MP-i

IA GÉNÉRATIVE

LIVRE BLANC TECHNIQUE

Comprendre l'IA générative : du bit au modèle

La mécanique d'un modèle de langage — poids, tokens, attention, fenêtre de contexte

Ce document explique ce qu'est réellement un modèle de langage et comment il produit du texte. Il s'adresse à toute personne qui doit décider, acheter ou encadrer un usage de l'IA générative sans être développeur.

Mattieu Pottier

Consultant en transformation digitale — SIG, géomatique, Web, ERP Odoo
MP-i — Mattieu Pottier Indépendant · mp-i.pro

Version 1.0.0 — Août 2026
Document distribué librement

SOMMAIRE

Préface

Ce document est né d'un constat. L'IA générative s'installe partout — dans les outils, dans les process, dans les budgets — et change en profondeur la façon de travailler. Mais ceux qui l'utilisent au quotidien ne savent pas vraiment ce que c'est, ni comment elle fonctionne.

J'ai écrit ce livre blanc en m'appuyant sur sept ouvrages techniques de référence, dépouillés intégralement. Chaque affirmation de ce document provient de l'un d'eux, ou d'une source publiée que je cite. Quand une source est prudente, je le suis aussi. Quand aucune source ne dit ce que j'aimerais écrire, je ne l'écris pas.

Le pari de ce livre blanc est le suivant : les noms de produits changent tous les six mois, les architectures tous les deux ans, mais les contraintes physiques ne bougent pas. Ce qui est expliqué ici sera encore vrai quand les modèles d'aujourd'hui auront été remplacés trois fois. Comprendre cela une fois, c'est comprendre pour longtemps.

Mattieu Pottier — MP-i · Mattieu Pottier Indépendant

Introduction

Tout le monde a utilisé l'IA générative. Presque personne ne sait ce qu'est le fichier qui tourne derrière. Les décisions se prennent alors sur des intuitions fausses — « j'ai 32 Go de RAM, ça devrait passer », « le modèle connaît nos documents », « il apprend de nos conversations ». Trois phrases courantes, trois erreurs qui coûtent cher.

Ce document explique la mécanique. Pas les usages, pas le marché, pas la prospective : le fonctionnement.

Pourquoi ce sujet est stratégique

Une technologie qu'on ne comprend pas se paie deux fois. Une première fois à l'achat, quand on dimensionne mal parce qu'on ignore ce qui consomme quoi.

Une seconde fois à l'usage, quand on découvre que ce qu'on croyait acquis ne l'est pas — que le modèle ne retient rien, qu'il ne connaît aucun document interne, qu'une réponse en français coûte plus cher qu'en anglais.

Ces malentendus ne sont pas des détails d'implémentation. Ils déterminent le budget, le périmètre fonctionnel et le niveau de risque d'un projet. Un décideur qui sait ce qu'est un token n'achètera pas la même chose qu'un décideur qui l'ignore.

Il y a une seconde raison, moins évidente. Le vocabulaire employé dans ce domaine est trompeur par construction : on dit qu'un modèle *apprend*, qu'il *comprend*, qu'il *raisonne*, qu'il *se souvient*. Chacun de ces verbes désigne une opération mécanique qui n'a rien à voir avec ce que le mot suggère. Tant qu'on raisonne avec ce vocabulaire, on se trompe sur ce que la machine peut faire.

Enfin, les contraintes exposées ici sont les seules qui ne se périmeront pas. Les modèles se remplacent, les fournisseurs changent de tarif, les interfaces se refont. La quantité de mémoire nécessaire pour tenir un million de nombres en 16 bits, elle, sera la même dans dix ans.

Le périmètre de ce document

Ce livre blanc couvre la mécanique d'un modèle de langage : ce que sont les poids et pourquoi leur nombre décide de la taille du fichier, comment un texte devient une suite d'identifiants numériques, ce que contient réellement une fenêtre de contexte (*context window*) et d'où vient son coût, ce qui distingue la fabrication d'un modèle de son utilisation.

Il ne couvre délibérément ni le choix d'un modèle, ni le dimensionnement d'une machine, ni l'arbitrage entre exécution locale et service en ligne. Ces questions dépendent de produits et de tarifs qui changent, et elles sont traitées séparément sur le blog MP-i. Il ne contient non plus aucune mesure effectuée sur une machine : tout ce qui est chiffré ici provient d'une source publiée et citée.

Aucun nom de modèle, d'éditeur ou de produit n'apparaît dans le corps du document. C'est un choix : ces noms seraient obsolètes avant que vous ne finissiez de le lire.

Profils de lecture

COMMENT LIRE CE DOCUMENT SELON VOTRE PROFIL

PROFIL	PARTIES PRIORITAIRES	OBJECTIF
Décideur, direction, DSI	Introduction, Parties 1 et 3, Lire une fiche de modèle	Savoir ce qu'on achète et ce qui détermine le coût
Chef de projet, consultant	Parties 1, 3, 4 et Lire une fiche de modèle	Cadrer un projet sans se faire vendre l'impossible
Profil technique non spécialiste	Intégralité, en particulier les Parties 4 et 5	Comprendre la mécanique avant de dimensionner

Partie 1 — Les briques : ce qu'est réellement un modèle

Commençons par les définitions officielles, puis démontons-les morceau par morceau. À la fin de cette partie, vous relirez la même phrase et vous la comprendrez entièrement.

1.1 Trois définitions officielles

Contrairement à ce qu'on pourrait croire, ces termes ne sont pas laissés au marketing. La Commission d'enrichissement de la langue française publie leurs définitions au **Journal officiel de la République française**, avec leur équivalent anglais. Voici les trois qui nous concernent.

> **intelligence artificielle, IA — artificial intelligence, AI** —
>
> « Champ interdisciplinaire théorique et pratique qui a pour objet la compréhension de mécanismes de la cognition et de la réflexion, et leur imitation par un dispositif matériel et logiciel, à des fins d'assistance ou de substitution à des activités humaines. »
> *Journal officiel du 9 décembre 2018*

Retenez le verbe : **imitation**. Il ne s'agit pas de reproduire la cognition, mais d'en imiter les mécanismes. La nuance est dans le texte officiel, et elle n'est pas décorative.

Notez aussi la date. L'intelligence artificielle est définie au *Journal officiel* depuis 2018, et le terme lui-même a été forgé par John McCarthy en **août 1955**, dans la proposition d'un atelier d'été qui se tiendra à Dartmouth l'année suivante. L'IA n'a pas commencé en 2022 : elle avait soixante-dix ans.

> **intelligence artificielle générative, IA générative — generative AI, GenAI** —
>
> « Branche de l'intelligence artificielle mettant en œuvre des modèles génératifs, qui vise à produire des contenus textuels, graphiques ou audiovisuels. »
> *Journal officiel du 6 septembre 2024*

Voilà la distinction que presque personne ne fait, réglée en une ligne : **l'IA générative est une branche de l'IA, pas son synonyme.** Un filtre antispam, un moteur de recommandation, un système de reconnaissance de plaques d'immatriculation relèvent tous de l'intelligence artificielle. Aucun n'est génératif. Ce qui distingue la branche générative, c'est qu'elle **produit** un contenu au lieu de classer, de prédire ou de détecter.

