Département Génie Industriel

Projet Bibliographique

Modèles de langage

Réalisé par :
M. ELadab Mohamed
M. klila Mohamed

Encadrant :
M. Ben Miled Marouane

1A Techniques Avancées

Année universitaire : 2024/2025

23 mars 2025

Remerciements

Nous tenons à exprimer nos sincères remerciements à toutes les personnes ayant contribué, de près ou de loin, à l'élaboration de ce projet.

Nous remercions tout particulièrement notre encadrant, ***M. Ben Miled Marouane****, pour son écoute, ses précieux conseils et son partage généreux de savoir. Ce projet n'aurait jamais vu le jour sans la formation et les orientations qu'il nous a fournies.*

Nos remerciements s'adressent également à notre responsable, ***Mme Chaker Hedia****, pour son soutien indéfectible, le temps qu'elle nous a consacré et ses encouragements constants.*

Enfin, nous adressons notre gratitude à toutes les personnes qui nous ont conseillés et relus lors de la rédaction de ce projet, notamment les anciens et nouveaux étudiants de la première année Techniques Avancées, avec lesquels nous étudions actuellement.

Table des matières

Table des figures

Liste des sigles et acronymes

LM
Language Model

LLM
Large Language Model

TAL
Traitement Automatique du Langage

NLP
Natural Language Processing

NLU
Natural Language Understanding

GPT
Generative Pre-trained Transformer

RNN
Recurrent Neural Network

LLAMA
Large Language Model Meta AI

OpenAI
Open-source artificial intelligence

CoT
Chain of Thought

RLHF
Reinforcement Learning From Human Feedback

FLAN
Fine-tuned Language Net

PaLM
Pathways Language Model

T5
Text-To-Text Transfer Transformer

Introduction

Au cours de la dernière décennie, le domaine du traitement du langage naturel (TAL, *Natural Language Processing* ; NLP) a connu une transformation sans précédent. Les avancées réalisées en intelligence artificielle (IA), rendues possibles par l'émergence des réseaux de neurones profonds, l'accès à d'énormes volumes de données, ainsi que l'augmentation exponentielle de la puissance de calcul, ont constitué les principaux moteurs de cette évolution.

Dans ce contexte, l'architecture Transformer a été développée, servant de base au développement des grands modèles de langage (LLMs, *Large Language Models*). Entraînés sur des corpus textuels massifs, ces modèles ont démontré des capacités surpassant l'état de l'art pour diverses tâches linguistiques, telles que la génération de texte et l'analyse des sentiments, marquant ainsi une avancée significative dans le domaine du NLP.

L'un des moments les plus marquants de cette évolution a été l'apparition de ChatGPT. Ce chatbot développé par OpenAI a non seulement prouvé sa capacité à répondre aux questions de manière presque indistinguable de celle d'un humain, mais il a également montré une aptitude à comprendre le contexte de la conversation et à interagir de manière cohérente avec l'utilisateur. Cette percée a ouvert la voie à une nouvelle ère du NLP, avec l'apparition de nombreux autres modèles de grande taille, tels que LaMDA et PaLM de Google.

Cette accélération fulgurante, illustrée par la multiplication rapide des LLMs toujours plus performants en l'espace de quelques mois seulement, témoigne de la rapidité avec laquelle ce domaine progresse.

1 Modèles de langage

1.1 Introduction

Les modèles de langage (LMs) occupent une place centrale dans le domaine du traitement automatique du langage naturel (TAL ou NLP). En effet, ils confèrent aux systèmes informatiques des capacités avancées en compréhension (Natural Language Understanding, NLU) et en génération de texte (Natural Language Generation, NLG) en s'appuyant sur de vastes ensembles de données textuelles.

Ce chapitre vise à introduire les modèles de langage en mettant en avant leur rôle probabiliste dans l'évaluation de la cohérence des séquences de mots. Nous commencerons par une définition formelle et une présentation de leurs principales applications en traitement du langage naturel, telles que la génération de texte et la traduction automatique.

Nous retracerons ensuite l'évolution des approches de modélisation, depuis les modèles statistiques basés sur les n-grammes jusqu'aux modèles neuronaux avancés.Améliorant ainsi la capacité des modèles à capturer les relations contextuelles au sein du langage.

1.2 Définition des modèles de langage (LM)

Un LM est un modèle probabiliste capable, à partir de données textuelles dans une ou plusieurs langues, de calculer la probabilité d'une séquence de mots dans un langage naturel. Dans le même ordre d'idées un modèle attribue une probabilité à un extrait de texte qu'il rencontre pour la première fois, en se basant sur les données d'apprentissage. Par exemple, un LM entraîné sur les archives d'un grand journal anglais devrait attribuer une probabilité plus élevée à la séquence « a bit of text » qu'à « aw pit tov tags » puisque les mots de la première phrase sont beaucoup plus fréquents dans le corpus d'entraînement [12] .

En d'autres termes, un LM est conçu pour capturer la distribution de probabilité des éléments (mots) dans une ou plusieurs langues à partir d'un ensemble de données, ce qui lui confère la capacité de prédire la probabilité associée à différentes suites de mots. Ces modèles sont utilisés dans de nombreuses applications, telles que [8] :

- **Génération du langage naturel** : Ils permettent de générer des textes qui imitent étroitement le langage humain
- **Traduction automatique** : Plutôt que de traduire littéralement, par exemple, la phrase anglaise « I am feeling blue » en « Je me sens bleu », un LM oriente la traduction vers « Je me sens déprimé » en se basant sur la fréquence et la pertinence des expressions dans la langue cible
- **Reconnaissance vocale** : Ils aident à choisir le mot le plus approprié indépendamment des variations acoustiques. Ainsi, un système de reconnaissance vocale peut distinguer entre des phrases semblables comme « Je veux voir des films de science-friction » et « films de science-fiction »
- **Recherche d'information** : Ils jouent un rôle clé dans l'interprétation précise des requêtes des utilisateurs, améliorant ainsi la pertinence des résultats
- **Systèmes conversationnels** : Ces modèles permettent de rendre les interactions avec les assistants virtuels et chatbots plus naturelles et intuitives

1.3 Évolution des modèles de langage

Les modèles de langage jouent un rôle central dans le traitement automatique du langage naturel en modélisant la distribution des mots et des phrases dans une langue donnée. Après avoir défini ces modèles, il est essentiel d'examiner leur évolution au fil du temps.

Dans cette section, nous retraçons brièvement l'évolution des modèles de langage, en commençant par les approches fondamentales comme les modèles n-grammes [22]. Par la suite, les progrès en puissance de calcul et en volume de données disponibles ont favorisé l'essor des modèles neuronaux, offrant des performances accrues grâce à des architectures plus complexes [3].

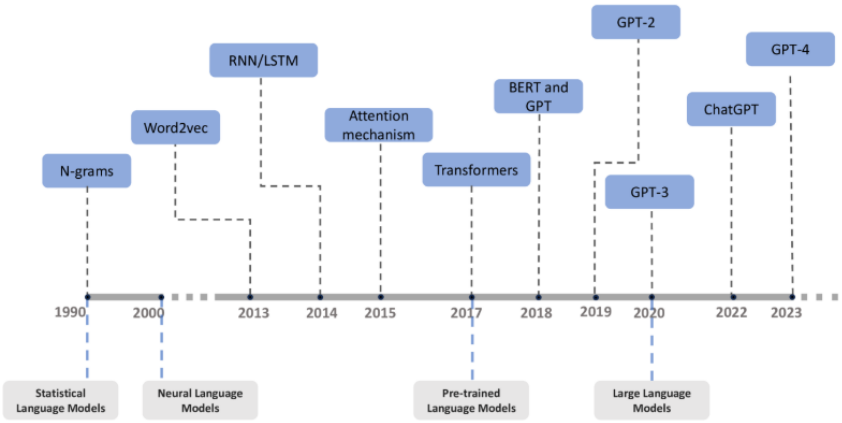

Figure 1 – Évolution des architectures utilisées pour la modélisation du langage au fil du temps [6].

1.3.1 Techniques statistiques pour la modélisation du langage

Au début du 20^{e} siècle, le mathématicien Andreï Markov a introduit les chaînes de Markov, un modèle mathématique permettant de représenter des systèmes suivant une logique probabiliste. Ces chaînes, qui portent son nom, ont profondément influencé la théorie des probabilités. Bien que Markov n'ait pas étudié les n-grammes, ses travaux sur la modélisation des séquences ont inspiré des approches probabilistes utilisées plus tard dans l'analyse du langage.

