenir: il m'a été très utile, car il est d'une exhaustivité remarquable.

103. Par exemple, j'ai résilié mon abonnement à Gemini quand, à mes demandes de me résumer la fin de romans célèbres qui étaient trop longs à lire jusqu'au bout, il m'a plusieurs fois fait des résumés de fin inventée! (Du même genre que dire que Mme Bovary, après sa crise existentielle, est retournée vivre heureuse avec Charles.)

104. Shawn Wang, plus connu sous le pseudonyme « swyx », est un ingénieur logiciel, auteur et formateur en intelligence artificielle, originaire de Singapour. Anciennement chez AWS et Temporal, il est aujourd'hui reconnu pour ses projets Latent Space, AI Engineer, ainsi que pour ses bulletins largement lus: AI News et DevTools.

Si vous allez dans les archives, vous verrez que ce contenu est produit presque tous les deux jours. Une partie est rédigée et organisée par des humains, mais une grande partie est générée automatiquement à l'aide de LLM. Si vous le parcourez régulièrement, vous ne manquerez probablement rien d'important de ce qui se passe en IA. Il est peu probable que vous le lisiez de manière exhaustive, car il est très long, mais les résumés tout en haut sont bien faits et, à mon avis, réalisés sous supervision humaine.

La dernière source que je mentionnerais est simplement X (anciennement Twitter). Beaucoup d'activités liées à l'IA s'y déroulent: vous pouvez donc y suivre des personnes que vous appréciez et en qui vous avez confiance, et y trouver les informations les plus récentes et les plus pertinentes. Ce sont là les principales sources qui se sont révélées efficaces pour moi au fil du temps.

03:18:34

## 24. Où trouver des LLM

Pour conclure, quelques mots sur les endroits où vous pouvez trouver les modèles et sur la manière dont vous pouvez les utiliser.

### WHERE TO FIND THEM

- Proprietary models: on the respective websites of the LLM providers
- Open weights models (DeepSeek, Llama): an inference provider, e.g. TogetherAI
- Run them locally! LMStudio

Figure 120: Où trouver des LLM.

Le premier point, je dirais, c'est que pour accéder à l'un des plus grands modèles propriétaires, il suffit de se rendre sur le site web du fournisseur de LLM correspondant.

Par exemple, pour OpenAI : <https://chatgpt.com/> — ou simplement <https://chat.com/>, qui fonctionne apparemment désormais.

Pour Gemini, c'est <https://gemini.google.com> ou <https://aistudio.google.com/>. Ils en ont deux pour une raison que je ne comprends pas. Personne ne le comprend du reste.

Pour les modèles à poids ouverts comme DeepSeek, Llama et d'autres, il faut généralement passer par un fournisseur d'inférence – c'est-à-dire un service d'hébergement qui exécute les modèles et les rend accessibles via une API. Mon favori est <https://together.ai>:

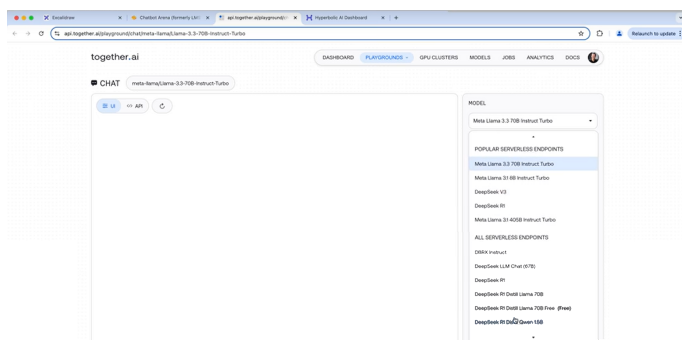


Figure 121: Bac à sable de <https://together.ai>.

J'ai montré que lorsque vous allez sur le « playground » de <https://together.ai>, vous pouvez choisir parmi de nombreux modèles différents (grâce au menu déroulant situé à droite de l'écran). Tous ces modèles sont ouverts, et vous pouvez les essayer dans la fenêtre de discussion.

Si vous souhaitez utiliser un *modèle de base* (par opposition à un *modèle d'assistant*), il est un peu plus difficile d'en trouver chez les grands fournisseurs d'inférence. La plupart des plateformes d'hébergement se concentrent sur les modèles de chat ou d'assistance, et non sur les versions brutes.