Reste le troisième terme, celui qui nous occupera jusqu'au bout de ce document.

> **grand modèle de langage, GML — large language model, LLM** —
>
> « Modèle génératif qui, à partir de grands volumes de données textuelles, calcule des probabilités des enchaînements de jetons textuels en vue de la génération automatique

de texte ou de code informatique. »
Journal officiel du 6 septembre 2024

Cette phrase est le programme de tout le livre blanc. Elle tient en trente mots et contient déjà tout ce qu'il faut comprendre : un **modèle**, des **données textuelles**, des **probabilités**, des **jetons textuels**. Quatre notions, dont aucune n'est évidente.

Le reste de cette partie démonte les deux premières — ce qu'est un modèle et de quoi il est fait. Les parties 4 et 5 traiteront des jetons et des probabilités.

> **Sur les termes anglais employés dans ce document**
>
> Le *Journal officiel* recommande les termes français, mais l'usage professionnel est massivement anglophone : personne ne dit « grand modèle de langage » à l'oral, tout le monde dit **LLM**.
> Ce document donne donc les deux au premier emploi, puis retient celui que vous entendrez réellement. Le glossaire final porte les équivalences complètes, avec leur date de publication au *Journal officiel*.

1.2 Bit, octet, byte : la seule arithmétique du document

Dans une machine, toute information se ramène en dernier ressort à l'état d'un circuit électrique : il laisse passer le courant, ou il ne le laisse pas. Deux états possibles, jamais trois. C'est cette unité élémentaire qu'on appelle un **bit**, contraction de l'anglais *binary digit* — chiffre binaire.

Un bit est une case qui vaut 0 ou 1. Huit bits font un octet. En anglais, l'octet se dit *byte* et se note **B** en majuscule ; le bit se note **b** en minuscule.

D'où un piège permanent sur les débits : 100 Gb(bit)/s et 100 Go(octet)/s ne sont pas la même chose, il y a un facteur 8 entre les deux.

1.3 Un neurone artificiel, c'est une multiplication et une addition

Le *Journal officiel* définit aussi cette brique, et sa définition est plus instructive qu'il n'y paraît.

neurone artificiel, neurone formel — artificial neuron —

« Dispositif à plusieurs entrées et une sortie, qui simule certaines propriétés du neurone biologique. La valeur de sortie du neurone artificiel est une fonction non linéaire, généralement à seuil, d'une combinaison de valeurs d'entrée dont les paramètres sont ajustables. »
Journal officiel du 9 décembre 2018

Traduisons. Un neurone artificiel prend des entrées, multiplie chacune par un nombre, additionne les résultats, et fait passer le total dans une petite fonction qui décide de la sortie. Le texte officiel dit d'ailleurs « simule **certaines** propriétés », pas « reproduit le fonctionnement ».

Andrew Trask, dans *Grokking Deep Learning*, le formule sans détour dès son chapitre 3 : un réseau de neurones, à ce stade, c'est un ou plusieurs poids que vous multipliez par les données d'entrée pour produire une prédiction. Empilez ces opérations en couches, faites-en des millions, vous obtenez un **réseau de neurones artificiels** (*artificial neural network*)

Quand ces couches se comptent par dizaines, on parle de **réseau profond** (*deep neural network*). Et la méthode qui consiste à en entraîner un porte elle aussi un nom officiel.

apprentissage profond — deep learning — « Apprentissage automatique qui utilise un réseau de neurones artificiels composé d'un grand nombre de couches dont chacune correspond à un niveau croissant de complexité dans le traitement et l'interprétation des données. »
Journal officiel du 9 décembre 2018

Attention au faux ami, il piège la plupart des lecteurs francophones. Le « profond » ne désigne **aucune compréhension plus profonde** : il compte les couches, rien d'autre. François Chollet le souligne explicitement — le mot renvoie à l'empilement de représentations successives, jamais à une quelconque profondeur de pensée.

Deux mots de la définition officielle portent tout le reste : « paramètres **ajustables** ». C'est là que se loge l'apprentissage.

La conséquence est celle-ci. **L'intelligence n'est pas dans l'opération — l'opération est triviale. Elle est dans la valeur des nombres.**

1.4 Les poids : tout ce que le modèle a appris tient dans des nombres

Dans le neurone décrit plus haut, chaque entrée est multipliée par un nombre. Un réseau en compte des milliards — un par connexion, sur chacune de ses couches. Ce sont eux, et eux seuls, qui font la différence entre un modèle qui écrit correctement et un modèle qui produit du charabia.

Ces nombres s'appellent des poids (*weights*), ou des paramètres. Sebastian Raschka, dans *Build a Large Language Model (From Scratch)*, les définit comme les variables internes du modèle, ajustées et optimisées pendant l'entraînement.

Ils ne sont pas écrits par un humain. La définition officielle de l'**apprentissage automatique** (*machine learning*, ML) le dit précisément : c'est un « processus par lequel un algorithme évalue et améliore ses performances sans l'intervention d'un programmeur », en modifiant les paramètres d'un modèle « de valeur initiale en général aléatoire, en fonction du résultat constaté ».

Autrement dit : on part de nombres tirés au hasard, et on les corrige des millions de fois jusqu'à ce que les sorties deviennent pertinentes. Personne n'a écrit ces valeurs, et personne ne sait lire ce qu'elles signifient individuellement.

Ce processus porte un nom : **l'entraînement** (*training*). Il a lieu une fois, chez celui qui fabrique le modèle, puis il s'arrête. Ce qui se passe ensuite, quand vous posez une question au modèle, est une opération entièrement différente — **l'inférence** (*inference*). La partie 3 chiffre l'écart entre les deux ; il se compte en ordres de grandeur.

Voici le point qui décide de la moitié des malentendus sur le sujet.

> ▲ **Ce qu'un modèle ne contient pas**
>
> **Un modèle n'a pas de base de données. Pas de fichiers. Pas de mémoire de ce qu'il a lu.** Il n'a que ces nombres, figés au moment où l'entraînement s'est arrêté.
> Raff, Farris et Biderman sont catégoriques dans *How Large Language Models Work* : un modèle de langage n'est entraîné à faire aucune des quatre choses suivantes — mémoriser du texte, produire des idées nouvelles, construire une représentation du monde, ou générer du texte factuellement exact.
> Il est entraîné à produire du texte qui **ressemble** à celui qu'il a vu. C'est aussi le mécanisme profond de ce qu'on appelle une **hallucination** — le mot est le même en anglais : un énoncé faux mais vraisemblable satisfait parfaitement l'objectif d'entraînement.
> Conséquence directe : **il ne « connaît » aucun de vos documents**, et aucun réglage ne changera cela.

Ce qu'on appelle « le modèle », c'est ce fichier

Ces poids, une fois l'entraînement terminé, sont enregistrés. Le résultat est un fichier — et c'est lui qu'on télécharge, qu'on charge en mémoire, qu'on appelle « le modèle ».

Sebastian Raschka en montre le contenu réel : un fichier de modèle, ouvert, se réduit à **deux choses**. D'un côté un dictionnaire de nombres, les poids proprement dits. De l'autre une poignée de réglages d'architecture — nombre de couches, nombre de têtes d'attention, dimension des vecteurs, taille du vocabulaire. Quelques lignes de configuration, face à des milliards de valeurs.