Plus tard, au milieu du 20^{e} siècle, Claude Shannon a développé les idées de Markov . Son approche visait à modéliser les probabilités d'occurrence de suites de caractères dans les textes en utilisant les chaînes de Markov. Pour cela, il a introduit la notion de n-grammes, qui représentent les séquences de n caractères successifs. En estimant les fréquences d'apparition de ces n-grammes dans un large corpus, Shannon a créé un modèle statistique capable de prédire les probabilités de différentes séquences de caractères. En d'autres termes, il s'agit de déterminer la probabilité qu'un certain n-gramme apparaisse, étant donné les n-grammes précédents [23]. La

formule de probabilité des n-grammes est donnée par l'équation suivante :

$$P(w_n \mid w_{n-1}, w_{n-2}, \ldots, w_1) = \frac{C(w_n, w_{n-1}, \ldots, w_1)}{C(w_{n-1}, \ldots, w_1)}$$

où $P(w_n \mid w_{n-1}, w_{n-2}, \ldots, w_1)$ est la probabilité conditionnelle du mot w_n étant donné les mots précédents, $C(w_n, w_{n-1}, \ldots, w_1)$ est le nombre de fois que la séquence de mots apparaît dans le corpus, et $C(w_{n-1}, \ldots, w_1)$ est le nombre de fois que la séquence de mots précédents apparaît dans le corpus.

Exemple d'analyse du modèle de génération du texte ou des séquences de signes en langue des signes taïwanaise basé sur les n-grammes

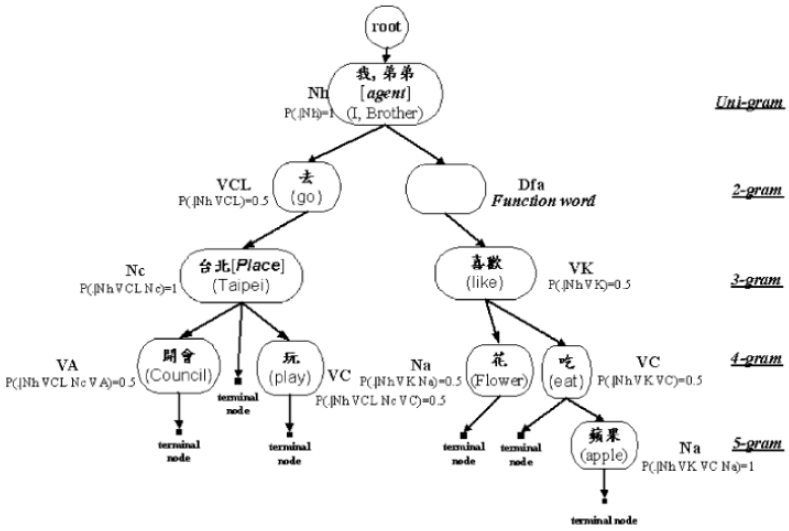

Figure 2 – Modèle probabiliste de génération de texte basé sur les n-grammes [32].

L'image utilise plusieurs abréviations pour catégoriser les mots dans la phrase. Voici leurs significations :

- **Nh (Agent) :** Désigne le sujet de la phrase, souvent une personne ou un pronom personnel (我 – *I*).
- **VCL (Verbal Classifier) :** Verbe indiquant une action en lien avec un lieu ou un déplacement (去 – *go*).
- **Nc (Noun - Place) :** Nom propre ou lieu, ici 台北 (*Taipei*).
- **VA (Adjective/Verb Adjective) :** Adjectif ou verbe d'état (開會 – *Council*).
- **VK (Preference Verb) :** Verbe d'appréciation ou de préférence (喜歡 – *like*).
- **VC (Transitive Verb) :** Verbe transitif nécessitant un complément d'objet (吃 – *eat*).
- **Na (Noun) :** Nom commun représentant un objet concret (花 – *Flower*, 蘋果 – *Apple*).

L'arbre illustré dans l'image représente une phrase construite progressivement selon un modèle probabiliste basé sur les *n-grammes*. Chaque niveau de l'arbre correspond à une structure syntaxique et sémantique des éléments du langage. Le modèle suit une approche hiérarchique où chaque niveau représente un n-gram spécifique.

— **Uni-gram (1-gram) :** Un seul mot, comme 我, 弟弟 (*I, Brother*), représentant l'agent de l'action (**Nh**).
— **Bi-gram (2-gram) :** Ajout d'un verbe d'action comme 去 (*go*) classé en **VCL** (verbal classifier).
— **Tri-gram (3-gram) :** Introduction d'un mot fonctionnel (**Dfa**) et d'un verbe de préférence, par exemple 喜歡 (*like*), classé en **VK**.
— **4-gram :** Inclusion d'un objet nominal (**Na**), comme 花 (*Flower*), et d'un verbe transitif 吃 (*eat*) classé en **VC**.
— **5-gram :** Ajout du dernier élément syntaxique, ici 蘋果 (*apple*), un autre nom classé en **Na**.

Ainsi l'exemple illustre comment un modèle basé sur les **n-grammes** permet de construire une phrase de manière progressive en suivant une approche probabiliste. En commençant par un mot unique (**uni-gram**), chaque ajout successif repose sur une **probabilité conditionnelle** qui détermine la plausibilité d'apparition d'un mot ou d'une catégorie grammaticale en fonction du contexte précédent. Cette approche hiérarchique permet d'assurer la cohérence syntaxique et sémantique de la phrase, reflétant ainsi la structure du langage. En NLP (Traitement Automatique du Langage), cette méthode est utilisée pour **analyser, générer et prédire** du texte, car elle capture les relations entre les mots en s'appuyant sur des séquences linguistiques observées dans un corpus. Ainsi, l'exemple met en évidence le principe fondamental selon lequel la probabilité d'un mot dépend de son **historique contextuel**, ce qui constitue la base de nombreuses applications, comme la traduction automatique, la correction orthographique ou encore la reconnaissance vocale [32].

1.3.2 Modèles de langage basés apprentissage profond

Word2Vec

Word2Vec est un modèle d'apprentissage automatique dédié à la représentation vectorielle des mots en traitement du langage naturel. Il transforme les mots en vecteurs de réels de dimension fixe, permettant ainsi de capter les relations syntaxiques et sémantiques qui existent entre eux. Le modèle fonctionne en prédisant les mots du contexte à partir d'un mot central, dans le but de rapprocher dans l'espace vectoriel des mots aux significations similaires. Cette approche facilite des tâches telles que la mesure de similarité, la résolution d'analogies (par exemple, « *roi* » − « *homme* » + « *femme* » ≈ « *reine* ») ainsi que l'amélioration des performances dans des applications comme la classification de textes et la traduction automatique [17].

Cependant, *Word2Vec* présente certaines limites, notamment l'absence de gestion de la polysémie. En effet, chaque mot est considéré comme une entité unique, ce qui implique que les différents sens d'un même mot (par exemple, « *banc* » pouvant désigner un établissement financier ou un siège) se voient attribuer un unique vecteur, indépendamment de leur contexte.

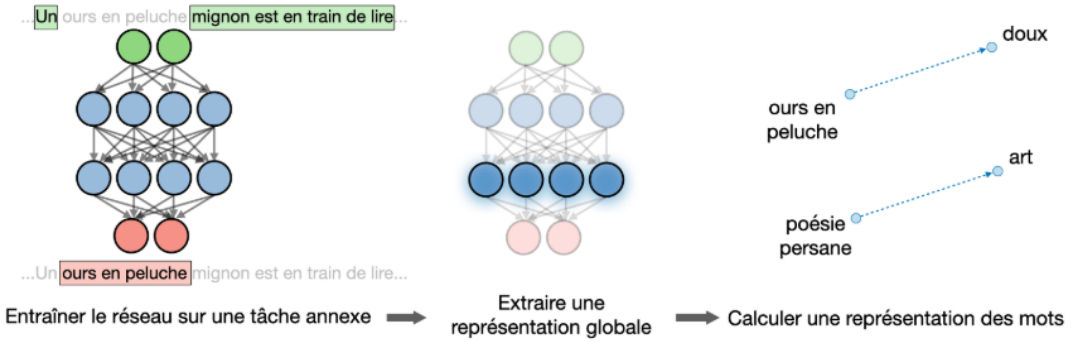

Figure 3 – *Word2Vec* représente les mots en estimant la probabilité qu'un mot donné apparaisse dans le contexte des mots qui l'entourent [1].

Réseaux de Neurones Récurrents

Les réseaux de neurones *Word2Vec* ne sont pas adaptés aux problèmes séquentiels, car ils ne possèdent pas de mécanisme de mémoire et ne tiennent pas compte du contexte, qu'il soit temporel ou spatial. De plus, ces réseaux fonctionnent sur des données de taille fixe, ce qui n'est pas compatible avec des données séquentielles dont la longueur varie.

Pour traiter des problèmes séquentiels, il est nécessaire d'utiliser des modèles dynamiques qui prennent en compte les états précédents d'une séquence, comme les réseaux de neurones récurrents (RNN). Ces derniers ont la particularité de recevoir non pas un vecteur unique en entrée, mais une séquence de vecteurs. Cette séquence est souvent obtenue à l'aide d'une fenêtre glissante qui parcourt le signal à analyser. Les éléments de cette séquence sont appelés des "trames".

Un RNN est constitué de couches contenant des neurones, également appelés cellules [24].