C'est pourquoi, lorsque j'ai besoin d'un modèle de base, je vais généralement sur <https://www.hyperbolic.ai/>, car ils hébergent le modèle Llama3.1base, que j'apprécie. Vous pouvez interagir directement avec lui sur leur site. À ma connaissance, Hyperbolic est l'un des meilleurs endroits pour accéder



à un modèle de base. Ce serait bien que davantage de fournisseurs en proposent, car ils sont très utiles dans certains cas.

Enfin, vous pouvez aussi utiliser certains modèles plus petits et les exécuter localement. Par exemple, avec DeepSeek, vous ne pourrez pas exécuter le plus grand modèle sur votre MacBook, mais il existe des versions plus petites du modèle, connues sous le nom de versions « distilled ». Vous pouvez également exécuter ces modèles à plus faible précision: non pas avec la précision native utilisée pour les modèles complets (comme FP8 pour DeepSeek ou BF16 pour Llama), mais avec des niveaux de précision plus faibles. Ne vous inquiétez pas si ces détails vous échappent: il suffit de savoir que vous pouvez exécuter des versions réduites et allégées de ces modèles, ce qui leur permet de tenir sur votre ordinateur. Ainsi, vous pouvez faire tourner des modèles déjà très performants sur votre propre portable.

Mon endroit préféré reste LM Studio (<https://lmstudio.ai/>), qui est une application téléchargeable.

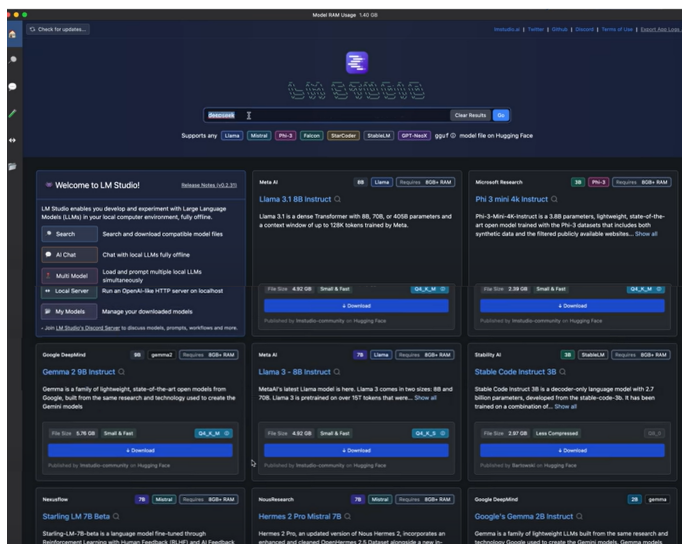


Figure 122: LM Studio.

Je trouve l'interface laide, et je n'aime pas le fait qu'elle affiche de nombreux modèles qui ne sont pas vraiment utiles – puisque, soyons honnêtes, tout le monde veut simplement exécuter DeepSeek. Je ne comprends pas pourquoi ils proposent ces trente-six variantes de modèles: la recherche est vraiment compliquée, et il faut choisir entre différentes versions « distil-  
led » et plusieurs niveaux de précision, ce qui rend l'ensemble très confus.

Mais une fois que vous avez compris comment cela fonctionne (ce qui nécessiterait en soi toute une vidéo), vous pouvez charger un modèle – par exemple, ici j'ai chargé Llama 3 Instruct 1 Billion – et simplement discuter avec lui. Je lui ai demandé une blague sur les pélicans:

UTILISATEUR: Raconte-moi une blague sur un pélican.

IA: En voici une: Pourquoi le pélican est-il allé chez le médecin ? Parce qu'il avait une « fowl cough » ! (tu saisis ?)

J'en ai demandé une autre:

UTILISATEUR: Une autre.

IA: Pourquoi le pélican est-il allé en thérapie ? Parce qu'il avait « le bec plein de problèmes ».

Et elle m'en a donné une autre, et ainsi de suite.

Pour faire cela, il faut d'abord télécharger et installer LM Studio, ainsi que le modèle spécifique que vous souhaitez utiliser. Ensuite, tout le traitement se fait localement sur votre ordinateur: rien n'est envoyé ailleurs, tout fonctionne directement sur le GPU de votre MacBook Pro. C'est très pratique ; et quand vous avez terminé, vous pouvez éjecter le modèle pour libérer la mémoire RAM.