C'est pourquoi l'expression employée dans tout ce document est littérale, et non métaphorique.

> → **Fichier de nombres**
>
> Un modèle est un **fichier de nombres** : l'ensemble des poids appris, accompagné de la courte notice technique qui indique comment les organiser.
> Il n'y a rien d'autre dedans. Pas de code métier, pas d'index documentaire, pas de règles écrites par quelqu'un. C'est un tableau de valeurs, très grand, et une notice pour le lire.
> Cette phrase reviendra à chaque partie de ce document, parce que tout en découle : ce que le fichier pèse, ce qu'il coûte à faire tourner, et ce qu'il ne saura jamais faire.

1.5 Le jeton textuel : l'unité que le modèle voit réellement

Dernière brique, et la seule qui ne concerne pas le modèle mais ce qu'on lui donne à lire.

> **jeton textuel — text token, token** — « Donnée élémentaire d'un grand modèle de langage, qui est constituée d'une suite de caractères obtenue par la segmentation automatique d'un texte. La segmentation en jetons textuels est utilisée aussi bien dans la phase d'apprentissage automatique que dans la phase de traitement d'une instruction générative. »
> *(« instruction générative » est le terme officiel pour ce que tout le monde appelle un **prompt**.)*
> *Journal officiel du 6 septembre 2024*

« Donnée élémentaire » : c'est bien une brique. Un modèle de langage ne lit ni lettres ni mots — il reçoit une suite d'identifiants numériques, chacun renvoyant à un fragment de texte. Raff, Farris et Biderman insistent sur ce point parce qu'il commande tout le reste : **les jetons sont la seule information avec laquelle le modèle interagit.**

Un ordre de grandeur, pour que ce ne soit pas abstrait. Raschka découpe une même nouvelle de trois façons dans son chapitre 2, et publie les trois résultats : **20 479 caractères**, **4 690 mots**, **5 145 jetons**.

Retenez donc **4 caractères par jeton, et 1,1 jeton par mot**. Un mot courant tient dans un seul jeton ; les mots longs, rares ou mal orthographiés en réclament plusieurs.

Une précision qui aura son importance : ce texte est en **anglais**, et c'est le cas le plus favorable. La section 4.3 chiffre ce que coûtent les autres langues.

Retenez aussi que la segmentation intervient **aux deux bouts** — quand on fabrique le modèle, et quand on s'en sert. C'est la même opération, et elle se facture des deux côtés.

Un mot enfin sur une ambiguïté du texte officiel, parce qu'elle prête à confusion sur les fiches techniques. Le *Journal officiel* précise que « le nombre de jetons textuels sert à mesurer la taille d'un grand modèle de langage », alors que l'usage professionnel annonce la taille en **paramètres** — c'est le « 7B » qu'on voit partout.

> ⚠ **Deux « tailles » qu'il ne faut pas confondre**
>
> La **taille du modèle** se compte en paramètres : 7 milliards, 70 milliards. Elle détermine le poids du fichier et la mémoire nécessaire.
> La **taille du corpus d'entraînement** se compte en jetons : des centaines ou des milliers de milliards. Elle ne dit rien de l'encombrement du modèle obtenu.
> Les deux chiffres circulent sous le même mot. Un modèle « entraîné sur 2 000 milliards de tokens » n'est pas un modèle de 2 000 milliards de paramètres — la partie 3 chiffre les deux.

1.6 Synthèse

Reprenons la phrase du *Journal officiel* posée en ouverture de cette partie, et relisons-la maintenant que chaque mot a un sens.

> *Un grand modèle de langage est un « **modèle génératif** qui, à partir de **grands volumes de données textuelles**, calcule des **probabilités** des enchaînements de **jetons textuels** en vue de la génération automatique de texte ou de code informatique. »*

Deux de ces quatre expressions sont désormais claires — et ce sont celles qui commandent les autres.

Modèle génératif. Vous savez ce qu'est un modèle : un fichier de nombres. Des milliards de paramètres ajustables, partis de valeurs aléatoires et corrigés automatiquement, organisés en couches de neurones artificiels, enregistrés dans un fichier avec leur courte notice d'architecture. Le *Journal officiel* le confirme dans sa propre définition — un modèle génératif est le « résultat d'un apprentissage automatique ». **Un résultat, pas un programme.**

Jetons textuels. Vous savez ce qu'ils sont : les données élémentaires que le modèle reçoit, les seules avec lesquelles il interagit. Ni lettres, ni mots — des fragments de texte convertis en identifiants numériques.

Les deux expressions restantes appartiennent aux parties suivantes.

OÙ LE RESTE DE LA DÉFINITION SERA TRAITÉ

- **Grands volumes de données textuelles** — le corpus d'entraînement, à ne pas confondre avec le fichier de modèle. Partie 3, avec ce que coûte sa fabrication et ce que pèse le fichier obtenu
- **Probabilités** — comment le modèle choisit le fragment suivant, et pourquoi ce mécanisme coûte de la mémoire à chaque mot produit. Partie 5

Ce que vous savez déjà, ce qu'il reste à voir

Acquis — un modèle est un fichier de nombres ajustés automatiquement, qui ne contient aucun document ni aucune base de données. Il ne voit du texte que des jetons.

À voir — comment il a été fabriqué, à quel prix, et ce que pèse le fichier obtenu (partie 3) · comment le découpage en jetons fonctionne et ce qu'il coûte selon la langue (partie 4) · ce que coûte le fait de lui donner à lire (partie 5).

Partie 2 — Une brève histoire, pour situer ce qui est nouveau

Presque rien de ce qui compose l'IA générative n'est neuf. Une seule chose l'est vraiment, et il vaut la peine de savoir laquelle — parce que cela dit où se situe réellement le progrès, et donc où il continuera.

1943-1969	1969-1986	1986-2012	2012-2017
Les fondations	**L'hiver**	**La théorie sans la machine**	**La bascule matérielle**
Neurone formel · Test de Turing · Perceptron	Critique du perceptron · Financements taris	Rétropropagation · Méthodes statistiques	AlexNet · Processeurs graphiques · Transformer

2.1 Soixante-dix ans avant le premier agent conversationnel

L'idée d'un réseau de neurones artificiels précède l'ordinateur personnel de quarante ans. Les premiers modèles datent de **1943**, sous la plume de Warren McCulloch et Walter Pitts. Sept ans plus tard, Alan Turing propose son jeu de l'imitation. En **1958**, Frank Rosenblatt construit le perceptron, premier dispositif capable d'apprendre à classer des données.

Deux moments comptent plus que les autres. En **1969**, Marvin Minsky et Seymour Papert publient *Perceptrons* et démontrent les limites de l'approche ; les financements se tarissent et la discipline hiberne près de vingt ans. En **2012**, AlexNet remporte le concours ImageNet et fait basculer toute la recherche vers les réseaux profonds (deep neural network) — non parce que la théorie avait changé, mais parce que les données et les processeurs graphiques étaient enfin disponibles.

Entre les deux, la rétropropagation décrite en **1986** par Rumelhart, Hinton et Williams avait pourtant rendu l'entraînement de réseaux à plusieurs couches praticable. La théorie attendait le matériel.