La figure 4 met en évidence la distinction entre un RNN et un *Word2Vec*. Dans un RNN, les neurones récurrents possèdent des poids supplémentaires (en rouge) qui relient leurs sorties à leurs entrées, ce qui permet de modéliser la mémoire nécessaire au traitement des séquences. Ainsi, la sortie d'une couche récurrente à un instant donné dépend de sa sortie précédente. Elle reçoit en entrée la sortie de la couche précédente au temps t et sa propre sortie au temps $t-1$.

$$a_{j,h}^{t} = \sum_{i=1}^{I} w_{i,j,h} x_i^t + \sum_{j'=1}^{J} w_{j',j,h} b_{j',h}^{t-1}$$

L'équation d'un neurone récurrent est définie comme suit :
Les poids de récurrence $w_{j',j,h}$ ajoutent un terme (en rouge).
Les entrées sont formulées sous la forme :

$$X = (x_0, x_1, ..., x_i, ..., x_I)$$

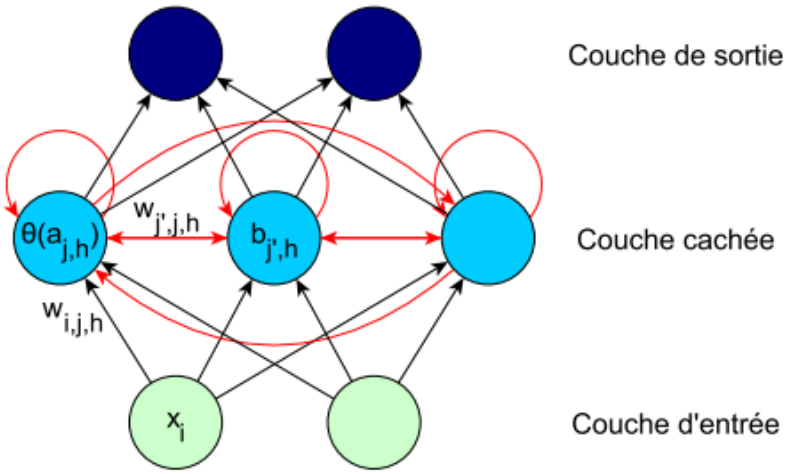

Figure 4 – Réseau de neurones récurrents [24].

tandis que les sorties des neurones récurrents comportent un exposant t, indiquant leur position temporelle dans la séquence :

$$(X_0, ..., X_t, ..., X_T).$$

Dans l'équation, $b_{j',h}^{t-1}$ représente la valeur du neurone j' calculée à l'itération précédente pour les entrées x_i^{t-1} de l'élément précédent de la séquence.

Lors du calcul de la passe en avant pour le premier élément d'une séquence X_0, aucune valeur n'est encore stockée en mémoire. Il est donc nécessaire d'initialiser les sorties des neurones récurrents à zéro [24].

2 Grand modèles de langage

2.1 Introduction

Dans le chapitre précédent, nous avons introduit les concepts fondamentaux des modèles de langage, en définissant leur rôle et leurs applications dans diverses tâches de traitement automatique du langage naturel (NLP). Nous avons également retracé l'évolution historique des approches de modélisation, depuis les modèles statistiques simples basés sur les n-grammes jusqu'aux approches plus avancées telles que Word2Vec et les réseaux de neurones récurrents (RNNs).

Dans ce chapitre, nous nous intéressons aux **grands modèles de langage** (LLMs), qui comptent plusieurs centaines de milliards de paramètres. L'émergence de ces modèles, tels que GPT-3, PaLM et LLaMA, a marqué une avancée majeure dans le domaine du NLP. Ces modèles se distinguent par leurs capacités exceptionnelles en compréhension et génération du langage, favorisant ainsi des progrès significatifs dans de nombreuses tâches linguistiques.

Nous explorerons le domaine des LLMs en détaillant les principales étapes de leur cycle de développement. Nous débuterons par une analyse des choix d'architecture, des sources de données et des techniques d'entraînement utilisées lors de

la phase de pré-entraînement, qui confère aux modèles des compétences fondamentales. Ensuite, nous examinerons les méthodes d'adaptation permettant d'affiner ces modèles pour des tâches spécifiques. Nous mettrons également en évidence certaines propriétés émergentes des LLMs, notamment les lois de mise à l'échelle et leurs capacités surprenantes. Enfin, nous discuterons des limites et défis que ces modèles rencontrent.

2.2 Pré-entraînement des LLMs

Le pré-entraînement est une étape fondamentale dans le développement des LLMs, car il leur confère des compétences essentielles en compréhension et génération du langage naturel. Cette phase repose sur trois éléments clés :

— La conception des architectures de modèles.
— La sélection des données d'entraînement.
— Le choix des tâches associées.

Cette sous-section explore en détail chacun de ces aspects.

2.2.1 Architectures

Les LLMs basés sur un décodeur se répartissent en trois grandes catégories :

— Encodeur-décodeur.
— Décodeur causal.
— Décodeur de préfixe.

Chaque architecture se distingue par un schéma d'attention spécifique.

Architecture encodeur-décodeur

Inspirée du modèle *Vanilla Transformer* [28], l'architecture encodeur-décodeur repose sur deux modules distincts : un encodeur et un décodeur. L'encodeur, constitué de plusieurs couches d'auto-attention multi-têtes empilées, transforme la séquence d'entrée en représentations latentes. Le décodeur exploite ensuite ces représentations grâce à un mécanisme d'attention croisée pour générer la séquence cible. Bien que cette approche soit efficace pour diverses tâches de traitement du langage naturel, elle est relativement peu adoptée par les LLMs, à l'exception de certains modèles comme Flan-T5 [7].

Architecture du décodeur causal

L'architecture du décodeur causal repose sur un masque d'attention unidirectionnel, garantissant que chaque jeton ne peut interagir qu'avec les jetons précédents et lui-même. Dans cette configuration, les jetons d'entrée et de sortie sont traités au sein du même décodeur. Des modèles emblématiques tels que *GPT-1*, *GPT-2* et *GPT-3* reposent sur cette architecture, *GPT-3* se distinguant par ses impressionnantes capacités d'apprentissage en contexte. De nombreux LLMs, comme *OPT*, *BLOOM* et *Gopher*, ont largement adopté cette approche [27].
Ainsi, bien que l'architecture du décodeur causal ait démontré son efficacité pour la

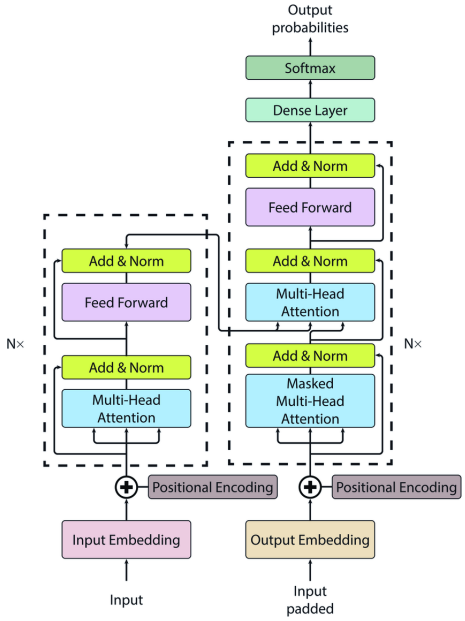

Figure 5 – Une architecture du *Vanilla Transformer*. [28]

modélisation du langage, certaines tâches nécessitent une approche plus fine permettant de capturer les relations causales entre les variables. C'est dans cette perspective que l'architecture du modèle causal structurel est introduite, détaillée dans la Figure 6.

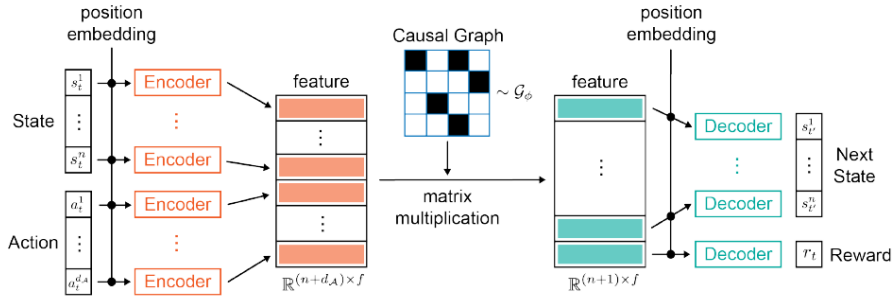

Figure 6 – Architecture du décodeur causal structurel. [10]

Dans les réseaux neuronaux classiques, l'entrée est traitée comme un tout et traverse des couches linéaires ou des couches de convolution. Cependant, cette approche mélange toutes les informations contenues dans l'entrée, rendant le **graphe causal** inefficace pour distinguer cause et effet. Pour remédier à cela, notre modèle intègre un **encodeur** partagé entre toutes les dimensions de l'entrée.

Étant donné que différentes dimensions peuvent contenir exactement les mêmes valeurs, nous ajoutons un **encodage de position apprenable** à l'entrée de l'encodeur. Ainsi, la dimension d'entrée de l'encodeur est $1 + d_{\text{pos}}$, où d_{pos} représente la dimension de l'encodage de position.

