Thierry-Séphine Goma-Legernard
thierry-sephine.goma@polytechnique.edu

Année scolaire 2023-2024
M2 Data Sciences

IA générative et agents conversationnels :
Elaboration d'un corpus documentaire et méthodologique pour les projets et mise en application à travers la production d'une solution originale pour un client
Non-Confidentiel

Référent pédagogique
Rémi FLAMARY
CMAP École Polytechnique

Maître de stage
Thomas CZERNICHOW
CPO chez ALLONIA

Sommaire

Introduction : contexte et enjeux des assistants virtuels basés sur les *LLMs* et l'IA générative **1**

1 Méthodes et état de l'art : La place des modèles Transformers et du RAG dans l'implémentation d'assistants virtuels s'appuyant sur les *LLMs* **3**

1.1 Compréhension détaillée du fonctionnement des Transformers : le modèle original de 2017 . . . 3
1.1.1 Les données d'entrée et l'*embedding* . . . 3
1.1.2 Encoder . . . 5
1.1.3 Decoder . . . 8
1.1.4 Encoder-decoder attention et *output* du modèle . . . 9
1.1.5 Discussion relative à l'importance du nombre de têtes . . . 9
1.2 Développement et innovations des modèles *Transformers* . . . 11
1.2.1 Du Transformer 2017 à GPT-3 . . . 11
1.2.2 Optimisations Architecturales : MoE (Mixture of Experts) . . . 11
1.3 Génération de texte par décodage auto-régressif avec Mistral 8x7B . . . 12
1.4 Le Retrieval-Augmented Generation (RAG) . . . 14
1.4.1 Présentation et mise en œuvre du modèle . . . 14
1.4.2 Mise en œuvre dans le développement d'un agent conversationnel . . . 15
1.4.3 Le RAG est-il en compétition avec le *fine-tuning* ? . . . 16
1.5 Les défis et les limites . . . 19

2 Contribution **20**

2.1 Gestion de projet en IA et employabilité des données . . . 20
2.2 Etude de la vectorisation des données tabulaires en entrée des *LLM* en vue d'un enrichissement des outils existants . . . 22
2.2.1 Problématique . . . 22
2.2.2 Étude menée . . . 23
2.2.3 Premières implémentations résultant de la recherche . . . 23
2.3 Réalisation de mission pour répondre aux besoins d'un client d'envergure : SPI . . . 25
2.3.1 Contexte et objectif . . . 25
2.3.2 Architecture de la solution proposée . . . 26
2.3.3 Paramètres . . . 26

3 Résultats : Évaluation des performances de l'assistant virtuel SPI **27**

3.1 Méthode d'évaluation de la performance du LLM développé . . . 27
3.2 Données nécessaires à l'évaluation . . . 28
3.3 Présentation des performances du modèle . . . 28
3.4 Perspectives d'enrichissement du modèle . . . 29

Conclusion **30**

Introduction : contexte et enjeux des assistants virtuels basés sur les *LLMs* et l'IA générative

Les avancées en matière de traitement du langage naturel (NLP) et de modèles de langage de grande taille (*LLM*) ont révolutionné le domaine de l'intelligence artificielle, nous plaçant désormais dans un paradigme où les textes sont calculables. Ces avancées ont été rendues possibles par une conjonction de facteurs, d'une part l'introduction des modèles *Transformers* particulièrement performants et dont le fonctionnement repose en partie sur des calculs matriciels, et d'autre part la disponibilité de GPU particulièrement adaptés aux calculs matriciels puisqu'ils tirent leur origine du traitement des images. La conjonction de ces technologies, permet dès lors un traitement approfondi des textes, ouvrant la voie à la réalisation de tâches toujours plus pertinentes, et soulevant ainsi la question de leur industrialisation et de leur application concrète dans divers secteurs industriels.

Dans le cadre de mon stage de fin d'études, j'ai souhaité me rapprocher des applications concrètes des modèles de langage (*LLMs*) et participer à la gestion de projets centrés sur des solutions d'intelligence artificielle (IA). J'ai effectué ce stage au sein de la société ALLONIA, qui exploite une plateforme SaaS permettant de développer, mettre en production et exploiter industriellement des modèles de Machine Learning, et notamment des *LLMs*. Cette entreprise, en pleine croissance, a notamment été sélectionnée au sein du projet *SCRIBE France2030*, au côté d'institutions telles que l'INRIA, témoignant ainsi de son rôle de premier plan dans le développement des technologies basées sur les *LLMs*.

L'activité d'ALLONIA comprend en 2024 une part significative de missions de création d'outils basés sur les *LLMs*, qui ont gagné en popularité auprès des clients depuis la démocratisation des *LLMs* par ChatGPT d'OpenAI. Ainsi, ALLONIA accompagne des acteurs très différents, qu'il s'agisse de collectivités territoriales telles que la région Île-de-France, de PME à travers le Pack IA du HUB France IA, ou encore d'entreprises et d'institutions privées. Le besoin exprimé peut varier d'une mission à l'autre : solutions de traitement des mails, conception d'assistant virtuel interne ou grand public, ou encore détection d'anomalies, pour citer quelques exemples.

Les ambitions fixées au commencement de mon stage de 5 mois étaient les suivantes :

- Suivre différents projets en tant que cheffe de projet, y apporter mon regard et les connaissances sur l'IA générative développées au cours de cette année scolaire dans la perspective de contribuer à bâtir un référentiel pour la gestion de projets IA.
- Mener des études et des revues scientifiques en vue d'améliorer les outils existants.
- Me permettre d'implémenter moi-même des solutions pour des clients, s'appuyant sur les *LLMs* et sur un modèle avec lequel je n'étais pas encore familière : le modèle *Retrieval-Augmented Generation* (RAG)
- Acquérir une meilleure compréhension du cheminement et des infrastructures nécessaires pour passer de l'implémentation en Python à la mise à disposition en ligne d'un produit fini pour un utilisateur.

Dans ce rapport, je m'attacherai à mettre en lumière les connaissances et compétences que j'ai développées au cours de ce stage.

Dans un premier temps, je proposerai une revue de l'état de l'art concernant les *LLMs* et le modèle RAG, une approche innovante et complémentaire aux techniques de *fine-tuning*. Cette analyse a été essentielle à ma compréhension des technologies sous-jacentes à l'offre d'ALLONIA.