Dhamani et Engler, dans *Introduction to Generative AI*, situent d'ailleurs le vrai tournant dès les années 1990 : les méthodes statistiques commencent à battre les méthodes à base de règles, portées par la disponibilité croissante de données et de calcul. **Le moteur de cette histoire n'est pas l'idée, c'est le matériel.**

2.2 2017 : le transformer, et pourquoi ça a tout débloqué

En 2014, des chercheurs proposent un mécanisme appelé **attention** — le mot est identique en français et en anglais, il n'en existe pas de traduction : au lieu de faire défiler un texte morceau par morceau, on laisse le modèle chercher dans toute la séquence d'entrée les passages les plus pertinents pour chaque partie de ce qu'il produit.

Trois ans plus tard, un article au titre resté célèbre, *Attention Is All You Need*, montre qu'on peut jeter tout le reste et ne garder que l'attention. Le gain n'est pas conceptuel, il est industriel : ces modèles, appelés **transformeurs** (*transformers*), sont **parallélisables**, donc entraînables sur des milliers de processeurs en même temps.

Raff, Farris et Biderman formulent le lien de façon nette : la multiplication de matrices rend l'attention efficace sur processeur graphique, parce que ces puces savent effectuer un très grand nombre de multiplications simultanément. Pouvoir traiter d'un coup tous les mots d'une séquence, au lieu de les enchaîner un par un, représente une accélération massive — et c'est le prérequis de tous les modèles de pointe actuels.

Résumé

L'IA générative moderne n'est pas née d'une idée sur l'intelligence. Elle est née d'une architecture qui exploitait mieux le matériel disponible. Et ce que les auteurs de *Introduction to Generative AI* écrivent de l'ouverture au grand public de fin 2022 vaut pour toute cette histoire : ce n'était pas une percée technique isolée, seulement la dernière amélioration itérative d'un domaine qui progressait vite.

Partie 3 — Entraîner n'est pas utiliser

Une confusion coûte cher dès les premières discussions budgétaires : on mélange deux métiers qui n'ont ni le même coût, ni la même machine, ni la même fréquence.

3.1 Deux métiers, deux coûts, deux machines

Entraîner, c'est fabriquer les poids. Des semaines de calcul, des milliers de processeurs graphiques, une fois. **Utiliser**, ou faire de l'inférence, c'est se servir des poids déjà fabriqués. En continu, à chaque requête.

ENTRAÎNEMENT	INFÉRENCE
Fabrique les poids à partir de rien	Utilise des poids déjà fabriqués
Des semaines, des milliers de processeurs graphiques	Quelques secondes par requête
Des centaines de milliers à des millions d'euros	Quelques centimes d'électricité par heure
Se fait une fois	Se fait à chaque usage, en continu
Hors de portée d'une PME, sans exception	Tient sur une machine de bureau

Les chiffres publiés donnent la mesure de l'écart. Raschka détaille le cas d'un modèle ouvert de 7 milliards de paramètres : **184 320 heures de processeur graphique haut de gamme**, deux mille milliards de tokens traités, pour un coût qu'il estime autour de **690 000 dollars**. L'entraînement d'un modèle antérieur, à 175 milliards de paramètres, est estimé à **4,6 millions de dollars**.

À l'autre bout du spectre, le même auteur mesure qu'un simple ajustement de classification sur un petit modèle tourne en **six minutes sur un ordinateur portable** ordinaire. Entre fabriquer les poids et se contenter de les retoucher, il y a environ six ordres de grandeur — et produire une seule réponse coûte encore infiniment moins.

Voilà pour le coût **unitaire**. Il faut maintenant regarder le coût **cumulé**, et le résultat s'inverse.

> ⚠ **Le paradoxe du coût unitaire**
>
> On lit partout que l'entraînement d'un modèle coûte des millions. On en conclut que c'est là que se joue la dépense. **C'est l'inverse.**
> L'entraînement coûte énormément, mais **une seule fois**. Une réponse ne coûte presque rien, mais elle est produite **des milliards de fois**, tous les jours, pendant toute la durée de vie du modèle.
> Au-delà d'un certain volume d'usage, le cumul des réponses dépasse donc le coût unique de la fabrication — et les modèles largement diffusés ont franchi ce seuil depuis longtemps.

C'est ce qui explique un chiffre qui resservira à toute discussion sur l'empreinte de l'IA : **l'inférence est couramment estimée à 80 ou 90 % du calcul consacré à l'IA**. L'estimation vient d'industriels plutôt que d'une mesure indépendante, mais la logique, elle, ne fait pas débat.

L'attention publique se porte sur l'entraînement parce qu'il est **spectaculaire** — des milliers de processeurs, des semaines de calcul, une facture à sept chiffres. La consommation réelle se joue dans l'usage quotidien parce qu'il est **massif** et invisible.

3.2 Ce qu'on obtient au bout : « 7B », « 70B » et le poids du fichier

Après ces semaines de calcul et ces centaines de milliers d'euros, il reste un fichier. Sa taille, elle, se calcule en une multiplication.

Le **B** de « 7B » est le *billion* anglais, c'est-à-dire notre milliard. Un modèle 7B, ce sont sept milliards de nombres. Chaque nombre occupe une place en mémoire, donc le fichier a un poids que vous pouvez établir vous-même. En 16 bits, soit 2 octets par nombre : 7 milliards × 2 = **14 Go**.

Ce n'est pas une extrapolation. Raschka fait ce calcul explicitement dans son chapitre 4 : pour un modèle de 163 009 536 paramètres stockés en 32 bits, soit 4 octets chacun, il obtient **621,83 Mo**, et il conclut que cela illustre la capacité de stockage relativement importante qu'exigent même de petits modèles. Pour la plus grande variante du même modèle, 1 637 792 000 paramètres, il arrive à **6 247,68 Mo**.

CE QUE PÈSE UN MODÈLE SELON SA PRÉCISION

PRÉCISION	OCTETS PAR PARAMÈTRE	MODÈLE 7 MILLIARDS	MODÈLE 70 MILLIARDS
32 bits (float32)	4	28 Go	280 Go
16 bits (float16)	2	14 Go	140 Go
8 bits (int8)	1	7 Go	70 Go
4 bits	0,5	3,5 Go	35 Go

Les lignes du bas correspondent à des modèles **compressés**. La technique s'appelle la **quantization** — on écrit aussi « quantification » en français, mais le terme anglais domine largement : on réduit le nombre de bits par paramètre, avec une perte de précision contrôlée. Dhamani et Engler en donnent la définition la plus économe — elle réduit la taille du modèle et son empreinte mémoire en diminuant le nombre de bits de précision utilisés. François Chollet et Matthew Watson chiffrent le résultat : un modèle quantifié en 8 bits est **quatre fois plus petit tout en restant proche de la précision du modèle d'origine**.

Comparez maintenant les deux ordres de grandeur de cette partie. Le corpus d'entraînement se compte en **milliers de milliards de jetons**, le fichier obtenu en **quelques dizaines de giga-octets**. C'est le rapport entre ce qu'un modèle a lu et ce qu'il en garde : il n'archive rien, il ne conserve que des paramètres ajustés.

Retenez surtout que la taille d'un modèle n'est pas une caractéristique commerciale, mais une multiplication que vous pouvez refaire : nombre de paramètres × octets par paramètre = poids du fichier, et donc mémoire nécessaire pour le charger.

