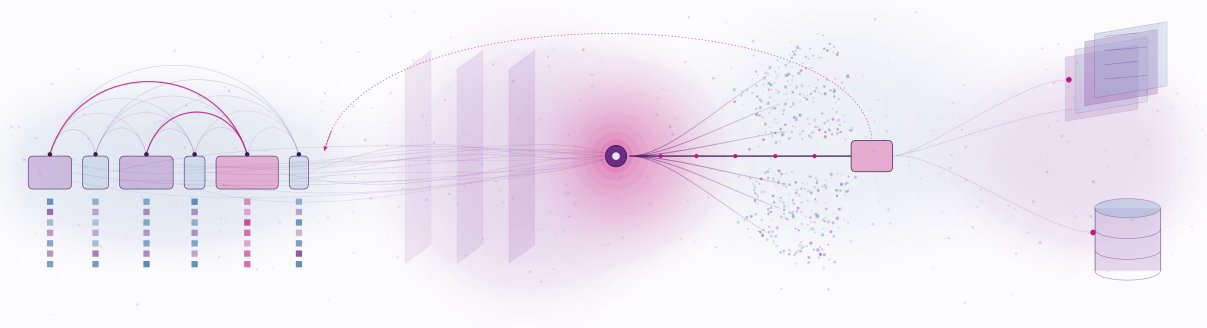
IA GÉNÉRATIVE • MODÈLES DE LANGAGE • MÉCANISMES INTERNES

Que se passe-t-il réellement lorsque vous posez une question à une IA ?

Des tokens à la réponse : comment un modèle prédit, exploite le contexte, se trompe parfois, et pourquoi il est aujourd'hui entouré d'outils.



Des fragments mis en relation, un réseau qui calcule, un éventail de suites possibles : une seule est retenue, puis réinjectée.

Frédéric Ros

Technopole d'Orléans — LAB'IA Loire Valley

La thèse de cet article. Un modèle de langage est entraîné à prédire le prochain *token*. Cette tâche apparemment triviale lui fait apprendre des régularités linguistiques et factuelles d'une richesse considérable. Mais une réponse plausible n'est pas automatiquement vraie : les IA d'aujourd'hui se comprennent comme des *systèmes* associant un modèle, un contexte, des outils, des vérifications et des humains.

Article de vulgarisation • Version 2.1 • octobre 2026

© 2026 Frédéric Ros — Technopole d'Orléans, LAB'IA Loire Valley. Tous droits réservés. Toute reproduction, représentation ou diffusion, totale ou partielle, est interdite sans autorisation écrite préalable de l'auteur.

Cofinancé par l'Union européenne

AVANT DE COMMENCER

L’essentiel en cinq minutes

Si vous ne lisez qu’une page de cet article, lisez celle-ci.

1 **Une IA conversationnelle ne consulte pas une base de faits : elle prédit.**

Le modèle de langage (LLM) produit, à chaque pas, le fragment de texte le plus plausible compte tenu de ce qui précède. La réponse n’existe nulle part avant d’être écrite.

2 **Votre phrase est découpée en tokens, puis chaque mot est lu à la lumière des autres.**

Le texte est converti en fragments numérotés. Le mécanisme d’attention, au cœur de l’architecture Transformer publiée en 2017, permet ensuite à chaque fragment de puiser dans le reste de la phrase de quoi préciser son sens.

3 **La réponse se construit un token à la fois, avec une part de hasard réglable.**

Le modèle calcule des probabilités ; un réglage extérieur, la température, décide à quel point le choix peut s’écarter du candidat le plus probable. C’est pourquoi la même question peut recevoir deux réponses différentes.

4 **Plausible n’est pas vrai.**

Le mécanisme qui produit les bonnes réponses produit aussi les erreurs, souvent avec la même assurance. Une réponse bien écrite n’est pas une réponse vérifiée.

5 **Le modèle est fiable quand la matière est sous ses yeux.**

Résumer, traduire ou reformuler un document fourni réussit bien ; retrouver de mémoire une date, une source ou un chiffre est fragile. D’où les IA d’aujourd’hui, entourées de documents, d’outils et de vérifications.

Trois façons de lire la suite

5 min

Cette page

Les cinq idées ci-dessus, et les cinq réflexes pratiques de la conclusion, page 18.

20 min

Le fil principal

Les bandeaux violets en tête de section, le texte courant et les encadrés « L’idée à retenir ». Sautez les encadrés « Sous le capot » et les sections marquées ⧫.

45 min

Tout l’article

Encadrés techniques compris, pour comprendre le détail des mécanismes et leurs limites.

Vous préférez manipuler ?

Un démonstrateur interactif accompagne cet article. En quelques minutes, vous y découpez une phrase en tokens, observez l’attention à l’œuvre, faites varier la température pendant que le modèle écrit, et comparez une réponse donnée de mémoire à une réponse appuyée sur un document.

Il est téléchargeable à côté de l’article sur la page Presse du LAB’IA (QR code ci-contre, ou `labia-loirevalley.fr/presse`). Il s’ouvre d’un double-clic dans votre navigateur, sans installation ni connexion. On peut commencer par lui, puis revenir au texte.

L'EXPÉRIENCE QUE NOUS FAISONS TOUS

Vous tapez :

« Qui a inventé Internet ? »

Deux secondes plus tard :

« Tim Berners-Lee et Robert Cailliau. »

La réponse est immédiate, bien formée, assurée. Elle mentionne deux noms réels, une collaboration réelle, un domaine cohérent.

Elle est fausse.

Tim Berners-Lee a inventé le **World Wide Web** au CERN, à la fin des années 1980, avec la collaboration déterminante de Robert Cailliau. Internet, lui, existait déjà : c'est le réseau de réseaux construit progressivement à partir des travaux sur la commutation de paquets, ARPANET et la famille de protocoles TCP/IP. Le Web est une application *qui circule sur* Internet. Confondre les deux, c'est confondre l'autoroute et le camion.

Cette erreur est un excellent point de départ, parce qu'elle est *typique*.[1] Elle n'est pas grossière : elle est plausible. Les deux notions sont associées des millions de fois dans les textes, les noms apparaissent constamment ensemble, la phrase « X a inventé Internet » est une tournure ordinaire. Le modèle a produit la suite la plus naturelle, pas la plus exacte.

L'IDÉE À RETENIR

Comprendre comment une IA produit ses réponses, c'est comprendre du même mouvement pourquoi elle se trompe. Le mécanisme qui fabrique les bonnes réponses est exactement celui qui fabrique les mauvaises. Il n'y a pas deux modes de fonctionnement, un « normal » et un « défaillant ».

Quatre idées reçues à démonter

Avant d'entrer dans la mécanique, écartons ce que la plupart des gens imaginent lorsqu'ils envoient une question à un assistant conversationnel.

Ce qu'on imagine	Ce qui se passe
Elle lit et comprend ma phrase comme un lecteur humain.	Elle la découpe en morceaux, les convertit en nombres, et les fait interagir dans un réseau de calcul.
Elle consulte une gigantesque base de données de faits.	Un modèle seul ne consulte rien. Il n'y a pas de fiche « Internet ». (Sauf si un outil de recherche a été branché : voir §13.)
Elle réfléchit, puis rédige la réponse qu'elle a trouvée.	Elle produit un morceau de texte, puis le suivant, puis le suivant. La réponse n'existe nulle part avant d'être écrite.
Quand elle ne sait pas, elle le dit.	Rien dans le mécanisme de base ne produit un « je ne sais pas ». Il faut le lui apprendre, ou l'organiser autour d'elle.