Je présenterai ensuite ma contribution, qui inclut initialement mon apport à la création d'un référentiel pour la gestion de projets en intelligence artificielle.Ce travail a mis en lumière, d'une part, le rôle crucial des données, en soulignant l'importance de leur qualité et de leur exploitation optimale, et d'autre part, la nécessité d'un cadre d'évaluation des résultats produits par les modèles. C'est dans cette optique que j'ai mené ensuite une étude approfondie sur le traitement des données tabulaires en entrée des *LLMs*, en vue d'améliorer les outils existants. Enfin, je détaillerai une mission concrète qui a consolidé ces acquis, au cours de laquelle j'ai exercé les fonctions de cheffe de projet et data scientist, et implémenté des solutions d'intelligence artificielle générative adaptées aux problématiques spécifiques du client.

Je conclurai en m'intéressant à l'évaluation des performances des outils développés, tout en ouvrant la discussion sur les perspectives d'amélioration et les développements futurs.

Méthodes et état de l'art : La place des modèles Transformers et du RAG dans l'implémentation d'assistants virtuels s'appuyant sur les *LLMs*

Dans le contexte de mai 2024, la société ALLONIA exploite des techniques avancées de *LLM* afin de fournir à ses clients des solutions novatrices. Ces solutions comprennent des agents conversationnels capables de répondre aux requêtes des utilisateurs, ainsi que des assistants virtuels destinés à être un support pour les professionnels dans leurs tâches quotidiennes, telles que la gestion des e-mails clients, entre autres.

Bien qu'ALLONIA dispose d'infrastructures qui permettent de manier les *LLMs* avec aisance et simplicité, il m'a fallu développer une connaissance approfondie des technologies en jeu afin de pouvoir apporter une contribution à celles-ci. Ainsi, dans ce but j'ai effectué une revue bibliographique afin de comprendre l'état de l'art, les mécanismes qui sous-tendent l'implémentation d'assistants virtuels et le RAG.

1.1 Compréhension détaillée du fonctionnement des Transformers : le modèle original de 2017

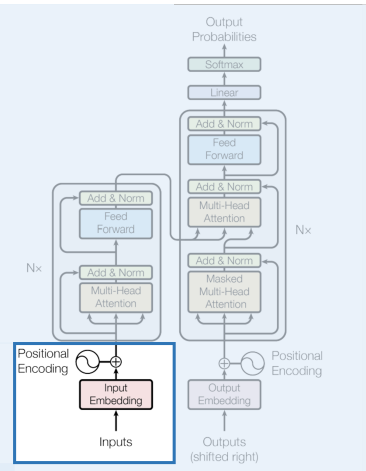
Le Transformer constitue un modèle d'apprentissage profond qui s'est imposé comme le pilier central de nombreuses méthodes avancées pour les tâches de compréhension et de génération du langage. Présenté et popularisé dans l'article intitulé Présenté et popularisé dans l'article intitulé *Attention is All You Need* [21] (Vaswani et al. 2017), qui traitait de traduction linguistique, les *Transformers* ont refaçonné le domaine *Natural Language Processing* (NLP) en permettant une gestion efficace des données séquentielles.
Je propose dans cette section une illustration détaillée du fonctionnement.

1.1.1 Les données d'entrée et l'*embedding*

Dans le papier original, le modèle est employé pour une tâche de traduction et il est fourni pour l'entraînement des paires de phrases dans deux langues différentes.

L'entraînement du Transformer nécessite un corpus $(C_i)_{i\in\mathcal{I}}$ et de se munir d'une paire d'éléments (x_i, y_i) où x_i est un input et y_i l'output attendu pour x_i
On considère le vocabulaire $\mathcal{V}$ associé au corpus.
On note $|\mathcal{V}|$ la taille du vocabulaire.

Embedding



Après que la dimension de l'*embedding* $dime_E$ ait été choisie, les inputs sont vectorisés via une couche d'*embedding* qui prend : en entrée des entiers représentant chaque élément du vocabulaire, avec des valeurs allant de 0 à $|\mathcal{V}| - 1$, et retourne des vecteurs denses de valeurs réelles, représentant ces *tokens* dans un espace vectoriel continu de dimension dime_E. Les vecteurs denses obtenus dépendent des poids de la couche d'*embedding*, qui sont en général initialisés de manière aléatoire. Ces poids seront ensuite mis à jour durant l'entraînement.
in : $t_i \in [0, |\mathcal{V}| - 1] \quad \longrightarrow \quad$ **out** : $e^i \in \mathbb{R}^{\text{dim}_E}$

FIGURE 1.1 – Architecture Transformer, zoom sur l'embedding. Source : [21]

J'ai cherché à comprendre les choix qui relevaient de l'usage et ceux qui étaient liés à des exigences computationnelles, notamment le choix de la dimension de l'*embedding*.
Le papier original utilise une dimension d'*embedding* de 512. Il ressort des éléments de ma revue de bibliographie ([3], [17]) que le choix de la dimension de l'*embedding* est un hyperparamètre, sur lequel il peut être pertinent de jouer pour impacter les performances du modèle.

Les ambiguïtés sémantiques, subtilités et nuances, ainsi que l'importance du contexte sont des facteurs qui orientent vers une augmentation de la dimension de l'*embedding*. Pour donner quelques ordres de grandeurs, le modèle BERT-Large comporte une dimension d'embedding de 1024, soit le double du modèle Transformer 2017, et la dimension d'embedding pour GPT-3 est variable, pouvant aller de 1536 jusqu'à 12288 (*déployement DaVinci*). Les limites imposées par les ressources computationnelles, ou une faible quantité de données disponibles sont des éléments qui conduisent à contenir cette quantité, cela permet notamment d'éviter l'overfitting ou des coûts computationnels qui ne seraient pas justifiables au regard de faibles données.

Positional Encoding

Un vecteur encodant la position de l'élément dans la phrase est également calculé. Sa dimension est identique à celle des vecteurs d'*embedding*. Ainsi pour chaque élément e^i, un vecteur de *positional encoding* p^i correspondant est calculé. Les composantes de ce vecteur sont $({p_k}^i)$, où $k \in [|0, \text{dim}_E)$
Les auteurs du papier original [21], indiquent que plusieurs choix peuvent être faits pour le *positional encoding*. Le choix qui a été fait dans le papier est le suivant. Pour un vecteur p^i donné, la valeur des $({p_k}^i)$ dépend :

— De la position de $t_i$dans la séquence d'input, que l'on note pos_i

— De la valeur de k

En notant $(k \bmod 2)$la congruence de k à 2, indiquant donc si k est pair ou impair, le calcul des composantes du vecteur de *positional encoding* $({p_k}^i)$ présenté dans le papier original est le suivant :

$$\forall k \in [|0, \text{dim}_E), \quad {p_k}^i = \begin{cases} \sin\left(\frac{pos_i}{10000^{k/\text{dim}_E}}\right) & \text{si } (k \bmod 2) = 0 \\ \cos\left(\frac{pos_i}{10000^{k-1/\text{dim}_E}}\right) & \text{si } (k \bmod 2) = 1 \end{cases}$$

Notons :

— la relation trigonométrique connue $\forall a \in \mathbb{R}, \sin(a + \frac{\pi}{2}) = \cos(a)$

— l'expression $2\left\lfloor \frac{k}{2} \right\rfloor$qui vaut k si $(k \bmod 2) = 0$ et $k - 1$ si $(k \bmod 2) = 1$

Combinées, elles permettent d'aboutir à une expression unique :

$$\forall k \in [|0, \text{dim}_E), \quad {p_k}^i = \sin\left(\frac{pos_i}{10000^{\frac{2\left\lfloor \frac{k}{2} \right\rfloor}{\text{dim}_E}}} + \frac{\pi}{2} \cdot (k \bmod 2)\right)$$

Les vecteur e^i et p^i ayant la même dimension, ils peuvent être sommés terme à termes. La somme $\widetilde{x_i} = e^i + p^i$ est utilisée comme input pour l'encoder du modèle.