Après l'encodeur, nous obtenons un ensemble de caractéristiques indépendantes pour chaque dimension de l'entrée. Ensuite, nous multiplions ces caractéristiques par un **graphe causal binaire apprenable** G.

« *The element* (i, j) *of the graph is sampled from a* ***Gumbel-Softmax distribution*** *with parameter* $\Phi_{i,j}$ *to ensure the loss function is differentiable w.r.t* Φ » [10].

La multiplication du **graphe causal** par les caractéristiques d'entrée permet de créer une combinaison linéaire des caractéristiques en fonction du graphe causal. Les caractéristiques ainsi obtenues sont ensuite transmises à un **décodeur** chargé de prédire l'**état suivant** et la **récompense**. Comme pour l'encodeur, le décodeur est **partagé** entre toutes les dimensions afin d'éviter toute fuite d'information entre elles. L'**encodage de position** est également inclus dans l'entrée du décodeur, et la dimension de sortie du décodeur est 1 [10].

Architecture du décodeur de préfixe

Également appelée décodeur non causal, l'architecture du décodeur de préfixe ajuste le mécanisme de masquage des décodeurs causals afin d'autoriser une attention bidirectionnelle sur les jetons de préfixe, tout en maintenant une attention unidirectionnelle sur les jetons générés. De la même manière que l'architecture encodeur-décodeur, elle permet de représenter le préfixe de manière bidirectionnelle et de générer les jetons de sortie de façon autorégressive en exploitant des paramètres partagés.
Parmi les LLMs adoptant cette approche, on retrouve notamment *GLM130B* et *U-PaLM* [27].

2.2.2 Données

Les données représentent un élément essentiel de tout modèle d'apprentissage automatique. Leur rôle est d'autant plus crucial pour les Grands Modèles de Langage, dont les performances dépendent directement de la qualité des corpus de pré-entraînement et des méthodes de prétraitement appliquées. Dans cette sous-section, nous commencerons par examiner la notion de qualité des données dans le contexte des LLMs, avant d'aborder le processus de collecte et de traitement des données utilisées pour leur pré-entraînement.

Qualité des données

La qualité des données est un facteur déterminant dans la performance des LLMs.elle peut être évaluée selon quatre dimensions [29] :

- **Qualité intrinsèque** : garantit la fiabilité des données, en s'assurant de leur exactitude, objectivité et crédibilité afin de limiter les erreurs factuelles et les biais.
- **Qualité contextuelle** : mesure la pertinence des données pour une tâche donnée, en veillant à leur actualité, leur adéquation et leur exhaustivité.
- **Qualité représentationnelle** : concerne la clarté et la lisibilité des données, en minimisant le bruit et les redondances.

— **Qualité d'accessibilité** : assure un accès facile et sécurisé aux données, tout en garantissant leur protection et leur utilisation efficace pour l'entraînement des modèles.

Source de données

Pour développer un modèle de langage performant, il est essentiel de constituer un large corpus de langage naturel provenant de diverses sources. Ces sources se divisent en deux catégories principales : les données textuelles générales, telles que les pages web, qui offrent une vaste diversité de styles et de contenus linguistiques (présentant *GPT-2*, un modèle ayant démontré la puissance des approches d'apprentissage auto-supervisé sur de grandes quantités de texte) [21], les conversations et les livres, qui apportent une richesse contextuelle et une structure narrative (introduisant *The Pile*, un corpus de 800 Go de textes variés conçu pour l'entraînement des modèles de langage) [11] .

D'autre part, les données spécialisées, comme les textes scientifiques, permettent d'affiner les modèles sur des connaissances précises et des formulations rigoureuses. Un modèle de langage entraîné spécifiquement sur des textes académiques pour améliorer la compréhension et la génération scientifique [26]. Le code source informatique constitue également une ressource clé pour enrichir les capacités des modèles, notamment en matière de raisonnement formel et de structuration logique du langage.

Cette diversité garantit une couverture complète du langage naturel. En particulier, l'utilisation de données issues du code a montré des résultats prometteurs dans l'entraînement des modèles de langage de grande taille (LLMs), en renforçant leur capacité à accomplir des tâches de programmation et à générer du code de manière efficace [33].

De plus, le pré-entraînement combiné sur des données textuelles et de code a contribué à améliorer les performances des LLMs sur des tâches de raisonnement avancé, telles que la réponse à des questions scientifiques et le raisonnement logique, renforçant ainsi leur utilité dans des domaines nécessitant une compréhension fine et une capacité d'inférence accrue.

Pré-traitement des données

Une fois une grande quantité de données textuelles collectée, une étape clé consiste à pré-traiter ces données pour créer le corpus de pré-entraînement. Ce processus inclut les étapes suivantes :

1. **Le filtrage de qualité**, qui vise à éliminer les données de mauvaise qualité, telles que les textes incomplets ou contenant des erreurs.
2. **La déduplication**, qui consiste à supprimer les données redondantes.
3. **La rédaction pour la confidentialité** *(Privacy Redaction)*, qui permet de garantir le respect de la vie privée en supprimant les informations sensibles.
4. **La tokenisation**, Elle peut être comparée au processus d'apprentissage de la lecture pour un enfant. Plutôt que de commencer directement par des paragraphes complexes, on commence par lui présenter des lettres, puis des

syllabes, et enfin des mots entiers. De la même manière, la tokenisation décompose de grandes quantités de texte en unités plus petites et plus compréhensibles pour les machines : **Tokens**.
Le principal objectif de la tokenisation est de transformer le texte en une forme que les machines peuvent traiter tout en préservant le contexte. En convertissant le texte en jetons, les algorithmes sont en mesure d'identifier plus facilement des motifs et des relations. Cette capacité de reconnaissance des formes est essentielle, car elle permet aux machines de comprendre et de répondre aux données humaines. Par exemple, lorsqu'une machine rencontre le mot "courir", elle ne le considère pas comme un élément isolé, mais comme un ensemble d'unités qu'elle peut analyser pour en déduire le sens [9].
Cette étape peut se faire à différents niveaux et à l'aide de diverses méthodes :

— **Tokenisation par mots.** Cette approche consiste à diviser le texte en mots individuels. Elle est largement utilisée, notamment pour les langues où les frontières entre les mots sont clairement définies, comme l'anglais.
— **Tokenisation par caractères.** Dans cette méthode, le texte est décomposé en caractères uniques. Elle est particulièrement adaptée aux langues où les frontières entre les mots sont floues ou lorsque des analyses plus fines sont nécessaires, comme dans le cas de la correction orthographique.
— **Tokenisation par sous-mots.** Cette méthode cherche à combiner les avantages de la tokenisation par mots et par caractères. Elle segmente le texte en unités plus petites qu'un mot entier, mais plus grandes qu'un simple caractère. Par exemple, le mot "Chatbots" pourrait être découpé en "Chat" et "bots". Cette approche est particulièrement utile pour les langues où le sens dépend de la combinaison de petites unités ou pour traiter les mots inconnus.

Voici un tableau expliquant les différences :

Type	**Description**	**Cas d'utilisation**
Tokenisation des mots	Décompose le texte en mots individuels.	Particulièrement adapté aux langues où les frontières entre les mots sont nettes, telles que l'anglais.
Tokenisation des caractères	Décompose le texte en caractères individuels.	Pratique pour les langues où les frontières entre les mots sont floues, ou pour les tâches demandant une analyse plus fine.
Tokenisation des sous-mots	Décompose le texte en unités plus grandes que des caractères mais plus petites que des mots.	Utile pour les langues à morphologie complexe ou pour traiter les mots hors-vocabulaire.

Table 1 – Différence entre les types de tokenisation et leurs utilisations [9]

2.2.3 Entraînement

Les modèles de langage de grande taille (LLMs) sont généralement pré-entraînés en utilisant trois approches principales : la modélisation auto-régressive du langage, la modélisation masquée du langage et l'auto-encodage avec suppression du bruit. Chaque approche implique une tâche spécifique visant à ajuster les paramètres du modèle de manière optimale en fonction des objectifs de pré-entraînement. Dans cette sous-section, nous présentons chacune de ces tâches et détaillons les principes mathématiques sous-jacents, ainsi que les bonnes pratiques pour leur implémentation.

Tâches de pré-entraînement

Les modèles de langage de grande taille (LLMs) sont généralement pré-entraînés en utilisant trois principales approches : la modélisation auto-régressive du langage, la modélisation masquée du langage et l'auto-encodage avec suppression du bruit. Dans ce qui suit, $P(a \mid b)$ désigne la probabilité conditionnelle de a sachant b.