POINT DE VIGILANCE

Une nuance importante dès l'ouverture. Dire qu'un modèle « ne comprend rien » est aussi discutable que dire qu'il « comprend comme nous ». Ce que l'on peut affirmer sans risque, c'est ceci : *son mécanisme élémentaire est la prédiction du token suivant à partir du contexte.* Ce que cette mécanique produit comme capacités, et le nom qu'il convient de leur donner, reste un débat scientifique ouvert et

[1] Exemple construit pour les besoins de l'article. Les assistants actuels répondent en général correctement à cette question précise, justement parce qu'elle est devenue un piège célèbre ; le mécanisme d'erreur qu'elle illustre, lui, reste d'actualité sur des sujets moins balisés.

sérieux. Cet article décrit le mécanisme ; il laisse le débat ouvert.

Le trajet que nous allons faire

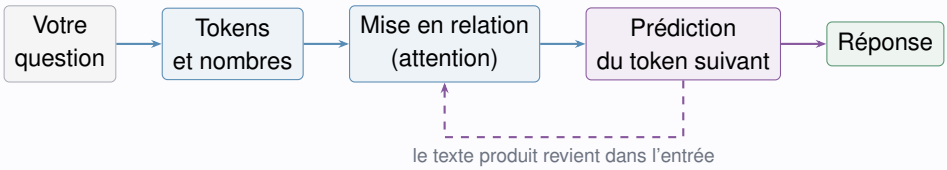


Figure 1. Le trajet complet d'une question. L'étape décisive est la boucle : chaque morceau de texte produit est réinjecté dans l'entrée avant de produire le suivant.

La phrase devient des morceaux : les tokens

PREMIER DÉCALAGE AVEC L'INTUITION

Le modèle ne voit ni vos mots, ni vos lettres. Il voit une suite d'identifiants numériques correspondant à des fragments de texte appelés **tokens**.

Un token, c'est une unité de découpage apprise sur un très grand corpus. Selon sa fréquence, un fragment peut être un mot entier, un morceau de mot, un signe de ponctuation, ou même un espace suivi d'un mot. Les séquences fréquentes obtiennent leur propre token ; les séquences rares sont reconstituées en assemblant plusieurs fragments.

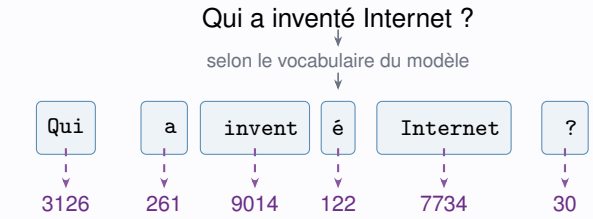


Illustration simplifiée. La segmentation exacte dépend du *tokenizer* employé, et les identifiants ci-dessus sont **fictifs** : ils ne correspondent à aucun modèle réel.

Figure 2. La tokenisation. Remarquez que « inventé » est ici coupé en deux et que l'espace qui précède un mot fait partie du token.

SOUS LE CAPOT – lecture avancée facultative

Pourquoi découper ainsi ? Un vocabulaire de mots entiers serait ingérable : il faudrait une entrée pour chaque forme conjuguée, chaque nom propre, chaque faute de frappe, dans chaque langue. Les algorithmes de tokenisation (BPE, WordPiece, Unigram) construisent un compromis : un vocabulaire de taille fixe (typiquement de 30 000 à 200 000 entrées selon les modèles), capable de représenter *n'importe quelle* chaîne de caractères en la décomposant. Rien n'est jamais « hors vocabulaire » : au pire, un mot inconnu est reconstitué caractère par caractère.

Trois conséquences très concrètes

Ce détail d'ingénierie a des effets que tout utilisateur rencontre sans en connaître la cause.

1. Les langues ne sont pas égales. Les vocabulaires ont historiquement été optimisés sur des corpus majoritairement anglophones. Une phrase française consomme souvent 15 à 30 % de tokens de

plus que son équivalent anglais ; certaines langues à écriture non latine, bien davantage. Comme la facturation des API et la fenêtre de contexte se comptent en tokens, cela se traduit directement en coût et en place disponible.

2. Le modèle « voit » mal les lettres. Demandez combien de « r » contient le mot « strawberry » : le modèle ne manipule pas des lettres mais deux ou trois fragments opaques. Il ne peut pas les compter en les regardant ; il ne peut que produire la suite plausible. C'est aussi pourquoi les jeux de mots, les anagrammes, les acrostiches et le comptage de caractères sont des faiblesses structurelles : non pas un manque d'intelligence, mais un problème de granularité d'entrée.

3. Les nombres sont fragiles. Selon le découpage, « 1789 » peut devenir un seul token ou trois fragments. L'arithmétique repose alors sur des motifs appris plutôt que sur un calcul. D'où la pratique consistant à faire exécuter les calculs par un vrai interpréteur, appelé comme outil.

EN PRATIQUE

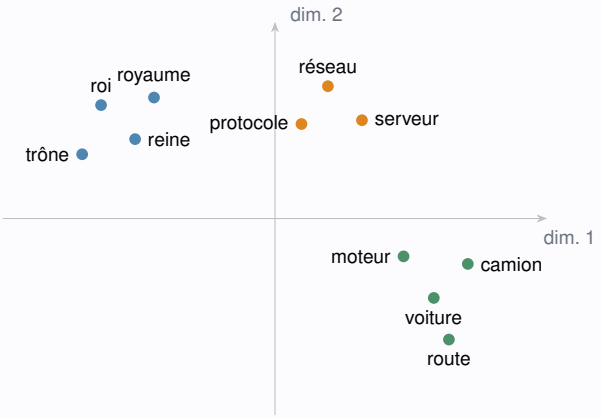
Quand une tâche dépend de la structure exacte des caractères (compter, épeler, respecter un nombre de signes, manipuler du code sensible aux espaces), ne faites pas confiance à l'intuition du modèle : demandez-lui explicitement de passer par un outil de calcul, ou vérifiez. Ce n'est pas de la méfiance de principe, c'est la connaissance d'une limite précise et identifiable.

Des nombres aux représentations

LES MOTS COMME POSITIONS

Chaque identifiant de token est converti en un vecteur : une longue liste de nombres, qui décrit une *position* dans un espace à plusieurs centaines, voire plusieurs milliers de dimensions.

L'idée à retenir est simple et forte : dans cet espace, la proximité a un sens. Des fragments employés dans des contextes similaires se retrouvent dans des régions voisines. Le modèle n'a pas reçu de dictionnaire ; ces positions ont été apprises, uniquement en observant quels mots apparaissent avec quels autres, sur des volumes de texte considérables.



Projection illustrative en deux dimensions d'un espace qui en comporte en réalité des centaines ou des milliers. Aucune des deux dimensions dessinées ici ne correspond à une grandeur interprétable du modèle.

Figure 3. Les fragments employés dans des contextes proches occupent des régions proches.

SOUS LE CAPOT – lecture avancée facultative

La distinction que presque tout le monde saute. Il faut soigneusement séparer deux objets :

L'embedding initial : la position de départ associée à un token dans le dictionnaire du modèle. Elle est fixe : le fragment « avocat » entre toujours avec le même vecteur.

La représentation contextualisée : ce que devient ce vecteur après avoir traversé les couches du réseau. Elle dépend entièrement de la phrase.

Autrement dit, « avocat » commence au même endroit dans « l'avocat a plaidé devant le juge » et dans « j'ai coupé l'avocat pour la salade », mais il n'y *finit* pas du tout au même endroit. C'est le contexte qui déplace la représentation. Ce déplacement progressif, couche après couche, est le cœur du fonctionnement d'un Transformer.