1.1.2 Encoder

Le papier original introduit un bloc *MultiHead Attention*, où les sorties d'un nombre $h \in \mathbb{N}$ de têtes d'attention sont concaténées l'une après l'autre, avec $h = 8$.

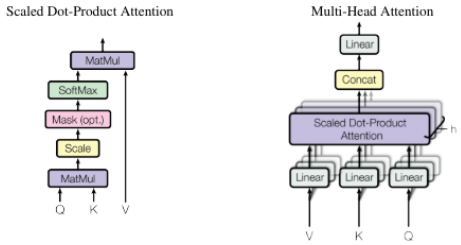


FIGURE 1.2 – Présentation des structures *scaled dot-product attention* et *MultiHead attention*. Source : [21]

La section suivante propose pour commencer une formalisation du fonctionnement de la *scaled dot-product attention* ou *simple head attention*.

Simple head scaled dot-product attention : Query, Key, Values

Dans la suite de notre explication, nous considérons $\tilde{x} \in \mathcal{M}_{n,E}$ tel que chaque ligne $\widetilde{x_i}$ correspond à la somme $(e^i + p^i)$ détaillée précédemment (1.1.1).
En soulignant qu'il est crucial de tenir compte de la *tokenisation*, on peut considérer que $\tilde{x}$ représente la vectorisation d'un élément du corpus de taille n, où chaque ligne $\widetilde{x_i}$ s'apparenterait à la représentation vectorielle du i-ème élément de la séquence d'input.

Trois matrices interviennent dans le processus : la matrice de *query* la matrice de *keys* la matrice de *values*, dont nous clarifions le rôle pour chacune dans la suite de cette section.

Vecteurs de Query

Pour chaque tête d'attention, une matrice de query est calculée à partir de $\tilde{x} \in \mathcal{M}_{n,E}$, représentation vectorielle des données d'entrée, et des poids $W^Q \in \mathcal{M}_{E,d_q}$, paramètres du modèle. La valeur de d_q est un hyperparamètre. Dans le papier original, les auteurs choisissent $d_q = \frac{\dim_E}{h}$ où h est le nombre de têtes.

L'opération suivante est réalisée :

$$\textbf{in} : \begin{array}{ll} \tilde{x} & \in \mathcal{M}_{n,E} \\ W^Q & \in \mathcal{M}_{E,d_q} \end{array} \quad \longrightarrow \quad \textbf{out} : Q = \tilde{x} \times W^Q \in \mathcal{M}_{n,d_q}$$

Ayant présenté cette première opération élémentaire d'obtention des vecteurs de *queries*, je la mettrai en lien avec les autres opérations afin de clarifier l'objectif qu'elle permet d'atteindre (1.1.2).

Vecteurs de keys

Pour chaque tête d'attention, une matrice de *keys* est calculée à partir de $x \in \mathcal{M}_{n,E}$ représentation vectorielle des données d'entrée, et des poids $W^K \in \mathcal{M}_{E,d_k}$, paramètres du modèle. La valeur de d_k est identique à celle de d_q, afin de permettre d'effectuer des opérations entre les différents vecteurs et matrices. L'opération suivante est réalisée :

$$\textbf{in} : \begin{array}{ll} \tilde{x} & \in \mathcal{M}_{n,E} \\ W^K & \in \mathcal{M}_{E,d_k} \end{array} \quad \longrightarrow \quad \textbf{out} : K = \tilde{x} \times W^K \in \mathcal{M}_{n,d_k}$$

Ayant présenté cette première opération élémentaire d'obtention des vecteurs de *keys*, je la mettrai en lien avec les autres opérations afin de clarifier l'objectif qu'elle permet d'atteindre (1.1.2).

Emploi des vecteurs de Query et Key pour le calcul des poids d'attention : Matmul, Scale et Softmax

Les matrices Q et K ainsi obtenues sont utilisées pour calculer par multiplication matricielle les scores d'attention. Ces scores permettent ensuite d'obtenir des poids d'attention normalisés. Ces probabilités reflètent l'importance relative de chaque *token* par rapport aux autres *tokens* dans la séquence. À noter que $d_q = d_k$.

$$\textbf{in} : \begin{array}{l} Q \in \mathcal{M}_{n,d_k} \\ K \in \mathcal{M}_{n,d_k} \end{array} \quad \longrightarrow \quad \textbf{out} : {}^{\text{Self}}AW = \sigma\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \in \mathcal{M}_{n,n}$$

On observe que la matrice d'*attention weights* ${}^{\text{Self}}AW$ obtenue en sortie est une matrice carrée dont la somme des éléments de chaque ligne est égale à 1. Cette matrice est construite de telle que le coefficient en position $\{i, j\}$ reflète l'importance relative du j-ème élément de la séquence d'entrée pour le i-ème élément de la séquence de sortie.

Vecteurs de values

Pour chaque tête d'attention, une matrice de values est calculée à partir de $x \in \mathcal{M}_{n,E}$ représentation vectorielle des données d'entrée, et des poids $W^V \in \mathcal{M}_{E,d_v}$, paramètres du modèle.

Conventionnellement, il est courant d'avoir $d_v = d_k$ Toutefois, rien n'empêche théoriquement d'avoir $d_v \neq d_k$. L'opération suivante est réalisée :

$$\textbf{in} : \begin{matrix} \tilde{x} & \in \mathcal{M}_{n,E} \\ W^V & \in \mathcal{M}_{E,d_v} \end{matrix} \quad \longrightarrow \quad \textbf{out} : V = \tilde{x} \times W^V \in \mathcal{M}_{n,d_v}$$

Output de la Single-Head Attention

Les poids d'attention sont ensuite multipliés par la matrice de Value (V) pour obtenir les vecteurs d'attention pondérés. Ces vecteurs sont des combinaisons linéaires des vecteurs de values, pondérées par l'importance calculée.

$$\textbf{in} : \begin{matrix} {}^{\text{Self}}AW = \sigma\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) & \in \mathcal{M}_{n,n} \\ V & \in \mathcal{M}_{n,d_v} \end{matrix} \quad \longrightarrow \quad \textbf{out} : \text{head} = {}^{\text{Self}}AW \times V \in \mathcal{M}_{n,d_v}$$

Ainsi, les opérations permettant d'obtenir une tête d'attention se présentent comme une succession de produits scalaires et de multiplications matricielles avec des matrices de poids. La valeur optimale de ces poids, permettant d'atteindre de bonnes performances pour la tâche choisie, est obtenue par entraînement du modèle.