3.3 Fine-tuning et LoRA : adapter sans tout refaire

Entre fabriquer un modèle et se contenter de l'utiliser, il existe une voie intermédiaire : reprendre un modèle existant et le réajuster sur ses propres données. Là encore, le procédé a une définition officielle.

> **apprentissage par transfert — transfer learning** — « Apprentissage automatique qui consiste à soumettre un modèle préentraîné à une nouvelle phase d'entraînement sur un volume réduit de données relatives à un domaine cible, afin que le modèle ainsi obtenu génère des réponses pertinentes pour ce domaine. » *Journal officiel du 6 septembre 2024*

C'est ce qu'on appelle couramment le **fine-tuning**. Le texte officiel en donne aussi l'intérêt : il « réduit la durée et le coût de l'apprentissage automatique dans un domaine cible, notamment lorsque le volume de données disponibles dans ce domaine est faible ».

LoRA (*low-rank adaptation*) en est la variante la plus répandue. Son principe tient en une phrase : au lieu de retoucher tous les poids, on n'en entraîne qu'une petite fraction, rangée dans un fichier séparé.

Raschka le chiffre : sur un modèle de 124 millions de paramètres, LoRA n'en rend entraînables que **2,7 millions**, soit environ 2 % — pour une qualité finale équivalente dans son cas.

Ce que ça permet

CE QUE LE FINE-TUNING PERMET

- ✓ Adapter un modèle sans la machine ni le budget d'un entraînement complet — quelques minutes sur un ordinateur portable, contre des semaines sur des milliers de processeurs
- ✓ Lui faire adopter un comportement : suivre des instructions, respecter un format de sortie, rester dans un périmètre donné

CE QUE LORA AJOUTE

- ✓ Garder un seul modèle de base et lui appliquer plusieurs adaptations — une par client, une par usage — au lieu de stocker plusieurs modèles complets
- ✓ Revenir en arrière : les poids d'origine restent intacts, l'adaptation est un fichier séparé qu'on applique ou qu'on retire

> ▲ **Ce que ça ne résout pas**
>
> Le fine-tuning n'est pas la réponse à « je veux qu'il connaisse nos documents ». Trois raisons.
>
> **Il fait oublier.** Entraîner sur de nouvelles données sans rejouer les anciennes fait perdre au modèle une partie de ce qu'il savait.
>
> **Il expose les données.** Raff, Farris et Biderman sont formels : n'ajustez pas un modèle sur des données que vous voulez garder privées.
>
> **Il ne se corrige pas.** On ne retire pas un document d'un modèle ajusté. On le retire d'un index — et c'est précisément le rôle de la **génération augmentée par récupération** (*retrieval-augmented generation*, RAG), qui va chercher les passages utiles et les glisse dans le contexte avant que le modèle ne réponde.

3.4 Ce que ça implique : vous n'entraînerez rien

Autant le dire franchement, ça simplifie tout le reste : **personne, dans une PME, n'entraîne un modèle de langage.** Ni vous, ni votre prestataire.

Raff et ses coauteurs chiffrent le ticket d'entrée à cent mille dollars minimum, et à cent millions si l'ambition est de rivaliser avec les laboratoires établis. Dhamani et Engler ajoutent qu'obtenir le nombre de processeurs graphiques nécessaires n'est pas simple, même avec l'argent.

> ◆ **Résumé**
>
> La seule question réaliste, pour une organisation qui n'est pas un laboratoire de recherche, est **quel modèle existant faire tourner, et où**.
>
> Tout ce qui suit dans ce document ne parle donc que d'inférence — l'usage de poids déjà fabriqués par d'autres, dont vous savez désormais ce qu'ils ont coûté et ce qu'ils pèsent.

Partie 4 — Le token, unité de compte de toute l'IA générative

La partie 1 a défini le jeton textuel. Cette partie explique comment il est fabriqué, et pourquoi il finit sur votre facture.

Car c'est bien l'unité dans laquelle se comptent le prix, la mémoire et la vitesse. Le *Journal officiel* le dit lui-même dans sa définition : le nombre de jetons « permet d'estimer et de facturer le coût de fonctionnement des grands modèles de langage commerciaux ». Une définition terminologique qui parle de facturation, c'est assez rare pour être relevé.

4.1 À quoi ressemble un texte découpé en tokens

Un mot courant vaut souvent un token. Un mot long, rare ou mal orthographié en vaut plusieurs. Voici un exemple réel, publié par Raschka et reproductible par n'importe qui avec l'outil de tokenisation d'usage courant :

```
CODE                                                    Copier

Texte  : "Hello, do you like tea? ◁endoftext▷ In the sunlit terraces
          of someunknownPlace."
^^
Tokens : [15496, 11, 466, 345, 588, 8887, 30, 220, 50256, 554, 262,
          4252, 18250, 8812, 2114, 286, 617, 34680, 27271, 13]
```

L'exemple est en anglais, et ce n'est pas un hasard — j'explique pourquoi deux sections plus loin. Le second exemple du même auteur montre mieux encore le découpage en sous-mots : la suite inventée `Akwirw ier` se décompose en `"Ak"`, `"w"`, `"ir"`, `"w"`, `"ier"` et l'espace qui les sépare, soit six identifiants pour neuf caractères.

Un ordre de grandeur pour situer : le vocabulaire des modèles de cette famille compte **50 257 tokens**, quand un adulte anglophone connaît environ 30 000 mots.

4.2 Comment un tokenizer est fabriqué

Le programme qui découpe le texte s'appelle un **tokenizer**. Le *Journal officiel* n'a pas retenu de terme français pour l'outil lui-même — seulement pour l'opération, la « segmentation en jetons textuels ».

Le procédé s'appelle l'encodage par paires d'octets (*byte pair encoding*, BPE). Il tient en quatre étapes, sans mathématiques.

LA CONSTRUCTION D'UN VOCABULAIRE

- On part des caractères isolés : « a », « b », « c »…
- On repère, dans un corpus immense, les paires de caractères qui apparaissent le plus souvent ensemble
- On fusionne ces paires en une nouvelle unité — « d » et « e » deviennent « de », fréquent dans « depuis », « demande », « monde »
- On recommence quelques dizaines de milliers de fois

Le résultat est le **vocabulaire** : une table figée, livrée avec le modèle, qu'on ne modifie plus.

Deux conséquences méritent d'être énoncées, parce qu'elles ne sont jamais dites. Le vocabulaire est **un choix de l'éditeur**, pas une propriété de la langue. Et il **reflète le corpus** sur lequel il a été construit — donc, très majoritairement, l'anglais.

4.3 Combien coûte un texte : anglais, autres langues, chiffres, noms propres

C'est ici que la mécanique devient une ligne de facture. Raff, Farris et Biderman donnent le chiffre : en prenant l'anglais comme référence, **le coût de traitement d'une requête en allemand ou en italien est supérieur d'environ 50 %**. Certaines langues, moins représentées dans les corpus, coûtent **plus de douze fois** le tarif anglais.

Le français, langue romane à l'orthographe accentuée, appartient à cette famille de langues pénalisées. Je ne donnerai pas de pourcentage précis pour le français : je n'ai pas trouvé de source qui le mesure sérieusement, et il n'est pas question d'en inventer un.