POINT DE VIGILANCE

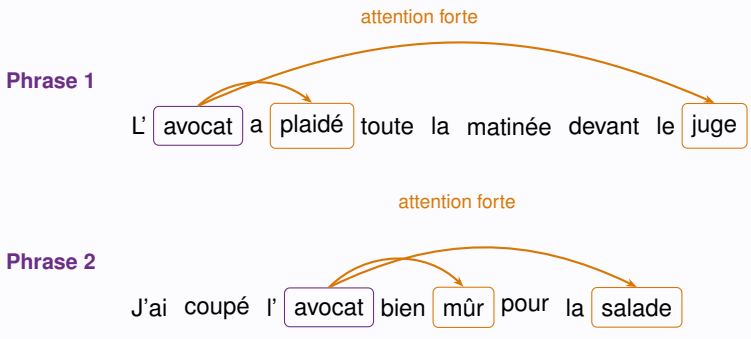
Vous avez peut-être rencontré la fameuse analogie « roi − homme + femme ≈ reine ». Elle est jolie et elle a réellement été observée, mais sur les *embeddings statiques* des années 2013–2015, une génération de modèles antérieure. Ce n'est pas une bonne description de ce que font les Transformers modernes, dont l'essentiel du travail consiste précisément à *sortir* des représentations fixes.

Mettre le contexte en relation : l'attention

L'IDÉE QUI A TOUT CHANGÉ

Chaque token calcule quelles parties du contexte lui sont les plus utiles pour mettre à jour sa propre représentation. C'est le mécanisme d'**attention**, et c'est le fondement de l'architecture Transformer.

Formulé autrement : au lieu de lire la phrase de gauche à droite en gardant un résumé de ce qui précède (ce que faisaient les architectures récurrentes antérieures), le Transformer met en relation tous les éléments du contexte, en pondérant ces relations. Un mot ambigu va « puiser » l'information dont il a besoin là où elle se trouve, même si elle est loin.



On ne représente ici que l'attention *issue du token* « avocat » (encadré violet), et seulement pour une tête et une couche. Un modèle réel calcule simultanément des dizaines de motifs d'attention différents, sur des dizaines de couches successives.

Figure 4. Le même fragment, deux contextes : ce ne sont pas les mêmes voisins qui comptent, donc ce ne sera pas la même représentation en sortie.

SOUS LE CAPOT – lecture avancée facultative

Comment se calcule une pondération d'attention ? Sans équations : chaque token émet trois

signaux dérivés de sa représentation courante. Une *requête* (« ce que je cherche »), une *clé* (« ce que je propose ») et une *valeur* (« ce que je transmets si on me choisit »). On compare la requête de chaque token aux clés de tous les autres ; plus la correspondance est forte, plus la valeur correspondante pèse dans la mise à jour. La somme pondérée devient la nouvelle représentation.

Deux amplifications : le mécanisme est répliqué en plusieurs *têtes* qui apprennent chacune un type de relation (accord grammatical, référence d'un pronom, dépendance longue distance…), et l'ensemble est empilé sur plusieurs *couches*, chaque couche travaillant sur le résultat de la précédente. La représentation d'un token en sortie de la dernière couche est le produit de ce raffinement successif.

L'IDÉE À RETENIR

Le coût de l'attention croît approximativement avec le *carré* de la longueur du contexte : doubler la longueur d'entrée ne double pas le travail, il le quadruple à peu près. C'est la raison technique pour laquelle les contextes très longs coûtent cher, et pourquoi tant d'efforts de recherche portent sur des variantes plus économes.

Générer, token après token

LE POINT QUE PRESQUE PERSONNE NE SOUPÇONNE

Le modèle ne rédige jamais une réponse. Il produit un token. Puis il recommence entièrement, avec ce token en plus dans l'entrée. Puis encore. Des centaines de fois.

À chaque tour, la sortie du réseau n'est pas un mot mais une *distribution de probabilités* sur l'ensemble du vocabulaire : un score pour chacune des dizaines de milliers d'entrées possibles. Un composant extérieur au réseau, l'algorithme de décodage, choisit alors quel token retenir.

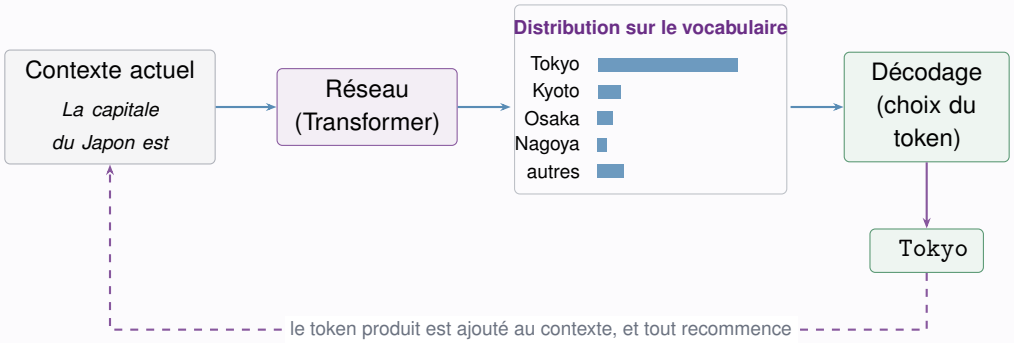


Figure 5. Une itération de génération. Le réseau produit des probabilités ; c'est le décodage, en aval, qui choisit.

POINT DE VIGILANCE

On lit souvent que « le modèle choisit le mot le plus probable ». C'est inexact, et cette imprécision empêche de comprendre un phénomène courant. Le modèle *calcule* des probabilités ; le système, lui, peut prendre le maximum, ou bien *échantillonner* parmi les candidats les plus crédibles. C'est précisément pourquoi la même question posée deux fois peut donner deux réponses différentes, sans que le modèle ait changé.

SOUS LE CAPOT – lecture avancée facultative

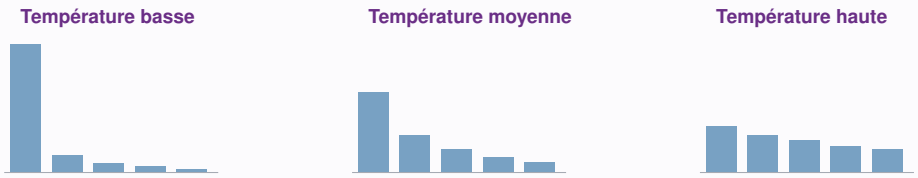
Les réglages du décodage.

Température. Elle accentue ou aplatit les écarts de probabilité. Basse, elle rend la sortie très prévisible et répétitive ; haute, elle laisse davantage de chances aux candidats moins évidents, donc plus de variété, mais aussi plus de risque de dérapage.

Top-k. On ne conserve que les k candidats les mieux classés, et on tire parmi eux.

Top-p (échantillonnage par noyau). On conserve les candidats les mieux classés jusqu'à atteindre une masse de probabilité cumulée p. Le nombre de candidats retenus s'adapte donc à la situation : peu quand le modèle est net, davantage quand il hésite.

Décodage déterministe. Température nulle ou recherche en faisceau : on vise la séquence la plus probable. Utile pour de l'extraction ou de la traduction, souvent plat et redondant pour de la rédaction.



Même distribution de départ, trois réglages. À gauche, la réponse sera presque toujours la même. À droite, les candidats secondaires ont une chance réelle d'être retenus : créativité accrue, fiabilité diminuée.

Figure 6. L'effet de la température sur les probabilités de sortie.

EN PRATIQUE

Si un assistant vous donne deux réponses différentes à la même question, ce n'est ni un bug ni un caprice : c'est l'échantillonnage. Et c'est exploitable : poser deux ou trois fois la même question factuelle et comparer les réponses est un test de robustesse rudimentaire mais étonnamment révélateur. Des réponses qui divergent signalent une zone où le modèle n'a pas d'appui solide.