Il ressort de ma revue, en m'appuyant notamment sur les travaux de [13] qui fournissent des perspectives complémentaires à [21], que la raison pour laquelle la *simple head attention* permet de capturer efficacement l'information réside dans sa capacité à modéliser les relations entre les différents éléments d'une séquence.

Le vecteur final constitué de la ligne i de la tête $head$ représente la "compréhension" du modèle du i-eme input en tenant compte des autres mots de la séquence. Ainsi, si un mot x_i est particulièrement important pour la compréhension d'un autre mot x_j, le poids d'attention entre ces deux mots sera élevé. Cela signifie que l'information de x_i est fortement intégrée dans la nouvelle représentation de x_j. Comme souligné par [13] les relations et les dépendances identifiées par les poids d'attention permettent un enrichissement enrichissement contextuel.

Concaténation de toutes les têtes

La *MultiHead* est obtenue en concaténant l'ensemble des $head_i$ issues des *scaled dot-product attention*. Sur la base des inputs x, chaque tête d'attention $head_i$ calcule ses propres requêtes, clés et valeurs en utilisant les transformations linéaires détaillées ci-dessus. Si un nombre h de têtes d'attention sont concaténées et que chaque tête produit une sortie de dimension d_v, alors la concaténation des sorties des h têtes résulte en un élément de dimension $h \times d_v$.
Aussi, le résultat de la concaténation est multiplié par une matrice de poids de projection finale W^0 qui est employée afin de redimensionner de façon idoine la concaténation vers la dimension d'origine du modèle en vue de garantir la faisabilité des opérations ultérieures.

La concaténation des sorties des têtes d'attention permet de combiner les différentes perspectives capturées par chaque tête, offrant ainsi une vue d'ensemble plus riche et détaillée des relations présentes dans les données.
Cette approche multi-têtes augmente la capacité du modèle à saisir des dépendances complexes

et à représenter des structures sous-jacentes plus diversifiées car elle enrichit la représentation finale en intégrant des informations provenant de multiples sous-espaces de représentation. Cette intégration améliore significativement la performance et la flexibilité du modèle, rendant possible la capture de relations complexes au sein des séquences d'entrée.

Dans ma démarche de revue de l'état de l'art, j'ai questionné l'impact du nombre de têtes. Notamment, j'ai eu à cœur de mieux comprendre pourquoi il n'y a pas de redondance entre les têtes d'attention ? Les travaux de [4] et ceux de [8] ont apporté une réponse en mettant en évidence l'apprentissage diversifié rendu possible par la propagation de gradients indépendants : pendant l'entraînement, les gradients rétropropagés pour chaque tête sont différents car chaque tête a ses propres paramètres dans la matrice de projection W^0, ce qui conduit à une spécialisation des têtes pour différents aspects du modèle. Ainsi, je comprends qu'à moins d'une coïncidence où la matrice W^0 serait de très faible rang, les différentes têtes apprennent à capturer des aspects complémentaires de l'information.

En effet, pour ce qui est de la rétropropagation, si l'on se concentre sur le bloc Encoder de l'architecture, l'erreur est d'abord propagée à travers les couches feed-forward du bloc. Les gradients des poids des deux couches linéaires sont calculés et utilisés pour mettre à jour ces poids via la méthode de descente de gradient. Ensuite, l'erreur est propagée en arrière à travers la matrice de projection W^0. Les gradients par rapport à W^0 sont calculés en fonction des gradients du vecteur z', et les poids W^0 sont mis à jour. Puis l'erreur est répartie parmi les différentes têtes d'attention. Chaque tête reçoit une partie de l'erreur, et cette erreur est propagée à travers les paramètres W^Q, W^K, W^V de chaque tête. Les gradients des poids de ces matrices sont calculés et les poids sont mis à jour en conséquence.

Je me suis également demandé si l'ajout de nouvelles têtes était toujours un avantage en termes de performances. A ce sujet, l'expérience proposée par [16] offre une perspective intéressante. J'en discute plus en détail dans la section 1.1.5, car cela nécessite d'avoir un aperçu général des différents blocs de l'architecture.

FeedForward

Après la concaténation et l'obtention de l'output *MultiHead*, une couche *feed-forward* intervient. Comme le souligne [16], cette étape est cruciale pour permettre aux différentes têtes d'attention d'interagir entre elles. Elle peut être décomposée ainsi :

— Une couche linéaire avec une dimension d'entrée de dim_E, qui transforme les vecteurs d'entrée en représentations de dimension plus élevée, facilitant ainsi l'apprentissage de relations complexes.

—Une couche de régularisation ReLU (Rectified Linear Unit) qui a pour fonction d'introduire de la non-linéarité dans le réseau.

— Une seconde couche linéaire avec une dimension de sortie égale à dim_E, cette couche ramène les vecteurs à leur dimension initiale, assurant la cohérence dimensionnelle tout au long du modèle.

1.1.3 Decoder

Le bloc décodeur comprend deux mécanismes d'attention : le self-attention du décodeur et l'encodeur-décodeur attention.

Encoding, *embedding* et positional encoding

Les mécanismes d'Encoding, *embedding* et positional encoding sont similaires à ceux qui ont été décrits précédemment. La différence réside dans le fait qu'au sein de ce bloc, ce sont les données cibles y_i attendues qui sont employées

Decoder Self-Attention

De façon similaire aux mécanismes décrits, des matrices Q, K, et V sont obtenues pour le décodeur. La suite de cette section se concentrera sur l'Encoder-Decoder Attention.