LM Studio est donc probablement mon outil préféré, même si son interface présente pas mal de défauts d'ergonomie et qu'il semble davantage conçu pour un public professionnel. Mais si vous regardez quelques tutoriels sur YouTube, vous comprendrez assez vite comment l'utiliser.

Ce ne sont là que quelques remarques sur les endroits où trouver et utiliser ces modèles.

03:21:46

## 25. Résumé et conclusion

Revenons à notre point de départ. La question était: quand nous allons sur ChatGPT, que nous saisissons une requête et que nous cliquons sur « Go », que se passe-t-il exactement ? À quoi avons-nous affaire et comment tout cela fonctionne-t-il ?

J'espère que ces explications vous auront permis de mieux comprendre ce qui se passe sous le capot: comment ces modèles sont entraînés et qu'est-ce qu'ils nous renvoient réellement.

Lorsque vous tapez votre requête, elle est d'abord découpée en tokens. Cette étape de « tokenisation » transforme l'entrée en petites unités – des sous-mots – que le modèle peut traiter. Votre prompt est ensuite intégré dans un format de conversation particulier qui structure le dialogue entre l'utilisateur et l'assistant. Le modèle reçoit ce prompt sous forme d'une séquence unidimensionnelle de tokens. Lorsque vous appuyez sur « Go », le modèle commence à générer la sortie en poursuivant cette séquence, un token après l'autre, un peu comme un système d'autocomplétion intelligent.

Pourquoi le modèle choisit-il ces tokens particuliers, et d'où viennent-ils ? Cela tient à la manière dont ces systèmes sont construits, en trois grandes étapes. La première, le pré-entraînement, consiste à acquérir des connaissances: absorber une immense quantité d'informations issues d'une grande partie d'Internet pour les encoder dans les paramètres du modèle. La deuxième étape, le fine-tuning supervisé, est celle où émergent la personnalité et le comportement du LLM. Des entreprises comme OpenAI créent d'énormes ensembles de dialogues entre un humain et un assistant (entièrement produits par des humains: les deux côtés de la conversation). Des annotateurs humains conçoivent les réponses « idéales » de l'assistant selon des consignes précises, et le modèle apprend ainsi à répondre de manière appropriée à n'importe quelle requête.

En réalité, lorsque vous recevez une réponse, c'est une imitation d'un de ces annotateurs humains d'OpenAI, entraîné à rédiger la meilleure réponse possible. Le modèle ne « pense » pas comme un humain: il effectue un calcul mathématique contraint pour prédire des tokens. Chaque génération de token est une procédure finie: une simulation approximative du raisonnement, pas une cognition véritable.

De ce fait, les modèles font parfois des erreurs ou « hallucinent ». Leurs capacités ressemblent à du gruyère: impressionnantes, mais pleines de trous qui causent des erreurs inattendues.

Malgré ces limites, les modèles récents dits « pensants », comme o1-mini ou o3-mini d'OpenAI, vont plus loin. Contrairement à GPT4-Turbo, qui repose surtout sur le fine-tuning supervisé (SFT), ces nouveaux modèles passent aussi par l'étape d'« apprentissage par renforcement avec feedback humain » (RLHF).

Cette troisième étape leur permet de développer des stratégies de raisonnement: des schémas proches d'un monologue intérieur. Ils s'exercent sur de nombreux problèmes élaborés par des chercheurs humains, ce qui leur fait acquérir des techniques inédites.

Ainsi, quand vous les interrogez, vous ne dialoguez plus seulement avec une simulation d'annotateur, mais vous observez les prémices d'un véritable raisonnement – une capacité émergente issue de ce processus de renforcement.

Cela dit, l'apprentissage par renforcement reste encore primitif: nous n'en sommes qu'au début. Ces modèles ont parfois des éclairs d'intuition – une créativité étonnante ou un raisonnement inédit – mais c'est encore en phase expérimentale. Leur force est dans les domaines vérifiables comme les mathématiques ou le code. Leur aptitude à transférer ces raisonnements vers des tâches ouvertes, comme l'écriture créative, reste en revanche fragile.

Personnellement, j'utilise ces modèles tous les jours, parfois des dizaines de fois, car ils accélèrent considérablement mon travail.