Pourquoi c'est cohérent, et pourquoi « plus long » n'est pas « mieux »

LA COHÉRENCE VIENT DE LA BOUCLE

Chaque token produit est réintégré à l'entrée. Le modèle ne garde aucune intention en réserve : à chaque tour, sa prédiction repose sur l'intégralité de ce qui précède.

C'est la propriété dite *autorégressive*. Elle explique la cohérence des textes produits : la phrase en cours n'est pas rédigée à partir d'une intention gardée en mémoire, mais reconstruite à chaque pas à partir de tout ce qui est déjà écrit. Le début de la réponse contraint la suite, puissamment.



À chaque ligne, l'entrée du modèle s'allonge d'un token et la prédiction repose sur tout ce qui précède. Aucune intention n'est gardée d'un tour à l'autre : seul le texte déjà écrit fait foi.

Figure 7. La boucle autorégressive : le contexte grandit, le calcul recommence.

POINT DE VIGILANCE

Il serait tentant d'en conclure que plus le contexte est long, meilleure est la réponse. **C'est faux**, et c'est une confusion coûteuse en pratique. Un contexte *pertinent* aide. Un contexte simplement *long* peut :

- noyer l'information décisive au milieu du bruit ;
- contenir des éléments contradictoires que le modèle devra arbitrer, parfois mal ;
- dépasser la fenêtre disponible, auquel cas le début est tronqué sans avertissement visible ;
- dégrader l'exploitation d'informations situées en milieu de document (un effet documenté et robuste) ;
- coûter sensiblement plus cher et plus lentement.

La bonne question n'est pas « puis-je tout mettre ? » mais « qu'est-ce qui est strictement utile ici ? ».

SOUS LE CAPOT – lecture avancée facultative

Fenêtre de contexte n'est pas mémoire. La fenêtre est la quantité maximale de tokens que le modèle peut prendre en entrée en un seul appel : la question, l'historique de la conversation, les documents joints, les consignes système et la réponse en cours de production, le tout additionné. Ce n'est pas un souvenir : c'est un champ de vision. Hors de cette fenêtre, il n'existe rien.

Quand un assistant paraît « se souvenir » d'une conversation d'il y a trois semaines, ce n'est jamais le modèle qui se souvient. C'est un dispositif extérieur qui a stocké des éléments et les a réinsérés dans le contexte avant l'appel. La distinction n'est pas un détail : elle détermine entièrement ce qui est effaçable, auditable et corrigible.

Une précision d'ingénieur. En pratique, les systèmes conservent en cache les calculs déjà effectués pour les tokens précédents, afin de ne pas les refaire à chaque pas. C'est une optimisation de vitesse : le résultat est le même qu'une relecture complète, et ce cache ne survit pas à l'appel.

Résumer, traduire, reformuler : pourquoi ces tâches réussissent si bien

UNE DISTINCTION TRÈS UTILE EN PRATIQUE

Il existe deux familles de demandes, dont les taux de réussite n'ont rien à voir. Ce qui les sépare : la matière nécessaire est-elle *dans le contexte*, ou faut-il aller la chercher *dans les paramètres* ?

Les utilisateurs sont souvent surpris par un contraste qu'ils constatent sans se l'expliquer : le même assistant produit un résumé remarquable d'un rapport de quarante pages, puis invente une référence bibliographique deux minutes plus tard. Ce n'est pas de l'inconstance. Ce sont deux tâches de natures radicalement différentes.

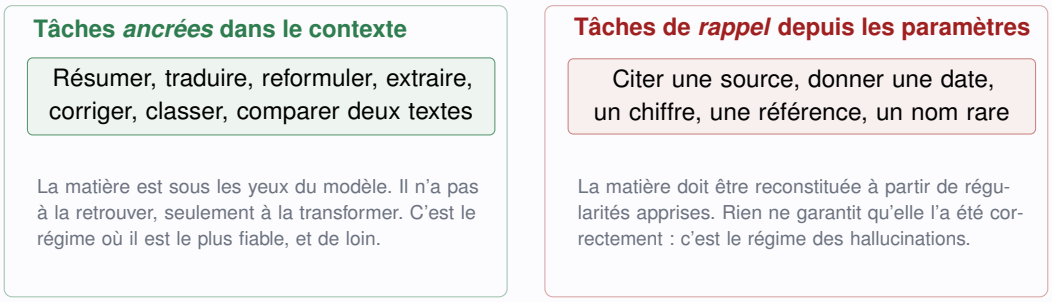


Figure 8. Deux régimes de fonctionnement pour un seul et même mécanisme.

L'IDÉE À RETENIR

La règle pratique qui en découle : **plus vous placez la matière dans le contexte, plus la réponse est fiable**. C'est la justification profonde du RAG (section 13), et c'est aussi la raison pour laquelle « colle ton document et demande » fonctionne tellement mieux que « pose la question de mémoire ».

Ce qu'est réellement un résumé produit par un LLM

Il faut se débarrasser d'une image fausse : le modèle ne sélectionne pas des phrases importantes qu'il recopierait. Il ne surligne rien. Il fait exactement ce qu'il fait toujours, prédire le token suivant, mais avec le document entier dans son contexte, et une consigne qui oriente la génération vers un texte plus court couvrant les mêmes points.

Le résumé est donc *reconstruit*, mot à mot, et non extrait. C'est précisément ce qui lui donne sa qualité : il peut reformuler, regrouper des éléments dispersés, changer de registre, s'adresser à un public particulier. Une simple extraction de phrases n'en serait jamais capable.

SOUS LE CAPOT – lecture avancée facultative

La contrepartie de cette reconstruction. Puisque le texte est regénéré et non copié, quelques phénomènes deviennent possibles, et méritent d'être connus.

Le lissage. Une nuance exprimée une seule fois dans le document, au détour d'une subordonnée, pèse peu face à un thème répété vingt fois. Le résumé tend vers ce qui est saillant statistiquement, pas nécessairement vers ce qui est décisif.

Les liaisons ajoutées. Pour produire un texte fluide, le modèle insère des connecteurs : « par conséquent », « ce qui explique que ». Ces liens de causalité peuvent ne pas figurer dans l'original. Le résumé est alors plus *affirmatif* que la source.

Le mélange des registres. Ce que le modèle sait par ailleurs sur le sujet peut s'infiltrer dans le résumé, sans marquage. Le lecteur ne peut pas distinguer ce qui vient du document de ce qui vient du modèle.

L'inégalité de traitement selon la position. Sur un document long, le début et la fin sont généralement mieux restitués que le milieu.

POINT DE VIGILANCE

La limite la plus sérieuse n'est pourtant aucune de celles-là. C'est celle-ci : **le modèle ne connaît pas votre critère d'importance**. Un résumé n'est pas une opération objective. Le même rapport se résume différemment selon qu'on l'aborde en juriste, en financier, en acheteur ou en syndicaliste. Sans consigne, le modèle produit le résumé le plus générique possible : statistiquement raisonnable, et potentiellement inutile pour vous, parce qu'il aura écarté exactement ce qui vous importait.

EN PRATIQUE

Quatre demandes qui transforment la qualité d'un résumé.