1.1.4 Encoder-decoder attention et *output* du modèle

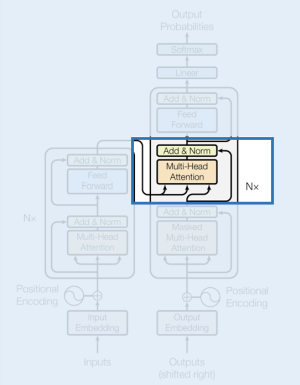


FIGURE 1.3 – Attention Encodeur-Decodeur. Source : [21]

Dans le mécanisme d'attention encoder-décodeur, les matrices de Value et de Keys proviennent de la sortie de l'encodeur, tandis que la matrice de Query est issue de la sortie de la couche d'auto-attention dans le décodeur. Afin de clarifier la notation, les éléments provenant de l'encodeur sont indexés par E et ceux issus du décodeur par D .

$$\textbf{in}: \quad \text{ED-AW} = \sigma\left(\frac{Q_D \cdot K_E^T}{\sqrt{d_k}}\right) \in \mathcal{M}_{n,n}$$
$$V_A \in \mathcal{M}_{n,d_v}$$

$$\textbf{out}: \quad \text{head} = \text{ED-AW} \times V_A \in \mathcal{M}_{n,d_v}$$

— Q : Matrice de requêtes
— K : Matrice de clés
— V : Matrice de valeurs
— d_k : Dimension de la clé (et requête)
— W^O : Matrice de projection de sortie
— W_i^Q, W_i^K, W_i^V : Matrices de projection pour la i-ème tête

Obtention de l'output

À l'issue de cette étape d'attention encoder-décodeur, des couches de projection linéaire, feed-forward entièrement connectées et de normalisation viennent compléter l'architecture. Les sorties du bloc décodeur sont passées à travers une couche dense finale suivie d'une activation softmax pour générer les probabilités de chaque token dans le vocabulaire. La sortie est ainsi obtenue en combinant les informations traitées par l'attention et ces couches supplémentaires, offrant la prédiction finale du modèle $\hat{y}$.

1.1.5 Discussion relative à l'importance du nombre de têtes

Le nombre de têtes d'attention et le nombre de couches d'attention pouvant être choisis selon les architectures, il m'a semblé pertinent d'investiguer l'impact de l'évolution du nombre de têtes sur la performance du modèle. Une étude de [16] a permis d'apporter des éléments de réponse.

Les auteurs proposent dans un premier temps une heuristique d'ablation itérative qui compare pour le modèle BERT Large à 16 têtes d'attention et 24 couches, l'évolution des performances à mesure qu'une proportion des têtes d'attention est élaguée incrémentalement.
Les performances mesurées pour BERT Large correspondent à l'*accuracy* pour une tâche d'inférence textuelle, s'appuyant sur les données du *MultiNLI* dataset.
L'ordre d'ablation des couches repose sur une mesure quantitative à même de refléter indirectement l'importance de la couche dans le modèle pour la tâche. Nommément, elle dépend de l'évolution marginale de la fonction de perte en masquant les paramètres de la couche donnée dans le calcul de la *MultiHead Attention*.

Les résultats de cette étude mettent en évidence que le nombre de têtes peut être réduit de 20% sans impact sensible sur les performances du modèle.

Dans un deuxième temps, l'expérience examine l'impact de l'élagage des têtes d'attention non pas globalement au sein de l'architecture mais selon l'un des trois types de mécanismes d'attention. Pour se faire les auteurs choisissent une tâche de traduction et un modèle issu de la compétition *Workshop on Machine Translation* qu'ils nomment *WMT model*. Comme le modèle Transformer 2017 présenté précédemment , le modèle WMT présente 3 blocs d'attention de différents types : **Enc-Enc** : Encodeur Self-Attention , **Enc-Dec** : Attention Encodeur-Decodeur ou **Dec-Dec** Decodeur Self-Attention

Cette distinction précisée, la méthodologie reste similaire : elle consiste à supprimer incrémentalement un pourcentage des têtes d'attention et à évaluer l'effet sur les performances de traduction, mesurées par les scores BLEU.

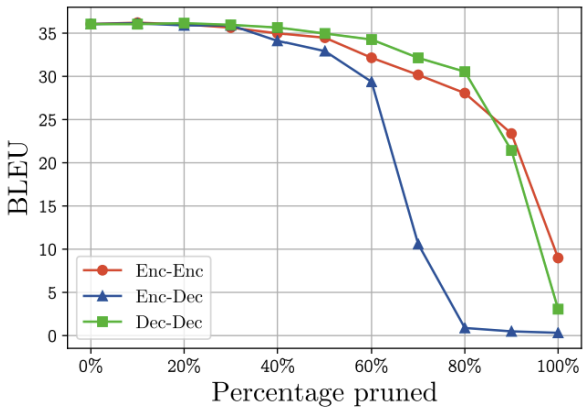


FIGURE 1.4 – Evolution du BLUE score du WMT model selon le % de tête élarguées, source [16]

Ces résultats mettent en évidence que :

— pour la partie **Enc-Dec** de l'architecture : La performance se dégrade rapidement lorsque plus de 60% des têtes sont élaguées, indiquant une forte dépendance à la multi-tête dans ces couches.

— pour les parties **Enc-Enc** et **Dec-Dec** : La qualité des traductions reste raisonnable (scores BLEU autour de 30) même avec seulement 20% des têtes d'attention, montrant une plus grande tolérance à l'élagage.

L'étude permet ainsi de conclure que, dans le cas d'un *WMT model* pour la tâche de traduction, l'attention encodeur-décodeur est plus critique pour la qualité des performances, nécessitant une plus grande proportion de têtes d'attention pour maintenir des performances optimales comparativement aux autres types d'attention.

1.2 Développement et innovations des modèles *Transformers*

1.2.1 Du Transformer 2017 à GPT-3

L'architecture *Transformer*, introduite en 2017 [21], a servi de fondement à de nombreuses avancées dans les modèles de *NLP*. Elle a permis de surmonter plusieurs limitations des architectures séquentielles, grâce à l'introduction du mécanisme d'attention.
Depuis, différents modèles, basés sur des variations de cette architecture, ont vu le jour, apportant chacun des innovations spécifiques. Cette section présente une chronologie simplifiée des principaux modèles, ainsi que leurs différences architecturales et innovations clés.

BERT : Un modèle basé uniquement sur l'Encodeur (2018) [8] Le modèle *BERT (Bidirectional Encoder Representations from Transformers)* est principalement basé sur la partie *encodeur* de l'architecture Transformer et peut contenir jusqu'à 340 millions de paramètres. Introduit par Devlin et al. en 2018, BERT est formé de manière bidirectionnelle : il prend en compte à la fois le contexte précédent et suivant dans une séquence. Il est pré-entraîné sur deux tâches principales : le *Masked Language Modeling* (MLM) et le *Next Sentence Prediction* (NSP).

T5 : Modèle Encodeur-Décodeur (2019) Le modèle *T5 (Text-To-Text Transfer Transformer)* de Google repose sur une architecture complète *encodeur-décodeur*. Introduit dans l'article [19], il comporte jusqu'à un milliard de paramètres. Il traite toutes les tâches de traitement du langage naturel comme une transformation de texte à texte, que ce soit pour la traduction, le résumé, ou d'autres tâches NLP.