CE QUI COÛTE CHER EN TOKENS, ET POURQUOI

TYPE DE CONTENU	EFFET	CAUSE
Anglais courant	**Référence**	Le corpus d'entraînement est majoritairement anglophone
Allemand, italien	**~50 % de plus**	Moins représentés, découpage en sous-mots plus fin
Langues peu dotées	**Plus de 12 fois**	Presque absentes du corpus, découpage caractère par caractère
Chiffres	**Variable**	Les modèles qui ont un token par chiffre calculent mieux
Noms propres, références	**Imprévisible**	Découpage arbitraire, sans lien avec la graphie

Deux exemples concrets valent mieux qu'un argumentaire. Dans le vocabulaire d'un modèle largement diffusé, **« Amoxicillin » et sa faute de frappe « Amoxicillan » ne partagent aucun token**. Si votre métier manipule des références produits, des noms de molécules ou des codes article, le risque est réel et il ne se voit pas.

Second exemple : les modèles disposant d'un token par chiffre tendent à mieux calculer que les autres, ce qui explique une partie des échecs en arithmétique que vous avez pu constater.

La traduction métier est directe. **À contenu égal, une facture d'API en français est structurellement plus élevée qu'en anglais.**

4.4 Du token au vecteur : l'embedding

Le terme employé est l'anglais **embedding** — on trouve « plongement » en français, mais l'usage professionnel a tranché pour l'anglais.

Une dernière étape avant que le modèle ne calcule quoi que ce soit. Chaque token est converti en une liste de nombres — un vecteur. Raschka décrit la couche qui fait ce travail comme une simple opération de consultation : on va chercher une ligne dans une grande table, à l'index correspondant au token.

La table en question n'est pas petite. Pour un modèle courant, elle mesure 50 257 lignes sur 768 colonnes, soit près de 39 millions de nombres rien que pour cette étape. Les modèles plus gros montent à 12 288 dimensions par token.

Ce qui rend ces vecteurs utiles, c'est que **la proximité entre deux listes encode la proximité de sens** — apprise statistiquement, jamais déclarée. C'est ainsi que « facture » et « note de frais » finissent voisins sans qu'aucune règle ne le dise. C'est aussi le mécanisme sur lequel repose toute recherche documentaire assistée par IA.

Résumé

Raff, Farris et Biderman assortissent ce constat d'une réserve importante : rien ne garantit que ces relations sémantiques se forment pendant l'entraînement, et elles ne sont pas une vérité découverte sur le monde — seulement **un reflet des données qui ont alimenté le processus**. Un modèle entraîné sur un corpus biaisé produira des voisinages biaisés, sans que rien ne le signale.

Partie 5 — La fenêtre de contexte et le cache

Tout le monde a entendu parler de « fenêtre de contexte ». Presque personne ne dit d'où vient son coût. C'est la partie la plus technique de ce document, et la plus rentable à comprendre.

5.1 Ce que contient réellement une fenêtre de contexte

La fenêtre de contexte contient **tout** : l'instruction système, l'historique complet de la conversation, les documents que vous avez joints, et la réponse en cours de production. Le tout se compte en tokens, l'unité de la partie précédente.

Ce n'est pas une mémoire persistante. À chaque nouvel appel, l'intégralité est renvoyée au modèle.

> ⚠ **Le contresens sur la mémoire**
>
> Le modèle ne se souvient de rien entre deux requêtes. **Rien.**
> Quand une conversation semble se poursuivre, c'est le logiciel client qui rejoue tout l'historique à chaque message. Le modèle, lui, redécouvre l'ensemble à chaque fois.
> Raff, Farris et Biderman insistent sur ce point : un modèle de langage ne peut ni planifier, ni prendre d'engagement, ni suivre un état interne. Et il est statique par défaut — quelle que soit sa précision lundi, elle sera exactement la même mardi, quel que soit le nombre de conversations entre-temps.
> Conséquence pratique : plus la conversation est longue, plus chaque nouveau message coûte cher. Le coût croît avec l'historique, pas avec votre dernière phrase.

Cette mécanique explique un malentendu très répandu, et c'est probablement celui qui touche le plus de monde.

> ⚠ **Les « espaces de travail » ne sont pas un entraînement**
>
> Les assistants grand public proposent de créer un espace où l'on dépose ses documents et ses consignes permanentes. Beaucoup d'utilisateurs sont convaincus d'y avoir « entraîné » le modèle sur leurs données.
>
> **Il n'en est rien, et vous savez maintenant pourquoi.** Les poids ne bougent pas d'un iota. Selon le volume déposé, ces documents sont soit rechargés dans la fenêtre de contexte à chaque conversation, soit récupérés par morceaux à la demande. Dans les deux cas, c'est du contexte — pas de l'entraînement.
>
> Deux conséquences très concrètes. Ce que vous avez déposé est relu, et donc payé, **à chaque échange** — pas une fois pour toutes. Et si vous retirez un document de l'espace, il disparaît réellement, ce qui serait impossible sur un modèle véritablement ajusté.

5.2 L'attention relit tout à chaque mot — le cache existe pour éviter ça

Pour produire le mot suivant, le mécanisme d'attention compare le token courant à tous les tokens précédents. Chaque token dispose pour cela de deux vecteurs, appelés **clé** et **valeur**.

L'analogie donnée par Raschka est directement transposable, parce qu'elle vient du monde des bases de données. La **requête** est la recherche en cours. La **clé** est l'index qui permet de retrouver l'information. La **valeur** est le contenu effectivement renvoyé.

Le point crucial est le suivant, et l'auteur le formule explicitement : même pour calculer un seul résultat, **les vecteurs clé et valeur de tous les éléments de l'entrée sont nécessaires**.

Sans optimisation, produire le millième mot d'une réponse exigerait donc de tout recalculer depuis le premier. C'est pour éviter cela qu'existe le **cache clé-valeur** (*key-value cache*, ou *KV cache*) : on garde en mémoire les calculs déjà faits pour chaque token traité, au lieu de les refaire.

C'est un échange explicite, et il faut le voir comme tel. **On paie de la mémoire pour économiser du calcul.** Le vocabulaire technique distingue d'ailleurs deux phases : le *prefill*, où le modèle avale d'un coup tout le texte d'entrée, et le *decode*, où il produit la réponse token par token. Ces deux termes n'ont pas d'équivalent français en usage.

5.3 Ce que pèse un contexte

Ce cache occupe de la place, et cette place se calcule.

L'ordre de grandeur tient en une ligne : sur un modèle courant, **chaque token de contexte coûte environ 128 kilo-octets de mémoire**. Pris isolément, c'est dérisoire. Multiplié par la longueur d'une conversation, ça ne l'est plus du tout.

CE QUE COÛTE LE CONTEXTE, EN PLUS DU MODÈLE

LONGUEUR DE CONTEXTE	ÉQUIVALENT APPROXIMATIF	POIDS DU CACHE
2 048 tokens	~8 pages	**256 Mio**
8 192 tokens	~30 pages	**1 Gio**
32 768 tokens	~120 pages	**4 Gio**
131 072 tokens	~480 pages	**16 Gio**

Un giga-octet de mémoire pour une trentaine de pages de texte. Et le mot important est *en plus* : ce giga-octet s'ajoute au poids du modèle, il ne s'y substitue pas. Un modèle de 14 Go qui traite trente pages occupe 15 Go, pas 14.