1. **Donnez le critère et l'usage.** « Résume ce contrat du point de vue des engagements financiers qui nous lient au-delà de douze mois » vaut infiniment mieux que « résume ce contrat ».
2. **Demandez des points d'ancrage.** Exiger, pour chaque point, la section ou la phrase d'origine rend le résumé vérifiable en quelques secondes, au lieu de le prendre sur parole.
3. **Demandez ce qui a été écarté.** « Quels éléments du document n'apparaissent pas dans ce résumé et pourraient être importants ? » est une question courte au rendement remarquable.
4. **Séparez les sources.** « Distingue ce qui figure dans le document de ce que tu ajoutes par ailleurs » limite nettement le mélange des registres.

L'IDÉE À RETENIR

Ces quatre réflexes se ramènent à un seul principe, qui vaut bien au-delà du résumé : *ce que vous ne spécifiez pas, le modèle le comble par ce qui est statistiquement le plus ordinaire*. Formuler une demande, c'est d'abord réduire l'espace de ce qui est plausible.

Comment fabrique-t-on un assistant ?

♦ *Section d'approfondissement : elle peut se sauter en première lecture.*

TROIS PHASES, TROIS OBJECTIFS DIFFÉRENTS

On résume souvent l'apprentissage d'un LLM à « prédire le mot suivant ». C'est vrai de la première phase seulement. Un assistant conversationnel est le produit d'au moins trois étapes bien distinctes.

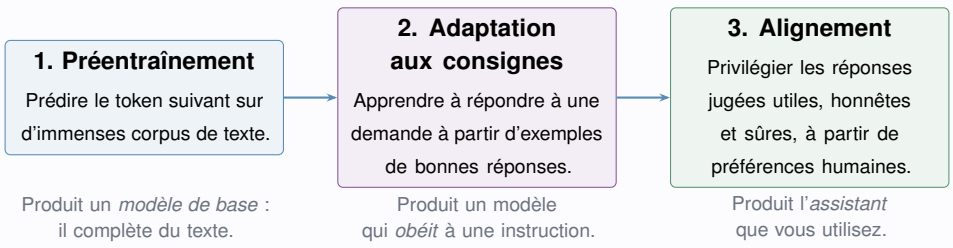


Figure 9. De la prédiction brute à l'assistant : trois phases aux objectifs distincts.

Phase 1 : le préentraînement

Le modèle parcourt d'énormes volumes de texte. À chaque position, on lui masque la suite, on lui demande de la prédire, on mesure son erreur, on ajuste très légèrement ses paramètres. Répétée un nombre astronomique de fois, cette procédure d'une simplicité désarmante fait émerger des régularités : syntaxe, accords, structures argumentatives, associations factuelles, styles, notions de code, correspondances entre langues.

> **L'IDÉE À RETENIR**
>
> C'est le point conceptuel central. Personne n'a « programmé » la grammaire française, ni la géographie, ni la notion de sarcasme. Ces compétences sont des *effets secondaires* d'un objectif unique : bien prédire la suite. Pour prédire correctement la fin d'un raisonnement, il faut avoir capté quelque chose de sa structure. C'est ce qui rend cette tâche apparemment triviale beaucoup moins triviale qu'elle n'en a l'air.

Phase 2 : l'adaptation aux consignes

Un modèle de base ne répond pas : il continue. Posez-lui « Quelle est la capitale du Japon ? » et il pourrait aussi bien enchaîner « Quelle est la capitale de la Corée ? Quelle est la capitale du Vietnam ? ». Une liste d'exercices est une continuation parfaitement plausible. Il faut donc lui enseigner un comportement : quand on te pose une question, tu réponds. Cela se fait par ajustement supervisé, sur des milliers d'exemples de paires demande/réponse soignées.

Phase 3 : l'alignement sur des préférences

Reste à départager les réponses *acceptables* : celle qui est claire contre celle qui est confuse, celle qui reconnaît une incertitude contre celle qui affirme. On présente à des évaluateurs humains des réponses concurrentes, on apprend un modèle de leurs préférences, et on ajuste le comportement en conséquence. Les travaux publiés sur ce type de méthode ont montré un résultat marquant : un modèle nettement plus petit mais correctement aligné pouvait être préféré à un modèle beaucoup plus gros qui ne l'était pas. **La taille seule ne produit pas l'utilité.**

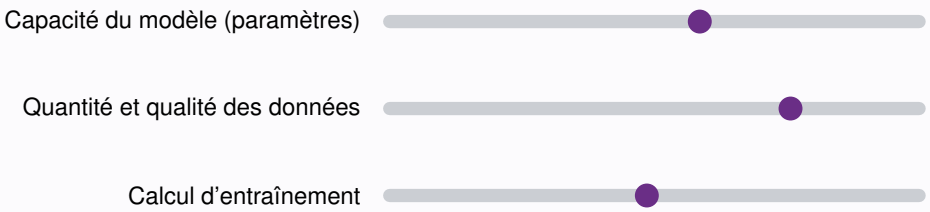
> **SOUS LE CAPOT** – lecture avancée facultative
>
> **Et le prompt système ?** Il existe une quatrième couche, qui n'est pas de l'apprentissage : le *prompt système*. Ce sont des consignes placées en tête du contexte à chaque appel : rôle, ton, interdits, format attendu. Cela ne modifie aucun paramètre : c'est du texte, ajouté à l'entrée, qui oriente la génération. D'où deux propriétés importantes : c'est modifiable instantanément, et c'est comparativement fragile, puisque le contenu du contexte peut entrer en concurrence avec ces consignes.
>
> Bien distinguer ces quatre leviers (préentraînement, ajustement, alignement, prompt) est ce qui permet de répondre correctement à la question « peut-on faire en sorte que le modèle fasse X ? ». Selon le X, la réponse n'est pas au même étage, ni au même prix.

La taille : trois curseurs, pas un

♦ *Section d'approfondissement : elle peut se sauter en première lecture.*

« PLUS GROS » N'EST PAS UNE STRATÉGIE

La performance d'un modèle ne résulte pas de son nombre de paramètres, mais de l'équilibre entre trois grandeurs.



Déplacer un seul curseur sans les autres produit des rendements décevants. Un très grand modèle sous-entraîné, ou entraîné sur des données médiocres, est un mauvais investissement, pas un bon modèle.

Figure 10. Trois curseurs à régler ensemble.

Le résultat le plus instructif de la littérature sur ce sujet est contre-intuitif : à budget de calcul donné, on a longtemps construit des modèles *trop grands et trop peu entraînés*. Un modèle nettement plus petit, mais nourri de beaucoup plus de données, peut surpasser des modèles plusieurs fois plus volumineux. L'arbitrage optimal n'est pas « maximiser les paramètres », mais « équilibrer paramètres et données pour le calcul dont on dispose ».

POINT DE VIGILANCE

Évitons un mot piégeux : les paramètres ne sont pas de la **mémoire**. On lit souvent que les modèles « stockent » des connaissances comme une base stocke des fiches. Ce n'est pas ce qui se passe. Les paramètres encodent des régularités distribuées ; une information factuelle n'est pas localisée quelque part, elle est reconstituée. Voilà pourquoi on ne peut pas « supprimer un fait » d'un modèle comme on supprime une ligne d'une table, et pourquoi les connaissances factuelles précises sont un usage fragile de ce type de système.

SOUS LE CAPOT – lecture avancée facultative

Pourquoi les petits modèles reviennent en force. Trois raisons convergent.

La distillation. Un grand modèle sert d'enseignant à un modèle plus compact, qui atteint sur un périmètre restreint des performances proches pour une fraction du coût.

Le calcul au moment de la réponse. Laisser un modèle moyen décomposer, vérifier et recommencer peut donner de meilleurs résultats que d'interroger un modèle géant une seule fois. On déplace le budget de l'entraînement vers l'usage.

Les contraintes réelles. Latence, coût, confidentialité, souveraineté des données, exécution locale. Un modèle qui tourne sur l'infrastructure de l'organisation, sans envoyer les documents ailleurs, a une valeur que le classement brut ne mesure pas.