Ils doivent être employés avec discernement – comme des outils, non comme des autorités. Il faut toujours vérifier leurs résultats: même les meilleurs peuvent produire des erreurs aléatoires, mal compter ou inventer des faits. Utilisés avec méthode – pour rédiger, explorer des idées ou affiner vos travaux – ces systèmes peuvent rendre votre travail infiniment plus productif. Ils peuvent vous faire faire des progrès extraordinaires <sup>105</sup>.

Nous vivons une période très stimulante de l’IA. C’est une révolution informatique aussi importante que l’invention du web, en 1989, ou des moteurs de recherche <sup>106</sup>.

J’espère que cette vidéo vous a été utile et vous a plu.

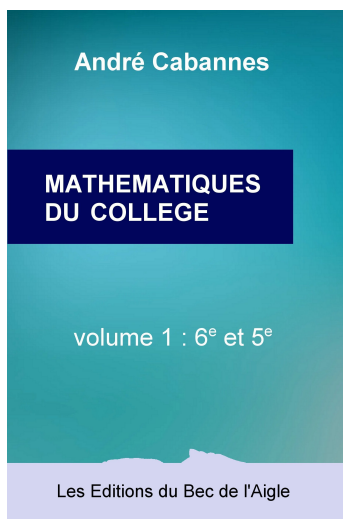
---

105. Comme un séjour de quelques années à l’étranger peut enrichir considérablement votre vision du monde et vous faire grandir.

106. Pour prendre la mesure de l’évolution de l’informatique grand public dans les années 80 et 90 voir [https://lapasserelle.com/formation\\_veille.html](https://lapasserelle.com/formation_veille.html)



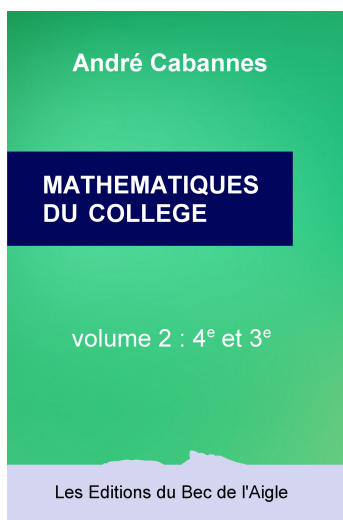
# Catalogue des ÉDITIONS DU BEC DE L'AIGLE



[www.amazon.fr/dp/2957239159](http://www.amazon.fr/dp/2957239159)  
Cours de mathématiques du col-  
lège.

Volume 1 : 6e et 5e.

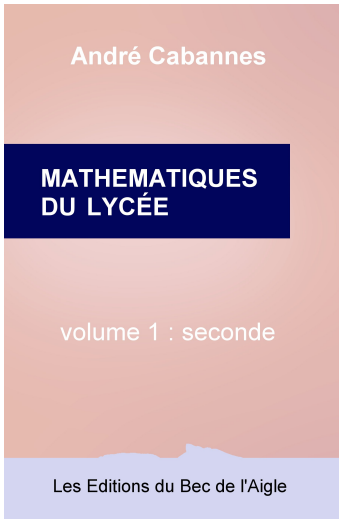
à l'intention des collégiens et de  
leurs parents



[www.amazon.fr/dp/2957239167](http://www.amazon.fr/dp/2957239167)  
Cours de mathématiques du col-  
lège.

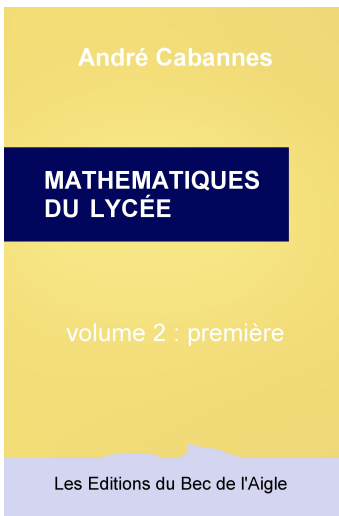
Volume 2 : 4e et 3e.