Deux facteurs font varier ce chiffre.

La **longueur du contexte**, évidemment — et elle joue linéairement, comme le montre le tableau. Doubler le texte donné à lire double la mémoire consommée.

Et l'**architecture du modèle**, ce qui est beaucoup moins intuitif. Deux modèles de taille identique peuvent avoir des caches très différents, parce que le coût dépend du nombre de couches et de têtes d'attention, pas du nombre de paramètres. C'est d'ailleurs ainsi que les modèles récents tiennent des contextes plus longs à mémoire égale : ils font partager un même jeu de clés et de valeurs à plusieurs têtes, ce qui divise la facture d'autant.

→ **Pour refaire le calcul vous-même**

Cette section peut se lire sans ce qui suit. Elle est là pour ceux qui veulent vérifier plutôt que croire.

Poids du cache par token = 2 (clés et valeurs) × nombre de couches × nombre de têtes clé-valeur × dimension par tête × taille d'un nombre.

Sur une configuration publiée — 32 couches, 8 têtes clé-valeur, 128 dimensions par tête, nombres en 16 bits :

`2 × 32 × 8 × 128 × 2 = 131 072 octets`, soit **128 Ko par token**.

`× 8 192 tokens = 1 073 741 824 octets`, soit exactement **1 Gio**.

Ce modèle compte 32 têtes de requête mais seulement 8 têtes clé-valeur : quatre requêtes se partagent le même jeu. C'est le **regroupement des têtes d'attention** (*grouped-query attention*, GQA). Sans lui, le même calcul donnerait 512 Ko par token — quatre fois plus.

5.4 « Un million de tokens » : d'où sort ce chiffre

Les chiffres de contexte annoncés sur les fiches produit sont des **maximums supportés par l'architecture**, pas des capacités offertes en permanence. Trois raisons à cela.

D'abord, le tableau ci-dessus. Le cache grandit linéairement avec l'usage réel, et finit par dépasser la mémoire disponible bien avant d'atteindre le maximum théorique.

Ensuite, l'efficacité sur les très longs contextes reste discutée. Je pèse mes mots : Raff, Farris et Biderman qualifient la question de domaine d'étude actif, et je n'affirmerai pas plus qu'eux. Ce qui est en revanche établi, c'est qu'**au-delà de sa fenêtre, un modèle perd le fil de ce qui a été discuté au début de la conversation**.

Enfin, côté fournisseur, le contexte est facturé. Un contexte long n'est pas offert, il est vendu.

◆ **Résumé**

Un modèle ne coûte pas seulement ce que pèse son fichier. Il coûte ce fichier **plus** le cache de tout ce qu'on lui donne à lire. C'est la raison pour laquelle deux organisations qui utilisent le même modèle peuvent avoir des factures — ou des besoins matériels — très différents.

Lire une fiche de modèle

Ce qu'on appelle couramment une *model card* porte un nom officiel en français : **notice de modèle préentraîné**, définie au *Journal officiel* du 6 septembre 2024 comme « l'ensemble des informations qui accompagnent un modèle préentraîné et qui définissent les caractéristiques de son jeu de données d'entraînement, son paramétrage et le cadre de son utilisation ».

Toute la mécanique exposée dans ce document se ramène à une compétence pratique : savoir lire une fiche technique de modèle sans se faire raconter d'histoires. Voici comment procéder, dans l'ordre.

Les quatre questions à poser

"01"

Combien de paramètres, et en quelle précision ?

"Multipliez le nombre de paramètres par les octets par paramètre. Vous obtenez le poids du fichier, donc la mémoire minimale pour le charger. Sans cette précision, une annonce en « milliards de paramètres » ne dit rien du besoin réel."

"02"

Quelle est l'architecture — couches, têtes, dimensions ?

"Ces trois nombres déterminent le coût du contexte, indépendamment du nombre de paramètres. Un modèle plus petit peut avoir un cache plus lourd. S'ils ne sont pas publiés, le coût du contexte est imprévisible."

"03"

Quelle taille de contexte, et à quel coût mémoire ?

"Appliquez la formule de la partie 5. Le chiffre annoncé sur la fiche est un maximum architectural, pas une capacité gratuite. Comptez le cache en plus du modèle, jamais à la place."

"04"

Quel tokenizer, et quel vocabulaire ?

"La taille du vocabulaire et le corpus d'origine déterminent le coût de traitement de vos textes. Un vocabulaire construit sur un corpus anglophone traitera vos documents français plus cher, à contenu identique."

Conclusion

Un modèle de langage est un fichier de nombres. Il ne contient aucun de vos documents, ne retient rien d'une conversation à l'autre, et n'a jamais été entraîné à dire la vérité — seulement à produire du texte qui ressemble à ce qu'il a lu. Ces quatre faits ne sont pas des limites provisoires que la prochaine version corrigera : ce sont les propriétés de l'objet.

De cette nature découle tout le reste. Le fichier a un poids, calculable, qui décide de la mémoire nécessaire. Le texte qu'on lui envoie se compte en tokens, dont le prix varie avec la langue et le vocabulaire. Et le contexte qu'on lui donne à lire se paie en mémoire, en plus du modèle, selon une formule que vous savez désormais appliquer.

Ce qui a rendu cette technologie possible n'est pas une percée sur l'intelligence, mais une architecture qui exploitait mieux le matériel disponible. C'est pour cette raison que les contraintes exposées ici tiendront : elles ne dépendent d'aucun produit, d'aucun éditeur et d'aucune version.

Reste la question que ce document n'a pas traitée, délibérément — que faire de tout cela ? Quel modèle choisir, sur quelle machine le faire tourner, à quel moment une exécution locale se justifie face à un service en ligne. Ces arbitrages dépendent de prix, d'offres et de matériels qui changent tous les six mois, et ils méritent d'être traités séparément. Ils le sont sur le blog MP-i.

CRITIQUE	Avant tout engagement, exigez le nombre de paramètres ET la précision. Sans les deux, aucun dimensionnement n'est possible.
IMPORTANT	Vérifiez le surcoût de tokenisation sur vos propres textes métier avant de bâtir un budget d'API.
UTILE	Formez les personnes qui décident, pas seulement celles qui intègrent — la majorité des erreurs de cadrage vient du vocabulaire, pas de la technique.