Sur un cas d'usage bien délimité, un modèle modeste correctement adapté et bien alimenté en documents bat régulièrement un modèle généraliste plus puissant employé sans préparation.

POINT DE VIGILANCE

Un mot sur les chiffres qui circulent. Les tailles exactes des modèles commerciaux récents ne sont pour la plupart pas publiées : les rapports techniques mentionnent explicitement ne divulguer ni l'architecture, ni le nombre de paramètres, ni la composition des données. Toute frise comparative de ce type relève

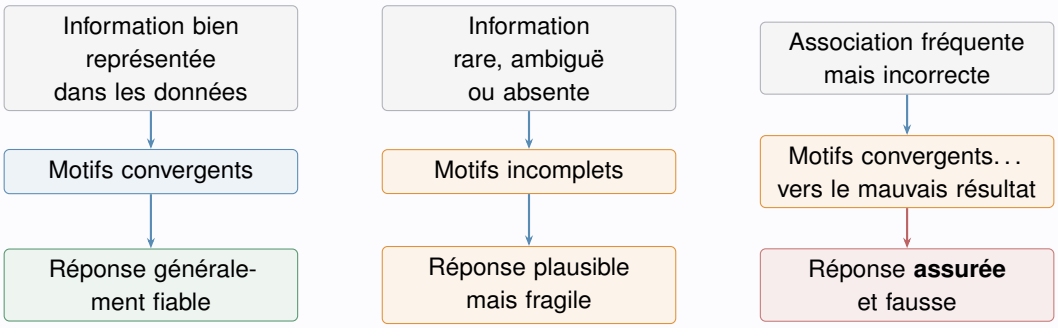
donc largement de l'estimation, et vieillit en quelques mois. C'est délibérément que cet article n'en propose pas.

Pourquoi se trompe-t-il ? Anatomie d'une hallucination

NI MENSONGE, NI PANNE

Une hallucination est une réponse fluide, bien formée, plausible, et fausse. Ce n'est pas un dysfonctionnement du mécanisme : c'est ce mécanisme, appliqué là où il n'a pas d'appui.

Commençons par écarter l'explication trop simple, celle qu'on lit partout. On dit volontiers : quand le modèle ne sait pas, sa distribution devient « plate », il hésite, donc il se trompe. L'image est utile mais *insuffisante*, et par endroits trompeuse. Car un modèle peut être parfaitement net et parfaitement faux. C'est même le cas le plus dangereux : sur notre exemple d'ouverture, l'association Berners-Lee/Internet est très fortement représentée dans les textes. Le modèle n'hésite pas. Il se trompe avec assurance.



Le troisième cas est celui que l'explication habituelle par « l'hésitation » ne capte pas. C'est aussi le plus difficile à détecter pour un lecteur.

Figure 11. Trois trajectoires, dont une seule mène à une réponse fiable.

SOUS LE CAPOT – lecture avancée facultative

Les causes réelles, plus variées qu'on ne croit.

1. **Connaissance absente ou mal apprise.** Le sujet est trop rare, trop récent, ou n'était présent qu'à travers des sources contradictoires.
2. **Conflit entre motifs.** Deux régularités fortes pointent vers des réponses différentes ; l'arbitrage penche du mauvais côté.
3. **Association fréquente mais erronée.** Le corpus lui-même véhicule la confusion (notre cas Internet/Web).
4. **Formulation trompeuse.** Une question qui présuppose un fait inexistant conduit le modèle à le confirmer, parce que la continuation la plus naturelle du texte est de le confirmer.
5. **Objectif de plausibilité.** Produire une réponse bien formée est ce que le modèle a appris à faire ; l'absence de réponse n'est pas une continuation naturelle du texte.
6. **Erreur accumulée en cours de génération.** Une première approximation devient un fait acquis dans le contexte, et contraint toute la suite du raisonnement.
7. **Contexte incorrect fourni en entrée.** Un document mal extrait, obsolète, ou hors sujet.
8. **Raisonnement intermédiaire défaillant.** L'erreur naît dans les étapes de travail que le modèle écrit avant de répondre, ou dans la manière dont le système enchaîne ses appels (point développé

à la section 11).

POINT DE VIGILANCE

La cause 4 mérite un test que vous pouvez faire tout de suite. Demandez à un assistant de résumer les conclusions d'une étude parfaitement imaginaire, en lui donnant un titre crédible et un nom d'auteur. La tentation de fournir un résumé est forte, car un résumé est la continuation naturelle attendue. Les modèles récents résistent nettement mieux à ce piège que leurs prédécesseurs, mais l'expérience reste instructive, et elle apprend beaucoup sur la manière de formuler ses propres demandes.

L'IDÉE À RETENIR

Point souvent négligé : **le modèle n'est pas mécaniquement obligé de répondre**. Rien dans son architecture ne l'y contraint. Ce qui l'y pousse, c'est que ses données d'apprentissage contiennent surtout des réponses, et que dire « je ne sais pas » n'est presque jamais la continuation la plus probable d'une question.

Cela peut se corriger, à condition de le vouloir explicitement. On peut entraîner un modèle à exprimer son incertitude ; on peut construire autour de lui un système qui déclenche une recherche documentaire, demande une précision, ou escalade vers un humain. C'est un choix de conception, pas une fatalité technique.

Ce qu'une IA ne peut pas garantir seule

FORMULER LES LIMITES SANS LES CARICATURER

Les formules chocs sur les limites de l'IA sont souvent trop absolues. Précisons-les : elles n'en deviennent que plus utiles.

Formule courante	Formulation défendable
« Une IA ne sait pas ce qui est vrai. »	Le modèle ne dispose pas, par son seul mécanisme de génération, d'un accès garanti à la vérité. Une forte probabilité linguistique n'est pas une preuve factuelle.
« Elle ne peut pas vérifier ses réponses. »	Elle ne peut pas *garantir seule* la véracité d'une réponse. Elle peut relire, chercher des contradictions, appeler une source, mais cette vérification reste faillible et suppose des critères externes.
« Elle ne sait pas qu'elle se trompe. »	Elle ne dispose pas d'un signal fiable d'erreur. La fluidité d'une réponse n'est corrélée ni à son exactitude, ni à la solidité de ce qui la fonde.
« Ses connaissances sont figées. »	Les paramètres sont figés après entraînement, mais le contexte, lui, est alimentable en temps réel : documents, recherche, outils. C'est là que se joue l'actualité.
« Elle est biaisée. »	Elle reflète les régularités de ses données et les choix opérés lors de l'alignement. Ces effets sont mesurables, et partiellement corrigibles, à condition d'être cherchés.

Une forte probabilité linguistique n'est pas une preuve factuelle.

EN PRATIQUE

Cette nuance a une traduction opérationnelle immédiate. Sur une tâche donnée, la question n'est pas « peut-on faire confiance à l'IA ? » mais : *où placer le point de vérification, et à quel coût ?* Un brouillon de courriel ne demande aucun dispositif. Une synthèse réglementaire en exige un, explicite et documenté. Le même modèle, la même mécanique, et deux niveaux de contrôle radicalement différents.

Pourquoi certaines IA semblent-elles réfléchir ?

TROIS NIVEAUX, UN SEUL MÉCANISME

Vous avez remarqué que certaines réponses arrivent instantanément et que d'autres prennent trente secondes, parfois en affichant des étapes. Le modèle n'a pourtant pas changé de nature.