—**La modélisation auto-régressive du langage** est l'un des objectifs de pré-entraînement les plus couramment utilisés pour les modèles (decoder-only), tels que *GPT-3*. Étant donnée une séquence de tokens, cette tâche vise à prédire de manière auto-régressive le prochain token (ou parfois la prochaine séquence de tokens) en se basant sur les tokens précédents, comme décrit dans l'équation suivante.

$$L_{LM}(x) = \sum_{i=1}^{n} \log P(x_i \mid x_{<i}) \tag{1}$$

où $x = (x_1, x_2, \ldots, x_n)$ représente la séquence de tokens, et $x_{<i}$ désigne la sous-séquence de tokens précédant x_i [4].

— La **modélisation masquée du langage** (Masked Language Modeling, MLM) consiste à masquer certains tokens dans une séquence, puis à entraîner le modèle à prédire ces tokens masqués en se basant sur leur contexte, comme défini par l'équation suivant.

$$L_{MLM}(x) = \sum_{i=1}^{N} \log P(\tilde{x}_i \mid x_{\setminus \tilde{x}_i}) \tag{2}$$

où $x = (x_1, x_2, \ldots, x_N)$ est la séquence d'origine, $\tilde{x}_i$ représente le i-ème token masqué, et $x_{\setminus \tilde{x}_i}$ désigne la séquence sans ce token masqué [4].

— **L'auto-encodage avec suppression du bruit** (Denoising Auto-Encoding, DAE) est un objectif de pré-entraînement moins fréquemment utilisé en raison de sa complexité d'implémentation. Cette complexité provient principalement de la génération et de la gestion des entrées, qui consistent en des textes corrompus où certains segments ont été remplacés aléatoirement. L'objectif du modèle est alors de reconstruire ces segments corrompus, comme décrit dans l'équation suivante.

$$L_{DAE}(x) = \log P(\tilde{x} \mid x_{\setminus \tilde{x}}) \tag{3}$$

où $\tilde{x}$ représente la séquence corrompue et $x_{\setminus \tilde{x}}$ désigne la séquence sans les segments corrompus [15].

2.3 Adaptation des LLMs

Durant la phase de pré-entraînement, les grands modèles de langage (LLMs) acquièrent des compétences générales qui leur permettent de gérer une variété de tâches linguistiques. Cependant, plusieurs recherches ont étudié l'adaptation de ces compétences à des besoins particuliers.

Dans cette section, nous présentons les deux principaux types d'adaptation explorés dans la littérature. **l'adaptation par instructions** (instruction tuning) et **l'adaptation par alignement** (alignment tuning). Le premier type se concentre principalement sur l'amélioration ou la mise en évidence des capacités des LLMs, tandis que le second cherche à ajuster leur comportement afin qu'il soit en accord avec les valeurs et préférences humaines.

Définitions clés

- **Fine-tuning** : Le *fine-tuning* est un processus permettant d'adapter un modèle de langage pré-entraîné à des tâches spécifiques. Il consiste à entraîner un modèle sur un jeu de données restreint et ciblé afin d'ajuster partiellement ou totalement les poids du modèle, dans le but d'améliorer ses performances sur une tâche donnée [31].
- **Adaptation basée sur les instructions** : L'adaptation basée sur les instructions est une approche spécifique du *fine-tuning*. Elle consiste à incorporer des instructions explicites dans les données d'entraînement pour améliorer la compréhension et l'exécution des tâches par les modèles de langage. Contrairement au *fine-tuning* classique, qui ajuste un modèle pré-entraîné sur des données spécifiques, cette approche guide l'apprentissage du modèle en structurant les entrées sous forme d'instructions détaillées [34].
- **Ensemble de données d'instructions** : Un ensemble de données d'instructions est constitué de plusieurs exemples d'apprentissage, chacun structuré selon un format spécifique. Ce format comprend généralement une *instruction* (description de la tâche), une *entrée* (les données nécessaires pour accomplir la tâche), ainsi qu'une *sortie attendue* (résultat attendu). Des *exemples supplémentaires* peuvent être fournis pour améliorer la compréhension du modèle [34].

2.3.1 Adaptation basée sur les instructions

L'adaptation basée sur les instructions est une approche qui guide l'apprentissage du modèle en structurant les exemples d'entraînement sous forme d'instructions détaillées. Contrairement au *fine-tuning* classique, qui consiste uniquement à ajuster un modèle pré-entraîné avec un ensemble de données spécifique, l'adaptation par instructions permet une meilleure généralisation et une meilleure capacité à exécuter des tâches variées.

Rôle des ensembles de données d'instructions

La qualité et la diversité des exemples dans les ensembles de données d'instructions jouent un rôle crucial dans la performance du modèle. Un bon formatage des

données et une conception rigoureuse des ensembles d'instructions permettent de mieux guider le modèle pour des tâches spécifiques. Plusieurs approches ont été proposées pour la construction de ces ensembles [34].

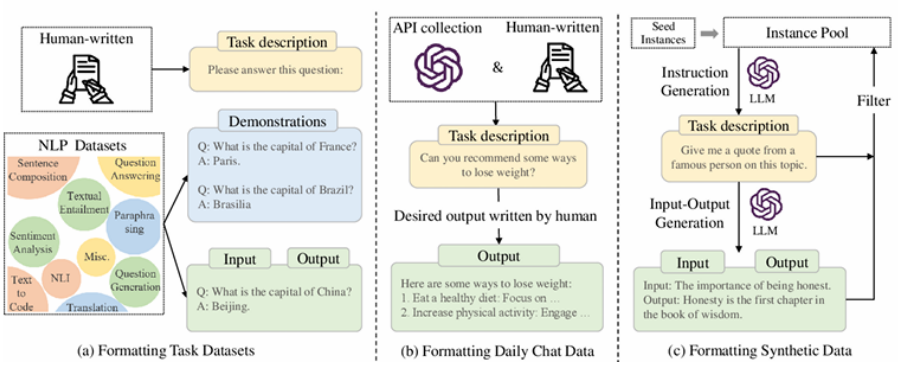

Figure 7 – Constitution des ensembles de donnée d'instructions [34].

Avantages de l'adaptation basée sur les instructions

Des recherches récentes [31] ont montré que cette approche améliore la généralisation des modèles de langage à de nouvelles tâches, réduisant ainsi leur dépendance aux exemples d'entraînement spécifiques. Elle s'est révélée particulièrement efficace avec des modèles de grande échelle tels que *FLAN-T5* et *InstructGPT*, où elle permet d'aligner les comportements des modèles avec les attentes des utilisateurs .

Exemple d'utilisation dans la réponse aux questions

Dans le cadre de la réponse à des questions, FLAN est capable de répondre dans une autre langue lorsqu'on lui fournit l'instruction appropriée, illustrant ainsi la flexibilité et l'efficacité de l'adaptation par instructions [34].

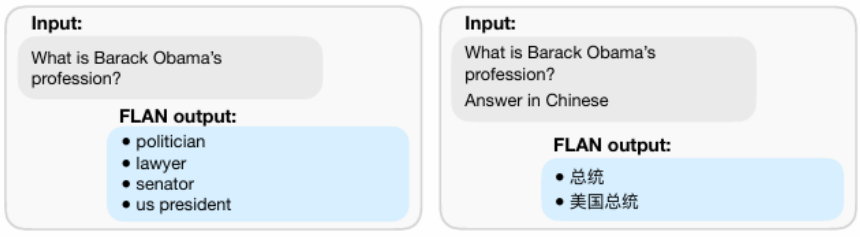

Figure 8 – Réponses de FLAN [34].

2.3.2 Adaptation par alignement

L'adaptation par alignement vise à limiter les risques associés à la génération de contenu nuisible, biaisé ou trompeur par les modèles de langage de grande taille (LLMs). Ces modèles peuvent produire plusieurs réponses possibles à une même requête sans nécessairement comprendre l'intention exacte de l'utilisateur. L'enjeu est

alors de sélectionner la réponse la plus appropriée, c'est-à-dire celle qui correspond le mieux aux attentes humaines.

Pour répondre à ce défi, plusieurs critères d'alignement ont été proposés afin d'évaluer la conformité des sorties générées. De plus, une approche largement adoptée dans ce contexte est l'apprentissage par renforcement à partir de retours humains (Reinforcement Learning from Human Feedback, RLHF), qui permet d'optimiser le comportement des modèles en fonction des préférences humaines. Dans cette section, nous détaillons ces critères ainsi que l'impact de cette approche sur les performances des LLMs.

Critères d'alignement

Les critères d'alignement sont des mesures qualitatives utilisées pour évaluer dans quelle mesure les réponses générées par un LLM respectent les attentes des utilisateurs. Parmi les critères les plus couramment adoptés dans la littérature [2, 19], on retrouve :

- **Utilité** : La réponse doit être pertinente et apporter une information utile en lien avec la requête initiale.
- **Intégrité** : Le contenu généré doit être factuellement correct et ne pas contenir de désinformation.
- **Innocuité** : La réponse ne doit pas contenir de contenu nuisible, biaisé ou offensant.

Reinforcement Learning from Human Feedback (RLHF)

L'apprentissage par renforcement à partir de retours humains (*Reinforcement Learning from Human Feedback*, RLHF) joue un rôle essentiel dans l'entraînement des modèles et des applications d'intelligence artificielle afin de produire des réponses plus naturelles et conformes aux attentes humaines.