Références et ressources

Ouvrages de référence

→ Raff E., Farris D., Biderman S. (2025), *How Large Language Models Work*, Manning : www.manning.com ↗

→ Raschka S. (2025), *Build a Large Language Model (From Scratch)*, Manning : www.manning.com ↗

→ Dhamani N., Engler M. (2026), *Introduction to Generative AI*, 2e édition, Manning : www.manning.com ↗

→ Chollet F., Watson M. (2026), *Deep Learning with Python*, 3e édition, Manning : www.manning.com ↗

→ Trask A. (2019), *Grokking Deep Learning*, Manning : www.manning.com ↗

→ Bahree A. (2024), *Generative AI in Action*, Manning : www.manning.com ↗

→ Freed G., Jacobs T., Rózsa M. (2025), *Effective Conversational AI*, Manning : www.manning.com ↗

Terminologie officielle

→ Commission d'enrichissement de la langue française, *50 termes clés de l'intelligence artificielle* — termes, expressions et définitions publiés au Journal officiel : www.culture.gouv.fr ↗

→ FranceTerme — base terminologique officielle, plus de 9 500 termes : www.culture.fr ↗

Articles fondateurs

→ Vaswani A. et al. (2017), *Attention Is All You Need* : arxiv.org ↗

→ Hu E. et al. (2021), *LoRA: Low-Rank Adaptation of Large Language Models* : arxiv.org ↗

→ McCulloch W., Pitts W. (1943), *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics : doi.org ↗

Configurations techniques citées

→ Meta AI (2024), *The Llama 3 Herd of Models* — configuration d'architecture utilisée pour le calcul du cache : arxiv.org ↗

Glossaire

Les termes marqués **[JO]** ont une définition officielle publiée au *Journal officiel de la République française* par la Commission d'enrichissement de la langue française ; la date de publication est indiquée. Les autres sont d'usage professionnel, sans équivalent français officiel à ce jour.

Apprentissage automatique — machine learning, ML **[JO 09/12/2018]**

Processus par lequel un algorithme améliore ses performances sans intervention d'un programmeur, en répétant son exécution sur des jeux de données. Sous-champ de l'intelligence artificielle.

Apprentissage profond — deep learning ** [JO 09/12/2018]**

Apprentissage automatique utilisant un réseau de neurones artificiels à grand nombre de couches. Le « profond » compte les couches, il ne désigne aucune compréhension plus profonde.

Attention — attention

Mécanisme par lequel un modèle compare le token courant à tous les précédents pour déterminer lesquels sont pertinents. Le terme est identique en français et en anglais. Introduit en 2014, généralisé par le transformeur en 2017.

Cache clé-valeur — key-value cache, KV cache

Mémoire de travail conservant les calculs déjà effectués pour chaque token traité, afin de ne pas les refaire à chaque nouveau mot produit. Son poids croît linéairement avec la longueur du contexte.

Embedding — embedding

Représentation d'un token sous forme de liste de nombres, dont la proximité avec d'autres listes encode une proximité de sens apprise statistiquement. On trouve « plongement » en français, peu employé.

Encodage par paires d'octets — byte pair encoding, BPE

Procédé de construction d'un vocabulaire, qui fusionne itérativement les paires de caractères les plus fréquentes d'un corpus en unités plus longues.

Fenêtre de contexte — context window

Quantité totale de texte qu'un modèle peut traiter en une seule requête, exprimée en tokens. Elle contient l'instruction, l'historique, les documents joints et la réponse en cours.

Fine-tuning — fine-tuning

Réajustement d'une partie des poids d'un modèle existant sur un corpus spécifique. Moins coûteux qu'un entraînement complet, mais destructif et non ciblable.

Grand modèle de langage — large language model, LLM **[JO 06/09/2024]**

Modèle génératif qui, à partir de grands volumes de données textuelles, calcule des probabilités d'enchaînements de jetons textuels en vue de la génération automatique de texte ou de code.

Hallucination — hallucination

Énoncé faux mais vraisemblable produit par un modèle. Conséquence directe de son objectif d'entraînement, qui récompense la ressemblance avec le corpus, non l'exactitude.

Inférence — inference

Utilisation d'un modèle déjà entraîné pour produire une réponse. À distinguer de l'entraînement, qui fabrique les poids. Représente l'essentiel du calcul consacré à l'IA.

Instruction générative — prompt **[JO 06/09/2024]**

Consigne donnée par un utilisateur à un modèle génératif, généralement en langue naturelle, qui décrit la tâche à accomplir. Le terme « invite » est officiellement déconseillé.

Intelligence artificielle — artificial intelligence, AI **[JO 09/12/2018]**

Champ interdisciplinaire ayant pour objet la compréhension des mécanismes de la cognition et leur imitation par un dispositif matériel et logiciel.

Intelligence artificielle générative — generative AI, GenAI **[JO 06/09/2024]**

Branche de l'intelligence artificielle mettant en œuvre des modèles génératifs, qui vise à produire des contenus textuels, graphiques ou audiovisuels.

Jeton textuel — text token, token **[JO 06/09/2024]**

Donnée élémentaire d'un grand modèle de langage, constituée d'une suite de caractères obtenue par segmentation automatique d'un texte. Unité de compte de la facturation, de la mémoire et de la vitesse.

LoRA — low-rank adaptation

Méthode de fine-tuning n'entraînant qu'une fraction des paramètres — de l'ordre de 2 % — en gardant les matrices d'adaptation séparées du modèle d'origine.

Modèle génératif — generative model ** [JO 06/09/2024]**

Résultat d'un apprentissage automatique destiné à générer des données comparables à celles de son jeu d'entraînement. Le terme « modèle de fondation » est officiellement déconseillé.

Neurone artificiel — artificial neuron **[JO 09/12/2018]**

Dispositif à plusieurs entrées et une sortie, qui simule certaines propriétés du neurone biologique. Ses paramètres sont ajustables — c'est là que se loge l'apprentissage.

Notice de modèle préentraîné — model card **[JO 06/09/2024]**

Ensemble des informations accompagnant un modèle préentraîné : caractéristiques du jeu d'entraînement, paramétrage et cadre d'utilisation.

Paramètre — parameter

Aussi appelé poids (*weight*). Variable interne du modèle, ajustée automatiquement pendant l'entraînement. Nombre de paramètres × octets par paramètre = poids du fichier.

Prefill / decode — prefill / decode

Les deux phases d'une réponse. Le *prefill* traite d'un coup tout le texte d'entrée ; le *decode* produit la réponse token par token. Sans équivalent français en usage.

Quantization — quantization

Réduction du nombre de bits utilisés par paramètre, afin de diminuer la taille du modèle et son empreinte mémoire, au prix d'une perte de précision contrôlée. On écrit aussi « quantification ».

Regroupement des têtes d'attention — grouped-query attention, GQA

Architecture où plusieurs têtes de requête partagent un même jeu de clés et de valeurs, ce qui divise d'autant le poids du cache clé-valeur.

Réseau de neurones artificiels — artificial neural network **[JO 09/12/2018]**

Ensemble de neurones artificiels interconnectés qui constitue une architecture de calcul.

Tokenizer — tokenizer

Programme qui découpe un texte en tokens selon un vocabulaire figé, construit lors de l'entraînement et livré avec le modèle. Le *Journal officiel* ne nomme que l'opération, la « segmentation en jetons textuels ».

Transformeur — transformer **[JO 06/09/2024]**

Réseau de neurones artificiels réalisant un traitement parallèle des données d'entraînement. Introduit en 2017, il est à la base de tous les modèles génératifs actuels. Sa propriété décisive est d'être parallélisable, donc entraînable à grande échelle.

À propos de l'auteur

Mattieu Pottier

Consultant indépendant en transformation digitale — MP-i · Mattieu Pottier Indépendant

Consultant freelance spécialisé en ERP Odoo, SIG (QGIS, PostGIS), applications web et automatisation. Intervient sur des projets de cadrage, conception, développement et accompagnement au changement pour des PME, ETI et collectivités. mp-i.pro

MP-i — Mattieu Pottier Indépendant