à l'intention des collégiens et de  
leurs parents



[www.amazon.fr/dp/2957239183](http://www.amazon.fr/dp/2957239183)  
Cours de mathématiques de se-  
conde

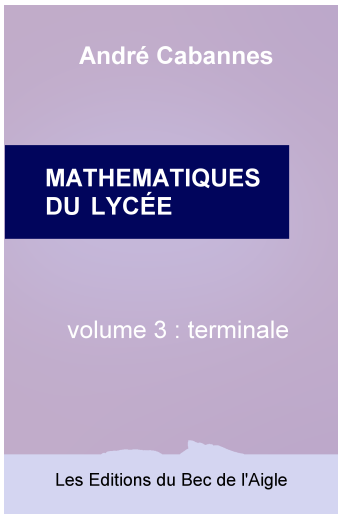
à l'intention des lycéens et de leurs  
parents



[www.amazon.fr/dp/2957239191](http://www.amazon.fr/dp/2957239191)  
Cours de mathématiques de pre-  
mière

à l'intention des lycéens et de leurs  
parents



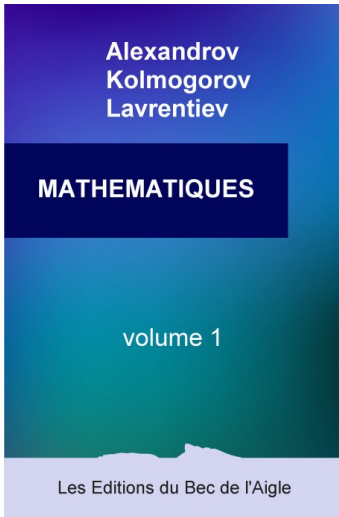


[www.amazon.fr/dp/2958738507](http://www.amazon.fr/dp/2958738507)  
Cours de mathématiques de terminale

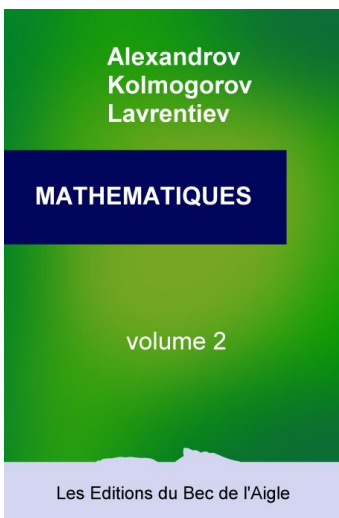
à l'intention des lycéens et de leurs parents



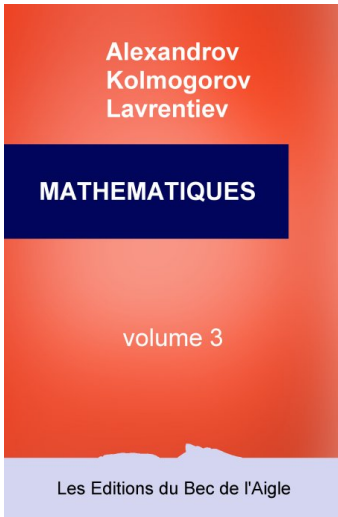
[www.amazon.fr/dp/2957239140](http://www.amazon.fr/dp/2957239140)  
Cours de comptabilité (niveau baccalauréat)



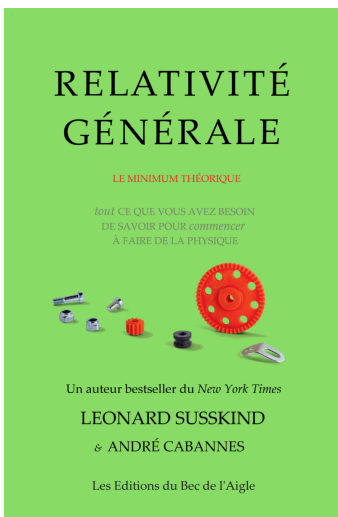
[www.amazon.fr/dp/2957239124](http://www.amazon.fr/dp/2957239124)  
Introduction aux mathématiques  
(niveau baccalauréat)



[www.amazon.fr/dp/2957239116](http://www.amazon.fr/dp/2957239116)  
Les mathématiques pour l'utilisateur  
(niveau première année d'université)



[www.amazon.fr/dp/2957239132](http://www.amazon.fr/dp/2957239132)  
Les mathématiques pour l'étudiant  
spécialisé et le chercheur (niveau li-  
cence)



[www.amazon.fr/dp/2957239175](http://www.amazon.fr/dp/2957239175)  
Cours de physique  
(niveau maîtrise)