Le processus RLHF se déroule en plusieurs étapes :

1. Le modèle génère une réponse qui tente d'imiter une formulation humaine.
2. Un superviseur humain évalue cette réponse en fonction de plusieurs critères liés aux valeurs humaines, tels que la convivialité, l'humeur, les sentiments associés au texte ou à l'image.
3. Une note est attribuée à la réponse, permettant ainsi au modèle d'ajuster ses futures prédictions en fonction du retour humain.

Par exemple, un modèle de traduction peut produire un texte techniquement correct, mais dont le style semble peu naturel à un lecteur natif. Grâce au retour humain, ces subtilités peuvent être détectées et prises en compte pour améliorer la qualité des futures générations de texte. Ainsi, le modèle affine progressivement ses réponses afin qu'elles correspondent mieux aux attentes et préférences des utilisateurs [13].

La formation d'un modèle de récompense repose sur plusieurs étapes essentielles visant à améliorer l'alignement des modèles de langage avec les attentes humaines[13]. Ces étapes incluent :

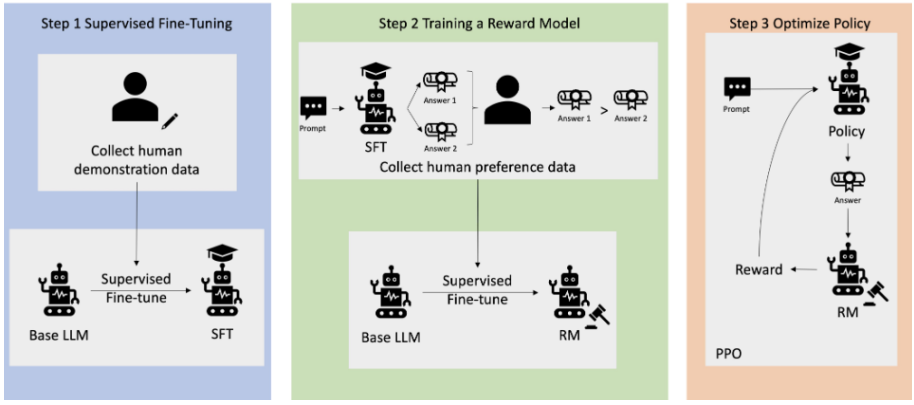

Figure 9 – un aperçu du processus d'apprentissage RLHF [13].

— **Définition du modèle de récompense** : Le modèle de récompense peut être conçu comme un système modulaire distinct ou intégré directement dans un modèle de langage de bout en bout.
— **Entraînement du modèle de récompense** : Cette phase repose sur un ensemble de données distinct de celui utilisé lors du pré-entraînement du modèle de langage. Cet ensemble est constitué de paires *invite-récompense*, où chaque invite représente une sortie attendue et les récompenses associées évaluent la pertinence et la qualité de la réponse.
— **Association des invites aux récompenses** : L'entraînement du modèle consiste à l'exposer à des invites (ou *prompts*) et à associer les sorties générées aux valeurs de récompense correspondantes, facilitant ainsi l'apprentissage des critères d'évaluation implicites.
— **Intégration du retour humain** : L'affinement du modèle s'effectue grâce aux retours humains, qui permettent d'ajuster les évaluations des réponses. Par exemple, *ChatGPT* exploite les retours des utilisateurs en leur demandant d'évaluer les réponses à l'aide d'indicateurs tels que le *pouce vers le haut* ou *vers le bas*.

2.4 Propriétés des LLMs

Les LLMs ont montré des performances exceptionnelles dans de nombreuses tâches. Ces performances résultent de propriétés que nous allons examiner dans cette section.

2.4.1 Capacités émergentes

Une capacité des LLMs est considérée comme émergente lorsqu'elle n'est pas présente dans les modèles de taille inférieure, mais qu'elle apparaît dans les versions plus grandes [30]. Dans la suite, nous allons présenter brièvement quelques-unes de ces capacités émergentes .

Apprentissage en contexte (few-shot learning)

Dans le domaine de l'intelligence artificielle, l'apprentissage *few-shot* émerge comme une approche innovante permettant de résoudre des problèmes complexes avec un nombre réduit de données d'entraînement. Cette technique a un impact majeur sur divers domaines, allant de la classification à la compréhension du langage naturel. En rendant possible l'apprentissage efficace à partir d'un nombre limité d'exemples, l'apprentissage *few-shot* constitue une méthode prometteuse pour concevoir des systèmes d'IA plus adaptatifs et performants. L'apprentissage *few-shot* appartient à une catégorie plus large appelée apprentissage *n-shot*, qui inclut également l'apprentissage *one-shot* (basé sur un seul exemple étiqueté par classe) et l'apprentissage *zero-shot* (ne nécessitant aucun exemple étiqueté). Cette famille de techniques vise à reproduire la capacité humaine à apprendre avec très peu d'exemples, marquant ainsi un changement de paradigme significatif dans le domaine de l'intelligence artificielle [18].

Figure 10 – Exemple d'apprentissage en contexte à 3-shot pour la traduction de l'anglais vers le français [4].

Dans l'exemple illustré par la figure 10, pour une tâche de traduction de l'anglais vers le français, le prompt inclut une description :

Translate English to French :

suivie de plusieurs exemples :

sea otter → loutre de mer, peppermint → menthe poivrée, plush giraffe → girafe peluche

puis d'un nouveau mot à traduire :

cheese →

Étant donné que trois exemples ont été fournis au modèle, il s'agit d'un apprentissage en contexte à 3-shot. Sans avoir été spécifiquement entraîné sur cette tâche, le modèle est capable de généraliser et de fournir des traductions correctes.

Raisonnement multi-étapes

Le raisonnement multi-étapes (*multi-step reasoning*) est une capacité émergente qui permet aux grands modèles de langage de résoudre des tâches complexes nécessitant plusieurs étapes de raisonnement. Cette capacité a également été observée

lorsque l'on combine l'apprentissage en contexte *few-shot* avec des explications fournies après la réponse finale.

La technique la plus couramment utilisée pour faire ressortir cette capacité est l'utilisation des « chaînes de raisonnement » (*Chain-of-Thought*, CoT)[30], qui guident les modèles à produire une séquence d'étapes intermédiaires avant de donner la réponse finale. Cette approche consiste à décomposer la tâche en plusieurs sous-tâches, chaque étape étant clairement énoncée et résolue avant de passer à l'étape suivante .

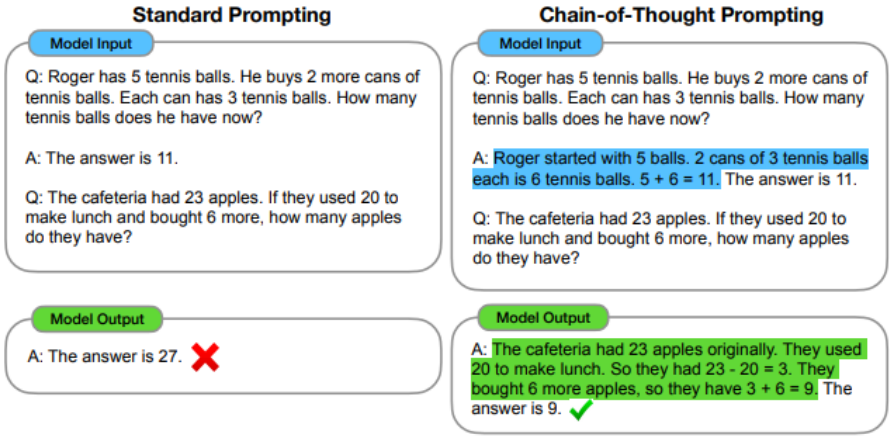

Figure 11 – Exemple de chaînes de raisonnement (CoT) pour résolution d'un problème mathématique [30].

Dans la figure 11, le modèle reçoit d'abord un exemple détaillé lui montrant comment décomposer un problème et résoudre chaque étape afin d'obtenir la réponse finale. Ensuite, il doit appliquer le même type de raisonnement pour résoudre un nouveau problème.

le suivi d'instructions

Le suivi d'instructions est une capacité émergente des modèles de langage de grande taille (LLMs), leur permettant d'exécuter diverses tâches à partir d'instructions écrites. Cette aptitude repose sur une approche appelée adaptation basée sur les instructions, qui consiste à entraîner ces modèles sur des ensembles de données spécifiques. Grâce à cette méthodologie, les LLMs peuvent accomplir des tâches déjà rencontrées lors de leur entraînement ou totalement nouvelles, sans nécessiter d'exemples explicites. Cette technique s'avère particulièrement efficace pour les modèles de grande envergure [31].

la figure 12 illustre une comparaison entre les réponses de *GPT-3 175B* et *InstructGPT 175B* à une même question : "What happens if you fire a cannonball directly at a pumpkin at high speeds ?"

L'analyse de ces réponses met en évidence l'amélioration du suivi des instructions et de la cohérence des réponses dans *InstructGPT* par rapport à *GPT-3*.