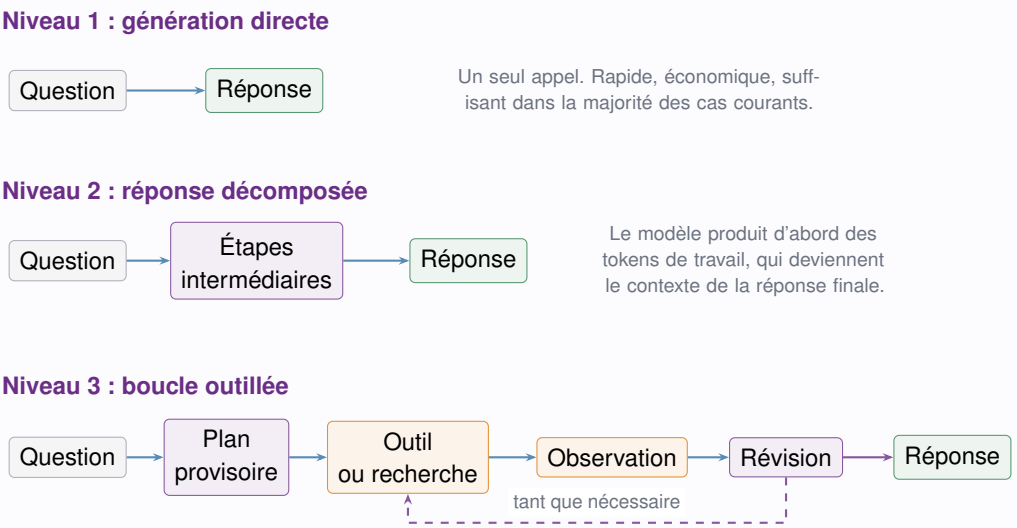


Figure 12. Ce qui change entre les trois niveaux n'est pas le modèle, mais le système placé autour de lui.

L'IDÉE À RETENIR

À retenir : **dans les trois cas, le Transformer fait exactement la même chose** : il produit des tokens, un par un, à partir de son contexte. Ce qui diffère, c'est ce qu'on lui demande de produire, combien de fois on l'appelle, et ce qu'on insère dans son contexte entre deux appels. Le « raisonnement » n'est pas une faculté supplémentaire du modèle : c'est une organisation du travail autour de lui.

Pourquoi la décomposition améliore les résultats

Parce que le modèle produit *un* token à la fois, avec une quantité de calcul bornée à chaque pas. Un problème qui exigerait plusieurs transformations enchaînées ne peut pas être résolu en un seul saut vers la réponse. En revanche, si les étapes intermédiaires sont écrites, elles entrent dans le contexte, et chaque nouveau pas s'appuie sur elles. Le texte produit sert d'espace de travail externe. Écrire les étapes, c'est littéralement se donner les moyens de les utiliser.

POINT DE VIGILANCE

Une mise en garde importante et peu diffusée : la trace de raisonnement affichée n'est pas nécessairement la cause réelle de la réponse. C'est du texte généré, avec les mêmes mécanismes et donc les mêmes failles. Une justification peut être élégante, ordonnée, convaincante, et reconstruite après coup. Une explication lisible n'est pas une preuve d'un cheminement correct. Lisez ces étapes comme un brouillon utile, pas comme un certificat.

SOUS LE CAPOT – lecture avancée facultative

Pourquoi trente secondes ? Ce délai ne mesure pas un effort cognitif : il mesure une quantité de travail système. Pendant ce temps, le dispositif a pu enchaîner plusieurs appels au modèle, lancer une ou plusieurs recherches, exécuter du code, ouvrir et découper des documents, comparer des résultats partiels, détecter une incohérence et recommencer.

C'est un arbitrage explicite : on dépense du calcul au moment de la réponse plutôt qu'au moment de l'entraînement. Sur les tâches où la vérification est plus facile que la résolution (calcul, code, extraction de données), ce transfert est particulièrement rentable.

Spécialiser un modèle : deux leviers distincts

ADAPTER LE COMPORTEMENT, OU FOURNIR LA MATIÈRE

Deux approches, souvent confondues, qui ne répondent pas du tout à la même question.

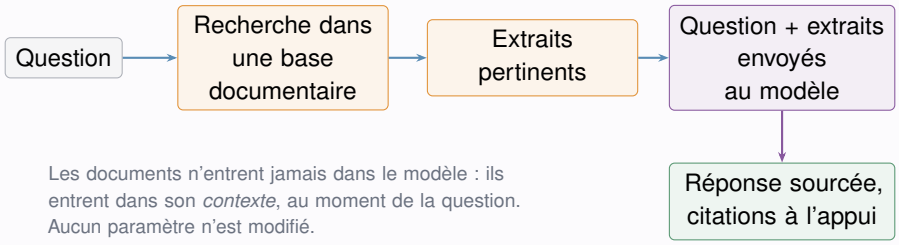


Figure 13. Le RAG (génération augmentée par recherche), étape par étape.

	Ajustement fin (fine-tuning)	RAG
Ce qui change	Les paramètres du modèle	Le contenu du contexte
À quoi ça sert	Le comportement, le style, le format, le vocabulaire métier, le type de tâche	Fournir des faits précis, récents, nombreux, vérifiables
Actualisation	Nécessite un nouvel entraînement	Immédiate : on met à jour les documents
Traçabilité	Faible : la réponse ne cite pas ses sources	Forte : les extraits utilisés sont identifiables
Coût	Élevé, ponctuel	Modéré, récurrent (indexation, recherche)
Point faible	Coûteux à maintenir ; peu fiable pour injecter des faits	Dépend entièrement de la qualité de la recherche

POINT DE VIGILANCE

On lit couramment que « le fine-tuning apprend de nouvelles connaissances au modèle ». C'est possible dans une certaine mesure, mais ce n'est pas sa fonction la plus fiable, ni la plus économique. Pour injecter des faits précis, nombreux, récents et *vérifiables*, le RAG est généralement mieux adapté : il rend la source consultable, ce que l'ajustement fin ne permet pas.

L'IDÉE À RETENIR

L'analogie corrigée : l'ajustement fin apprend à l'étudiant une *manière de travailler et de répondre*. Le RAG lui met sous les yeux *les documents dont il a besoin maintenant*.

Les deux se combinent naturellement : un modèle adapté au vocabulaire et aux usages d'un métier, qui consulte une documentation externe tenue à jour. C'est aujourd'hui la configuration la plus courante en entreprise.

EN PRATIQUE

Un échec de RAG est presque toujours un échec de *recherche*, pas de génération. Si les bons extraits ne remontent pas, aucun modèle ne rattrapera le coup : il produira une réponse plausible à partir de ce qu'on lui a donné. Avant d'incriminer le modèle, il faut donc toujours regarder ce qui a effectivement été placé dans son contexte. C'est le premier réflexe de diagnostic, et il résout une large part des problèmes constatés.

Du modèle au système

CE QUE VOUS UTILISEZ N'EST PLUS UN MODÈLE

Le LLM est devenu un composant. Ce que vous interrogez est une chaîne de traitement dont il n'est qu'un maillon.