GPT-2 et GPT-3 : Modèles basés uniquement sur le Décodeur (2019, 2020) Des modèles basés sur la partie *décodeur* de l'architecture Transformer ont été introduits pour l'inférence textuelle : ils prédisent le mot suivant en prenant en compte les tokens précédents. Un modèle comme GPT-3 excelle aujourd'hui dans une grande variété de tâches NLP, et comprend 175 milliards de paramètres.

1.2.2 Optimisations Architecturales : MoE (Mixture of Experts)

Les architectures comme celles de GPT sont dites denses, car chaque unité de calcul (neurone) est activée pour chaque token, ce qui signifie que toutes les couches du modèle sont sollicitées à chaque passage.
Ce fonctionnement induit des limitations : bien que puissants, ces modèles denses deviennent de plus en plus coûteux en termes de calcul à mesure que la taille des modèles augmente.

Pour améliorer les performances tout en réduisant les coûts computationnels, plusieurs techniques d'optimisation ont été introduites. L'une des plus intéressantes est l'architecture MoE (Mixture of Experts). MoE est une architecture conçue pour rendre les modèles plus efficaces en termes de calcul : l'idée est de diviser les neurones en plusieurs groupes appelés "experts". Chaque expert se spécialise dans une tâche spécifique ou un aspect des données. Lors de la formation et de l'inférence, seul un petit sous-ensemble d'experts est activé, guidé par un mécanisme de routage. Au lieu d'activer tous les neurones dans chaque couche comme dans les modèles denses, MoE active uniquement les experts les plus pertinents pour chaque token.

L'approche SMOE présentée dans le papier [20] introduit une activation '*sparse*' des experts, afin d'améliorer l'efficacité computationnelle et la scalabilité.
Les modèles SMOE, parmi lesquels on peut citer Artic [18] de Snowflake ou encore Mistral 8x7B [11], peuvent ainsi être beaucoup plus larges en termes de nombre de paramètres sans multiplier proportionnellement les coûts en calcul, car seuls certains experts sont activés pour chaque entrée.

Pour le développement d'agents conversationnels, ALLONIA a fait le choix de l'architecture Mistral 8x7B [11] dans un premier temps. Ce modèle est reconnu pour sa position avantageuse dans l'état de l'art, en particulier en langue française. J'en propose une vue d'ensemble dans la suivante section.

1.3 Génération de texte par décodage auto-régressif avec Mistral 8x7B

Mistral 8x7B est un modèle de langage de grande taille qui dispose d'un bon niveau dans la génération de texte et la réponse à des questions. Il se distingue particulièrement par ses performances remarquables en langue française.

Le choix d'ALLONIA s'est porté sur ce modèle de taille moyenne (environ 45 milliards de paramètres) en raison de plusieurs avantages : sa disponibilité en *open-source*, son entraînement sur une base de données significative en français, et sa capacité suffisante pour des tâches "conversationnelles" [1]
Une des particularités de ce modèle est son architecture dite *Sparse Mixture of Experts* (SMOE), qui permet un découpage stratégique des ressources computationnelles.

Le processus de génération de texte pour un agent conversationnel s'opère par **décodage auto-régressif** : tout d'abord, le texte d'entrée est transformé en une représentation vectorielle qui capture les informations contextuelles du texte. Cette transformation s'effectue au moyen d'opérations successives : tokenisation, embedding, et attention. Ces étapes convertissent le texte en vecteurs exploitables par le modèle pour générer des réponses.
Une fois le texte d'entrée *encodé*, le processus de génération du texte commence. Il est possible de spécifier une limite pour la longueur maximale de la sortie générée. La génération se fait de manière auto-régressive : le modèle prédit chaque *token* suivant en fonction des mots déjà générés et de l'entrée précédente, dans la limite de la taille de la fenêtre de contexte, en utilisant le décodeur du modèle Transformer pour cette tâche.
À chaque étape de la génération, le modèle prédit la distribution de probabilité de chaque token.

1. Depuis septembre 2024, ALLONIA utilise également le modèle Llama 3.1, qui possède les mêmes avantages.

Ensuite, il choisit soit le *token* le plus probable (décodage glouton), soit échantillonne un *token* en fonction de cette distribution de probabilité (échantillonnage stochastique), ce qui permet d'introduire une certaine diversité dans les générations.
Le processus de génération se poursuit jusqu'à ce qu'un *token* spécial de fin de séquence soit généré, ou jusqu'à ce que la limite de longueur soit atteinte.

Les auteurs du papier mettent en avant des spécificités du modèle, notamment la grouped-query attention (GQA) [1] et la sliding window attention (SWA) [6], dont la synergie pourrait améliorer la vitesse d'inférence et le traitement des séquences longues. Bien que ces techniques présentent un potentiel intéressant, une analyse plus approfondie serait nécessaire pour en évaluer pleinement l'impact. Les avantages évoqués plus haut sont ceux qui ont justifié le choix de Mistral 8x7B comme modèle sequence-to-sequence pour la génération de texte dans le cadre des agents conversationnels implémentés par ALLONIA.

Ayant introduit ici le modèle *sequence-to-sequence* (seq2seq) permettant de générer du texte, le chapitre suivant est consacré à la présentation du RAG, modèle qui, associé aux *LLMs*, est employé pour implémenter des solutions d'agents conversationnels pour les clients d'ALLONIA.

1.4 Le Retrieval-Augmented Generation (RAG)

1.4.1 Présentation et mise en œuvre du modèle

Le modèle Retrieval-Augmented Generation (RAG) proposé par Lewis et al. en 2020 [15] est un modèle s'appuyant sur un *LLM Transformer* seq2seq pré-entraîné qui permet notamment d'accomplir des tâches de question answering en s'appuyant à la fois sur une connaissance implicite paramétrique, acquise lors de l'entraînement du *LLM*, et sur une connaissance externe permettant d'étendre ou de réviser la mémoire du modèle.
L'architecture RAG combine deux modules. Le premier est un module de recherche (Retriever) qui s'appuie sur une requête et un corpus de contenu spécialisé pour y trouver de l'information. Le second est un module de génération de texte (Generator), modèle *LLM* pré-entraîné seq2seq, qui peut produire des réponses de qualité et spécifiques en ajoutant le résultat de la recherche au contexte.