GPT-3 175B génère une réponse hallucinée, affirmant que la citrouille agit comme un aimant puissant capable de manipuler des objets métalliques.

Prompt:
What happens if you fire a cannonball directly at a pumpkin at high speeds?

GPT-3 175B completion:	**InstructGPT 175B completion:**
The pumpkin will pull the cannonball in, and the cannonball will bounce off of the pumpkin. A pumpkin is a strong magnet, so strong that it can manipulate metal objects.	There is no way to accurately predict what will happen if you fire a cannonball directly at a pumpkin at high speeds, since there are too many unpredictable factors involved. Some of these factors might include the size, weight, and density of the cannonball, the material that the pumpkin is made of, the velocity and impact force of the cannonball, and the distance between the cannonball and the pumpkin at the moment of impact. Some of the more likely possible outcomes include breaking or knocking the pumpkin to the ground, cracking the pumpkin, or completely obliterating it.

Figure 12 – une comparaison entre les réponses de GPT-3 175B et InstructGPT 175B à une même question [19].

InstructGPT 175B fournit une réponse plus logique et réaliste, expliquant les facteurs physiques qui influencent l'impact et listant des résultats plausibles comme la rupture ou l'éclatement de la citrouille.

2.5 Limites des LLMs

Bien que les grands modèles de langage (LLMs) aient démontré des performances remarquables dans la compréhension et la génération de texte, ils présentent plusieurs limitations qui peuvent affecter leur fiabilité et leur utilité. Parmi ces limites, on retrouve notamment l'hallucination, l'incohérence du raisonnement, les erreurs dans les calculs numériques et la difficulté à mettre à jour leurs connaissances. Cette section explore ces défis en analysant leurs causes, leurs implications et les stratégies envisagées pour les atténuer.

2.5.1 Hallucination

Définition 1 : L'**hallucination** dans les modèles de langage de grande taille (LLMs) désigne la génération de contenu erroné ou non fondé sur les données d'entraînement du modèle [14]. Ce phénomène se traduit par la production d'informations incohérentes, inventées ou factuellement incorrectes, compromettant ainsi la fiabilité des réponses fournies .

L'une des principales causes de ces erreurs réside dans la nature statique des connaissances des LLMs, qui ne sont mises à jour qu'à travers un nouvel entraînement, limitant ainsi leur capacité à intégrer des informations récentes ou spécialisées. De plus, l'absence d'accès en temps réel à des sources externes fiables renforce ce risque, en particulier dans des domaines où la précision est essentielle, comme la médecine ou le droit . Pour atténuer ces limitations, des approches récentes explorent l'intégration de mécanismes de récupération d'information permettant au modèle d'accéder à des bases de connaissances externes lors de la génération de texte, réduisant ainsi la fréquence des hallucinations et améliorant la pertinence des réponses [15].

2.5.2 Raisonnement avancé

Définition 2 : Le **raisonnement avancé** désigne la capacité d'un système à analyser, structurer et relier des informations de manière logique afin de résoudre des problèmes complexes, au-delà de la simple reconnaissance de patterns statistiques. Il inclut des compétences telles que l'inférence causale, le raisonnement mathématique, et la planification stratégique [20].

Bien que les modèles de langage de grande taille (LLMs) excellent dans la génération de texte fluide et la reformulation d'informations, ils rencontrent des difficultés significatives lorsqu'il s'agit de résoudre des problèmes nécessitant un raisonnement avancé. Ces modèles opèrent principalement en identifiant des patterns statistiques dans les données d'entraînement, sans véritable capacité de raisonnement symbolique ou logique [20]. Par conséquent, ils peinent à exécuter des tâches impliquant plusieurs étapes de raisonnement logique, comme les démonstrations mathématiques, la planification stratégique ou la résolution de problèmes nécessitant une inférence causale.

Dans une étude expérimentale menée dans le cadre de travaux exploratoires sur l'intelligence artificielle générale, Bubeck, Chandrasekaran et Eldan [5] (2023) ont évalué les capacités de *GPT-4* sur divers tests de raisonnement logique. Leurs résultats montrent que les modèles plus avancés, tels que *GPT-4*, échouent fréquemment sur des tests nécessitant une compréhension approfondie et une application correcte des règles logiques. Par exemple, sur des benchmarks mathématiques comme *MATH* et *GSM8K*, les performances des LLMs restent inférieures à celles d'experts humains, et les erreurs produites sont souvent masquées par la fluidité du langage généré . Une des raisons principales de ces échecs réside dans l'incapacité des modèles à manipuler et stocker des représentations abstraites de concepts complexes, contrairement aux approches traditionnelles en intelligence artificielle basées sur des règles formelles [16].

L'intégration d'architectures hybrides associant des modèles neuronaux à des moteurs de raisonnement symbolique est une approche visant à surmonter ces limitations.[16]. Ces approches visent à améliorer la capacité des LLMs à exécuter des raisonnements précis et explicables, notamment en les couplant à des systèmes capables de vérifier leurs réponses et de corriger leurs erreurs. Toutefois, ces solutions restent encore à un stade expérimental et nécessitent une amélioration de la capacité des modèles à généraliser au-delà des données vues en entraînement.

2.5.3 Calcul numérique

Définition 3 : Le **calcul numérique** désigne l'ensemble des méthodes et techniques permettant de résoudre des problèmes mathématiques en utilisant des opérations arithmétiques et algébriques, souvent mises en œuvre à l'aide d'algorithmes et de systèmes informatiques . Il inclut notamment la résolution d'équations, le calcul matriciel, l'optimisation numérique et les simulations scientifiques.

Résultat expérimental

Les modèles de langage, comme `ChatGPT`, rencontrent souvent des difficultés dans le calcul numérique précis. Prenons un exemple simple, mais significatif :

Supposons la somme suivante à calculer :

$$1+1+1+3+1+1+1-3+2+1+2+3$$

Le résultat exact de cette opération est 14. Cependant, lorsque `SUNMAN [25]` a posé cette question à *ChatGPT*, le modèle a répondu 18, une erreur de 4 unités . Cette erreur met en lumière les limites du modèle, qui, bien que performant dans la génération de texte, peut facilement faire des erreurs de calcul même sur des opérations simples.

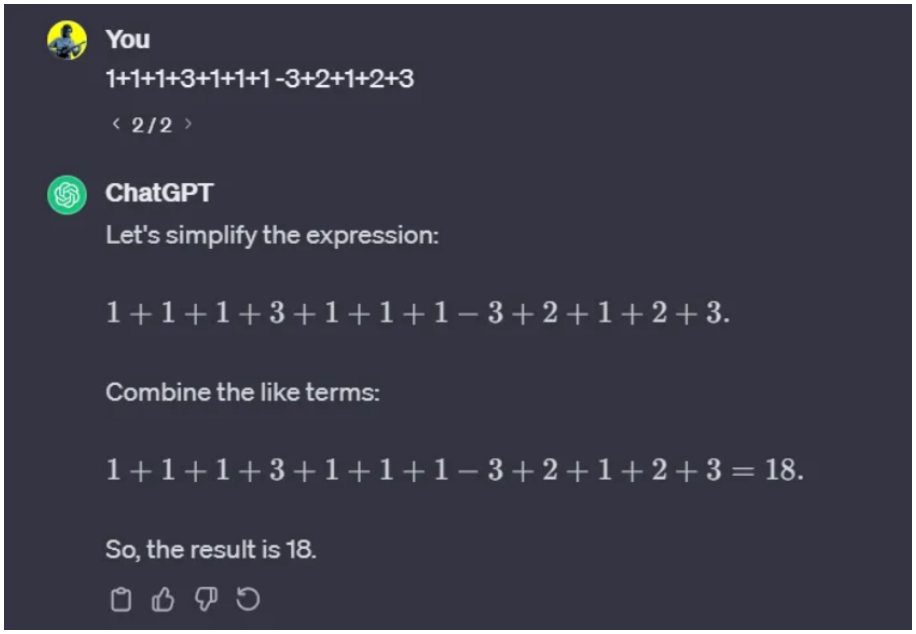

Figure 13 – Exemple d'erreur que `ChatGPT` peut commettre [25]

Analyse de l'erreur

Cette erreur provient du fait que *ChatGPT* ne suit pas une procédure algorithmique rigoureuse pour les calculs numériques. Au lieu de cela, il prédit les séquences de texte en fonction des modèles de probabilité qu'il a appris lors de son entraînement. Dans ce cas, l'erreur de calcul est due à une interprétation statistique des chiffres, où le modèle n'applique pas systématiquement les règles arithmétiques de manière fiable.

2.6 Conclusion

Ce chapitre nous a permis de développer une compréhension approfondie du domaine des grands modèles de langage, en suivant les principales étapes de leur cycle de développement. Nous avons d'abord exploré les choix architecturaux, les données

et les techniques d'entraînement utilisées lors de la phase de pré-entraînement. Ensuite, nous avons examiné les différentes méthodes d'adaptation de ces modèles à des tâches spécifiques, en mettant particulièrement l'accent sur l'adaptation basée sur les instructions et l'alignement. Nous avons également discuté de certaines propriétés des grands modèles de langage, telles que leurs capacités émergentes. Enfin, nous avons mis en lumière les principales limites auxquelles ces modèles sont confrontés.