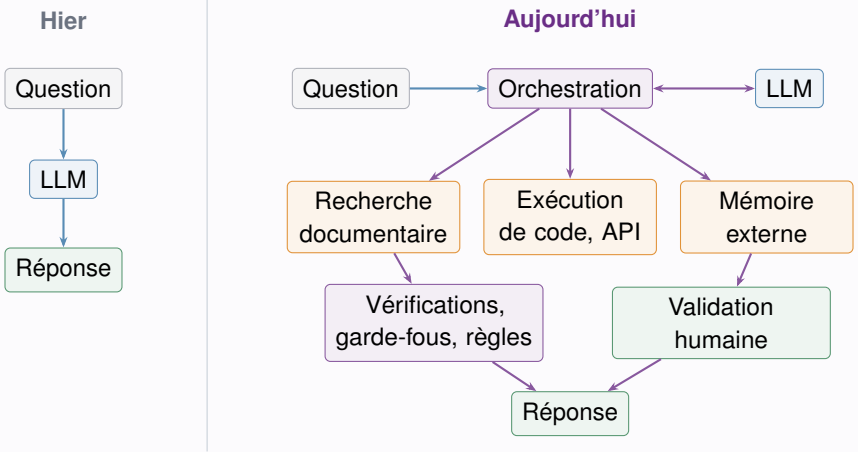


Figure 14. L'IA que vous utilisez au quotidien est un système, pas un modèle isolé. Le modèle produit toujours des tokens ; tout le reste est de l'ingénierie autour.

Cette évolution est la clé de lecture des progrès récents. Beaucoup de ce que l'on attribue à « un modèle plus intelligent » vient en réalité de l'organisation qui l'entoure : de meilleurs outils, de meilleures recherches, de meilleures boucles de vérification, de meilleurs points de reprise humaine. Le modèle compte. Le système compte au moins autant.

SOUS LE CAPOT – lecture avancée facultative

Ce que le passage au système change vraiment.

- **L'actualité** devient possible : le contexte est alimenté à la demande, sans réentraîner quoi que ce soit.

- **La traçabilité** devient possible : on peut journaliser ce qui a été recherché, trouvé, transmis au modèle, et produit.
- **L'abstention** devient possible : le système peut décider que les extraits sont insuffisants et déclencher autre chose qu'une réponse.
- **Le contrôle** devient possible : les règles métier, les seuils, les validations obligatoires vivent dans l'orchestration, pas dans les paramètres du modèle.

Aucune de ces quatre propriétés n'est atteignable en agissant sur le seul modèle. Toutes sont des propriétés du système.

L'IDÉE À RETENIR

Conséquence pratique : face à un projet d'IA, la bonne première question n'est pas « quel modèle choisir ? », mais « quelle tâche, avec quelles sources, quels contrôles et quels points de reprise humaine ? ». Le choix du modèle vient ensuite, et il est souvent moins décisif qu'on ne le croit.

Conclusion : prédire avant de prétendre comprendre

CE QU'IL FAUT RETENIR

Lorsque vous discutez avec une IA, vous n'assistez pas à la recherche d'une réponse dans une gigantesque base de connaissances.

Vous observez un système qui :
- découpe votre question en fragments et les convertit en nombres ;
- met en relation tous les éléments du contexte, en pondérant ce qui compte pour chacun ;
- calcule à chaque pas une distribution de probabilités sur le token suivant ;
- en choisit un, l'ajoute au contexte, et recommence intégralement ;
- et, de plus en plus souvent, s'interrompt pour chercher, calculer, vérifier ou demander.

Cette mécanique paraît trop simple pour produire ce qu'elle produit. C'est sans doute le fait le plus intéressant de toute l'histoire récente de l'intelligence artificielle : un objectif d'apprentissage d'une banalité absolue, deviner la suite, a fait émerger des capacités que personne n'avait spécifiées.

Il faut alors éviter deux erreurs symétriques, aussi répandues l'une que l'autre. La première consiste à mystifier ces systèmes, à leur prêter une compréhension, des intentions, une conscience, ce que rien n'établit. La seconde consiste à les réduire à de simples « perroquets probabilistes », formule commode qui ne rend pas justice aux capacités effectivement observées, ni à ce qu'elles permettent de construire.

Le mécanisme est une prédiction. Ce qu'il produit dépend de ce qu'on place autour.

Les progrès qui viennent ne se joueront pas seulement sur la taille des modèles. Ils se joueront sur notre capacité à faire collaborer efficacement des modèles, des outils, des connaissances, des processus, et des humains.

EN PRATIQUE

Cinq réflexes issus de cet article.

1. **La fluidité ne prouve rien.** Une réponse bien écrite n'est pas une réponse vérifiée. Les deux

qualités sont indépendantes.
2. **Soignez le contexte plutôt que de l'allonger.** Pertinent vaut mieux que volumineux, toujours.
3. **Sur un point factuel précis, exigez la source.** Sans source consultable, considérez l'information comme une hypothèse.
4. **Demandez la décomposition sur les tâches à étapes.** Puis lisez les étapes en sachant qu'elles sont un brouillon, pas une preuve.
5. **Placez le contrôle selon l'enjeu.** La bonne question n'est jamais « faut-il faire confiance ? », mais « où vérifier, et à quel coût ? ».

Petit glossaire

Token	Fragment de texte élémentaire manipulé par le modèle : mot, morceau de mot, ponctuation, ou espace suivi d'un mot.
Tokenizer	Algorithme qui découpe un texte en tokens selon un vocabulaire appris. Deux modèles peuvent découper la même phrase différemment.
Embedding	Vecteur associé à un token dans le dictionnaire du modèle. Position de départ, identique quel que soit le contexte.
Représentation contextualisée	Ce que devient ce vecteur après traversée des couches : elle dépend de la phrase entière.
Attention	Mécanisme par lequel chaque token pondère les autres éléments du contexte selon leur utilité pour lui.
Transformer	Architecture fondée sur l'attention, sans récurrence ni convolution dans sa formulation d'origine.
Autorégressif	Se dit d'une génération où chaque élément produit est réinjecté dans l'entrée pour produire le suivant.
Fenêtre de contexte	Quantité maximale de tokens pris en entrée en un appel : consignes, historique, documents et réponse en cours cumulés.
Température	Réglage de décodage qui accentue ou aplatit les écarts de probabilité entre candidats.
Top-k / top-p	Stratégies limitant les candidats retenus au tirage, par nombre fixe ou par masse de probabilité cumulée.
Paramètres	Les milliards de nombres ajustés pendant l'entraînement. Ils encodent des régularités, pas des fiches consultables.
Décodage	Étape, extérieure au réseau, qui choisit le token retenu à partir des probabilités calculées.
Préentraînement	Phase initiale : prédire le token suivant sur de très grands corpus.
Ajustement fin	Poursuite de l'entraînement sur des données ciblées, pour adapter le comportement, le style ou la tâche.
Alignement	Ajustement du comportement à partir de préférences humaines, pour des réponses jugées utiles, honnêtes et sûres.
Prompt système	Consignes placées en tête du contexte à chaque appel. Ne modifie aucun paramètre.
RAG	Génération augmentée par recherche : on retrouve des extraits pertinents, on les place dans le contexte, on répond en s'appuyant dessus.
Hallucination	Réponse fluide et plausible, mais fausse. Conséquence du mécanisme, pas d'une panne.

Pour aller plus loin

EN PRATIQUE

Cet article présente les principes essentiels sans équations. Les lecteurs souhaitant comprendre les mécanismes internes des Transformers, reproduire les calculs et construire progressivement ces architectures (attention, blocs complémentaires, encodeurs, décodeurs, génération autorégressive) retrouveront l'ensemble des développements, des illustrations et des laboratoires pratiques dans

> l'ouvrage *Brick-by-Brick Approach to Modern Deep Learning Architectures*, publié chez Springer.

Démonstrateur interactif. Un petit démonstrateur accompagne cet article (`demo_llm_comment_ca_marche.html`, à ouvrir dans un navigateur ; il est téléchargeable sur la page Presse du LAB'IA, voir le QR code de la page 1). En quatre étapes, il fait manipuler la tokenisation d'une phrase, l'attention sur le mot « avocat », la génération token par token avec son réglage de température, et la différence entre une réponse tirée de mémoire et une réponse appuyée sur un document.