Les auteurs schématisent son fonctionnement comme suit :

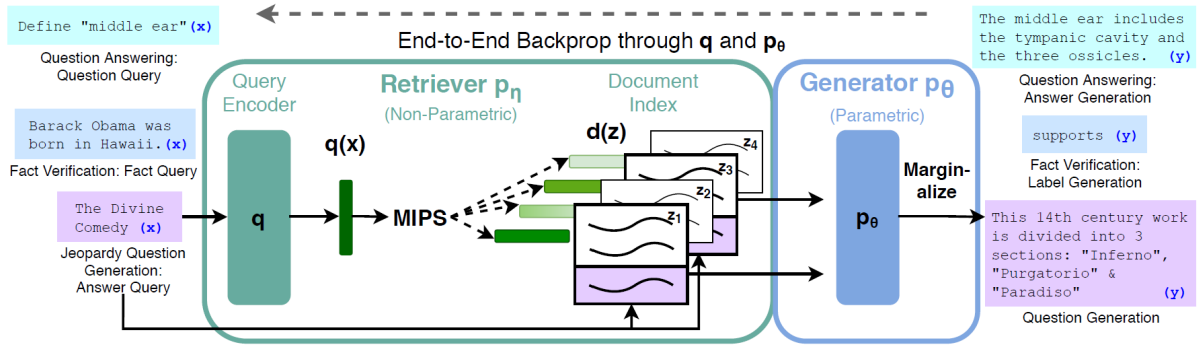


FIGURE 1.5 – Vue d'ensemble de l'approche RAG.

Applications possibles

Comme visible sur le schéma, bien que très utile dans le cadre du *question answering*, le RAG (Retrieval-Augmented Generation) est présenté par ses auteurs [15] comme un outil multifacette capable de réaliser trois tâches principales :

- **Open-domain question answering** : En combinant des connaissances paramétriques issues du modèle seq2seq pré-entraîné avec les informations récupérées de la mémoire externe non paramétrique, le modèle RAG parvient à fournir des réponses précises et appropriées au contexte.
- **Classification en trois classes (supports/refutes/not enough info)** : Le modèle peut être utilisé pour classifier la véracité d'une assertion via une classification en trois catégories, ce qui trouve son intérêt pour des tâches telles que le fact-checking.
- **Génération de questions ouvertes** : Le modèle RAG démontre enfin des capacités de génération dans un contexte connexe au *question answering*. Les auteurs proposent la génération de questions, sur le modèle du jeu populaire "Jeopardy". Cette tâche est présentée comme exigeante car elle nécessite de générer des questions précises et factuelles sur la base d'entités-réponses. Par exemple, en fournissant l'input « La Coupe du Monde », la question générée est « En 1986, le Mexique est devenu le premier pays à accueillir deux fois cette compétition sportive internationale. »

1.4.2 Mise en œuvre dans le développement d'un agent conversationnel

Éléments requis pour implémenter l'architecture RAG

Dans le cadre du développement d'un agent conversationnel basé sur le modèle RAG (Retrieval-Augmented Generation), plusieurs composants sont nécessaires pour constituer le *Retriever* et le *Generator*.

— Concernant le *retriever*, le premier élément est un *embeddings* EMB_α qui permettra d'obtenir des représentations denses des documents. Celui-ci permet de transformer les documents en représentations vectorielles. Dans le papier original, les auteurs choisissent les *embeddings* issus de $BERT_{base}$ [8]. En pratique le *retriever* est utilisé pour obtenir une représentation vectorielle d'un corpus supplémentaire contenant de l'information spécialisée.

— Un *embedding* de requêtes EMB_α est également utilisé pour produire une représentation vectorielle de la requête. Dans le papier original, l'*embedding* utilisé est également basé sur $BERT_{base}$.

— Concernant le module de génération, il nécessite un modèle *LLM* seq2seq performant. Le papier original propose BART-large [14], un *Transformer* seq2seq pré-entraîné avec 400 millions de paramètres.

Inputs fournis au modèle

— Question en langage naturel x — Corpus de documents : PDF, sites web, etc.

Étapes successives du fonctionnement

— **Étape 1** : Extraction des informations du corpus au format textuel et création de *chunks* (segmentation du corpus). Pour cette segmentation, des paramètres clés sont utilisés, notamment la taille des segments $chunk\_size$ et le chevauchement entre eux $chunk\_overlap$. Une liste de séparateurs courants, tels que ".", ", ", ou"$\backslash n$", peut également être spécifiée. Ces 3 paramètres sont fournis à un outil de division (*Text Splitter*), avec un dernier paramètre définissant la stratégie de découpe ainsi que la hiérarchie à suivre pour respecter les règles de segmentation.

— **Étape 2** : *embedding* de chaque *chunk* au moyen de EMB_α. Pour chaque *chunk* z_i, les *embeddings* $\widetilde{z}_i$ ont la même dimension. La concaténation de ces vecteurs conduit à la création d'une base de données vectorielle $d(z)$.

Ces deux étapes ne sont à effectuer qu'initialement ou en cas de mises à jour du corpus spécialisé. Une fois réalisées, des requêtes peuvent être soumises au modèle via un prompt.

— **Étape 3** : Obtention de l'*embedding* $\widetilde{x}$ de la question x au moyen de EMB_α.

— **Étape 4** : Recherche dans la base de données vectorielle des *chunks* les plus similaires. Pour ce faire, des mesures de similarité telles que la *cosine similarity* peuvent être utilisées. Les vecteurs provenant du même modèle d'*embedding* permettent des opérations comparatives fiables. À cette étape, on extrait un ensemble K parmi $\mathcal{I}$, correspondant aux $|K|$ *chunks* les plus similaires à $\widetilde{x}$. Dans le papier original, la mesure de similarité utilisée est le *Maximum Inner Product Search*, c'est-à-dire que les *chunks* sélectionnés maximisent le produit scalaire avec le vecteur d'*embedding* de la *query* $\widetilde{x}$. Le contenu récupéré est noté Z.

— **Étape 5** : Rédaction du prompt avec lequel on alimente le module générateur : Z et x sont concaténés avant d'être fournis en argument au générateur. Celui-ci reçoit cette concaténation comme **instruction finale** et génère une réponse.

1.4.3 Le RAG est-il en compétition avec le *fine-tuning* ?

Le modèle RAG propose de combiner un modèle seq2seq pré-entraîné avec une "mémoire externe" non paramétrique, tandis que le *fine-tuning* incorpore les connaissances supplémentaires directement dans le modèle lui-même. Face à ces comportements différents, il semble légitime de se demander si le RAG offre des avantages comparatifs par rapport aux approches de *fine-tuning*.

En janvier 2024, une équipe de chercheurs de Microsoft [5] a réalisé une étude visant à mettre en évidence les compromis entre ces différentes approches et à comparer leurs performances pour plusieurs modèles de langage (*LLMs*) populaires à cette date, notamment Llama2-13B, GPT-3.5 et GPT-4. Cette étude a été effectuée dans le cadre d'un cas d'usage lié au secteur agricole, présenté comme nécessitant des contextes spécifiques et des réponses adaptatives.