Conclusion

Depuis l'émergence de l'intelligence artificielle, l'humanité s'efforce continuellement de concevoir des machines capables de comprendre et de produire divers types de communication. Les modèles de langage, qui ont évolué des systèmes statistiques aux réseaux neuronaux profonds, sont le fruit de cette quête incessante. Leur objectif est de saisir le contexte, de générer du contenu innovant et d'interagir avec les utilisateurs de manière plus naturelle. Les progrès réalisés aujourd'hui résultent de nombreuses années de travail acharné, de recherche et de développement.

Dans cette étude, nous avons retracé l'évolution de ce domaine, en commençant par une présentation générale des modèles de langage, puis en explorant les versions plus avancées de ces modèles. Dans un premier temps, nous avons introduit le sujet en fournissant différentes définitions issues de la littérature. Ensuite, nous avons retracé l'évolution historique des modèles de langage afin de mieux comprendre leur développement. Cela inclut une présentation des techniques statistiques, ainsi que celles basées sur l'apprentissage profond.

Nous avons ensuite détaillé les grands modèles de langage, en analysant chaque étape de leur développement, du pré-entraînement. Nous avons exploré leurs architectures, les données d'entraînement et les techniques utilisées dans l'entraînement. Nous avons également abordé les méthodes d'adaptation, notamment l'adaptation basée sur les instructions et l'adaptation par alignement.
Enfin, nous avons présenté les propriétés et les limites de ces modèles.

Bien que le domaine de la modélisation du langage ait traversé plusieurs phases distinctes, les récentes avancées, en particulier l'émergence des grands modèles de langage (LLMs), ont révolutionné l'état de l'art pour de nombreuses tâches linguistiques. Cependant, la recherche sur ces modèles est loin d'être terminée. De nouvelles technologies verront probablement le jour, remplaçant les concepts actuels et marquant une nouvelle ère dans ce domaine. En attendant cette révolution à venir, nous pensons que l'amélioration des grands modèles de langage passera par une meilleure compréhension et une étude approfondie de leurs propriétés.

Références

[1] A. Amidi et S. Amidi. *CS 230 – Pense-bête de réseaux de neurones récurrents*. Récupéré le 1 janvier 2024, à partir de https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-recurrents#overview. 2019.

[2] A. Askell et al. « A General Language Assistant as a Laboratory for Alignment ». In : *arXiv preprint arXiv :2112.00861* (2021). url : https://doi.org/10.48550/arXiv.2112.00861.

[3] D. Bahdanau, K. Cho et Y. Bengio. « Neural Machine Translation by Jointly Learning to Align and Translate ». In : *3rd International Conference on Learning Representations, ICLR 2015*. 2014.

[4] T. Brown et al. « Language Models are Few-Shot Learners ». In : *Advances in Neural Information Processing Systems*. 2020, p. 1877-1901.

[5] S. Bubeck, V. Chandrasekaran et R. Eldan. « Sparks of artificial general intelligence : Early experiments with GPT-4 ». In : *arXiv preprint arXiv :2303.12712* (2023).

[6] Zhibo Chu et al. « History, Development, and Principles of Large Language Models—An Introductory Survey ». In : *arXiv preprint arXiv :2402.06853* (2024).

[7] Hyung Won Chung et al. « Scaling Instruction-Finetuned Language Models ». In : *arXiv* 2210.11416v5 (déc. 2022). Version 5, consulté le 17 février 2025. arXiv : 2210.11416 [cs.LG]. url : https://arxiv.org/pdf/2210.11416.

[8] Cloudflare. *Qu'est-ce qu'un grand modèle de langage (LLM) ?* Consulté le 22 mars 2025. 2024. url : https://www.cloudflare.com/fr-fr/learning/ai/what-is-large-language-model/.

[9] DataCamp. *What is Tokenization ?* Accessed : 2025-03-01. 2025. url : https://www.datacamp.com/fr/blog/what-is-tokenization.

[10] Wenhao Ding et al. « Seeing is not Believing : Robust Reinforcement Learning against Spurious Correlation ». In : *arXiv* 2307.07907v1 (juill. 2023). Consulté le 17 février 2025. arXiv : 2307.07907 [cs.LG]. url : https://arxiv.org/pdf/2307.07907.

[11] L. Gao et al. « The Pile : An 800GB Dataset of Diverse Text for Language Modeling ». In : *arXiv* arXiv :2101.00027 (2020). doi : 10.48550/arXiv.2101.00027. url : https://doi.org/10.48550/arXiv.2101.00027.

[12] D. Hiemstra. « A probabilistic justification for using tf× idf term weighting in information retrieval ». In : *International Journal on Digital Libraries* 3.2 (2000), p. 131-139.

[13] Innovatiana. *Apprentissage par RLHF pour les LLMs et autres modèles*. Consulté le [date d'accès]. 2024. url : https://www.innovatiana.com/post/rlhf-our-detailed-guide.

[14] Z. Ji et al. « Survey of Hallucination in Natural Language Generation ». In : *ACM Computing Surveys (CSUR)* 55.12 (2023), p. 1-38.

[15] P. Lewis et al. « Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks ». In : *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[16] G. Mialon, P. Dognin et J. Sennhauser. « Mathematical reasoning in neural models : How far can they go ? » In : *Journal of Artificial Intelligence Research* (2023).

[17] T. Mikolov et al. « Efficient Estimation of Word Representations in Vector Space ». In : *arXiv preprint arXiv :1301.3781* (2013).

[18] Nicolas. *Apprentissage Few Shot : définition et cas d'utilisation*. Consulté le 17 février 2025. Sept. 2024. url : https://www.innovatiana.com/post/few-shot-learning-in-ai.

[19] L. Ouyang et al. « Training language models to follow instructions with human feedback ». In : *Advances in Neural Information Processing Systems*. Sous la dir. de S. Koyejo et al. T. 35. Curran Associates, Inc., 2022, p. 27730-27744.

[20] N. Rach, Y. Lu et B. Schölkopf. « Longitudinal evaluation of reasoning in large language models ». In : *NeurIPS Proceedings* (2021).

[21] A. Radford et al. « Language Models are Unsupervised Multitask Learners ». In : *OpenAI Blog* (2019). url : https://openai.com/blog/language-unsupervised/.

[22] M. Raeini. « The evolution of language models : From n-grams to LLMs, and beyond ». In : *SSRN Electronic Journal* (2023).

[23] C. E. Shannon. « A mathematical theory of communication ». In : *Bell System Technical Journal* 27.3 (1948), p. 379-423.

[24] Bruno Stuner. *Cohorte de réseaux de neurones récurrents pour la reconnaissance de l'écriture*. Réseau de neurones [cs.NE], Normandie Université. 2018.

[25] K. Sunman. « Two Things ChatGPT Can Get Wrong : Math and Facts ». In : *Medium* (2023). Accessed : 2025-03-21. url : https://medium.com/@k.sunman91/two-things-chatgpt-can-get-wrong-math-and-facts-why-b40e50dc68cd.

[26] R. Taylor et al. « Galactica : A Large Language Model for Science ». In : *arXiv* arXiv :2211.09085 (2022). doi : 10.48550/arXiv.2211.09085. url : https://doi.org/10.48550/arXiv.2211.09085.

[27] Unite.AI. *Decoder-Based Large Language Models : A Complete Guide*. Consulté le 17 février 2025. 2025. url : https://www.unite.ai/fr/decoder-based-large-language-models-a-complete-guide/.

[28] A. Vaswani et al. « Attention is All you Need ». In : *Advances in Neural Information Processing Systems*. 2017, p. 1-12.

[29] Richard Y. Wang et Diane M. Strong. « Beyond Accuracy : What Data Quality Means to Data Consumers ». In : *Journal of Management Information Systems* (1996).

[30] J. Wei et al. « Chain-of-thought prompting elicits reasoning in large language models ». In : (2022).

[31] Jason Wei et al. « Finetuned language models are zero-shot learners ». In : *arXiv preprint arXiv :2109.01652* (2021).

[32] Chung-Hsien Wu, Yu-Hsien Chiu et Chi-Shiang Guo. « Text Generation From Taiwanese Sign Language Using a PST-Based Language Model for Augmentative Communication ». In : *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 12.4 (2004), p. 441-446.

[33] K. Yang et al. « If LLM Is the Wizard, Then Code Is the Wand : A Survey on How Code Empowers Large Language Models to Serve as Intelligent Agents ». In : *arXiv* abs/2401.00812 (2024). url : `https://arxiv.org/abs/2401.00812`.

[34] W. X. Zhao et al. « A Survey of Large Language Models ». In : *arXiv preprint* 2303.18223 (2023). doi : `10.48550/arXiv.2303.18223`.