Pour l'évaluation, l'étude emploie AzureML Model Evaluation (Microsoft, 2023) et retient trois critères principaux pour mesurer la qualité des réponses en les évaluant de 1 à 5 sur la base d'une comparaison entre les résultats obtenus et les réponses de référence :
— La cohérence, qui mesure la concordance entre les *ground truth* et les prédictions, pour un contexte donné.
— La pertinence, qui évalue à quel point la réponse aborde les principaux aspects de la question en fonction du contexte.
— La fondation ou filiation (*groundedness*), qui détermine si la réponse découle logiquement des informations contenues dans le contexte.

Pour chacune de ces métriques, des exemples de bon score et de mauvais score sont disponibles (en pages 17 et 18 du papier [5]).

Les éléments saillants de cette étude sont les suivants :
— Le coût initial élevé dû au travail considérable requis pour fine-tuner le modèle sur de nouvelles données est un facteur important à prendre en compte.
— Toutefois, le *fine-tuning* s'est avéré utile pour enseigner au modèle de nouvelles compétences spécifiques au domaine agricole et pour fournir des réponses plus précises et succinctes qu'avant le *fine-tuning*.

Une comparaison des réponses de GPT-4 et d'un expert agronome à la même requête posée pour trois États américains différents met en évidence le résultat suivant :
— Alors qu'un expert fournit des réponses contextualisées basées sur la tradition climatique et agricole spécifique de chaque état, les *LLM* fine-tunés donnent une réponse générique qui, bien que correcte, n'est pas aussi précise pour chaque état que celle de l'expert.
— Le modèle RAG s'est avéré très efficace dans les cas où les données sont contextuellement pertinentes, comme dans l'interprétation des données agricoles, tout en conduisant également à des réponses plus succinctes que le modèle de base. De plus, les extraits d'informations auxiliaires sont cruciaux pour réduire les hallucinations et adapter la réponse à la région géographique ou au phénomène d'intérêt pour l'agriculteur.

Les auteurs s'intéressent ensuite à l'application conjointe du *fine-tuning* et du *RAG*. Dans la mesure où les phrases des réponses générées seront rarement rigoureusement identiques aux phrases des réponses attendues, une méta-mesure de similarité classifiée en 3 catégories (Similaire, Plutôt similaire, Non-similaire) permet de mettre en évidence la similarité entre la réponse obtenue et la *groundtruth*. La mesure de l'*accuracy* est ensuite définie comme la proportion de similaires . Les résultats obtenus sont les suivants :

TABLE 1.1 – Performance des modèles de base et des modèles fine-tunés avec et sans RAG. source [5] - (table 21)

Modèle	**Performance : modèle seul (*accuracy* %)**	**Performance : modèle + RAG (*accuracy* %)**	**Evolution des performances avec RAG (pts)**
Llama-2-chat 13B	32%	49%	+17%
Vicuna	28%	56%	+28%
GPT-4	36%	60%	+24%
Llama2 13B fine-tuned	29%	49%	+20%
GPT-4 fine-tuned	45%	61%	+16%

Le tableau met en évidence plusieurs éléments :

— Pour chaque modèle, fine-tuné ou non, le RAG conduit à une augmentation de l'*accuracy* allant de +16% pour GPT-4 fine-tuned à +28% pour Vicuna.

— Le *fine-tuning* seul conduit à un gain de performance pour le modèle GPT-4 mais pas pour le modèle Llama-2-chat 13B.

— L'*accuracy* de {GPT4 + RAG} vaut 60%, surpassant ainsi GPT-4 simplement fine-tuné pour cette tâche relative à l'expertise agricole

En conclusion, sans être concurrentes chacune de ces approches est pertinente à différents égards et il ressort que leur conjonction, lorsqu'elle est justifiée, conduit à une amélioration des résultats. Le tableau comparatif suivant peut ainsi être établi :

TABLE 1.2 – Synthèse des avantages et inconvénients entre le RAG et le Fine-Tuning

	RAG (Génération Augmentée par Recherche)	**Fine-Tuning**
Avantages	— **Accès à des connaissances externes** : Permet d'accéder en temps réel à une vaste base de connaissances externes, offrant des réponses plus précises et adaptées au contexte. — **Efficacité** : Élimine la nécessité de réentraîner le modèle entier, ce qui économise du temps et des ressources informatiques. — **Flexibilité** : S'adapte dynamiquement à diverses requêtes et tâches sans nécessiter plusieurs modèles distincts. — **Coût initial faible** : Faible coût initial pour la création des *embeddings*.	— **Optimisation spécifique à la tâche** : Adapte le modèle à des tâches précises, améliorant ses performances pour des applications spécialisées et assurant un alignement optimal avec les exigences spécifiques. — **Généralisation robuste** : Le *fine-tuning* peut assurer une performance solide sur diverses instances d'une même tâche. — **Coût – taille des tokens d'entrée** : Minime. — **Coût – taille des tokens de sortie** : Précis, ajusté pour la brièveté.
Inconvénients	— **Coût – taille des tokens d'entrée** : Taille des prompts augmentée. — **Coût – taille des tokens de sortie** : Plus verbeux, plus difficile à guider. — **Complexité** : Requiert la gestion et l'interrogation d'une base de connaissances externe, ce qui augmente la complexité du système. — **Dépendance aux données externes** : La performance repose sur la qualité et la disponibilité des données externes.	— **Nécessite de grands ensembles de données spécifiques** : Requiert souvent de grandes quantités de données annotées pour la tâche spécifique — **Coûteux en ressources** : Exige des ressources informatiques avancées et un temps d'entraînement considérable, surtout pour les grands modèles. — **Coût initial élevé** : Le *fine-tuning* entraîne des coûts importants en ressources informatiques, création et traitement des données. Des jugements d'experts estiment un ordre de grandeur de 100 heures pour un LoRA et de 10K à 50K heures pour un alignement *Full Fine-Tuning*, pour un coût d'environ 4€/heure et par serveur. — **Risque de surapprentissage** : Peut réduire la capacité du modèle à généraliser.

1.5 Les défis et les limites

Ma revue de l'état de l'art a permis de clarifier certains points essentiels tels que l'impact de la multiplication des têtes d'attention dans ces modèles, ou encore les architectures permettant de contenir le coût computationnel en dépit de milliard de paramètres.
L'implémentation du modèle RAG dans la production d'agents conversationnels met quand à elle en lumière plusieurs défis majeurs liés à la nature complexe des données traitées.

Ces limites soulevées prédessinent un ensemble d'axe de contribution.

Premièrement, il est crucial de sortir du cadre purement théorique pour se plonger dans des projets pratiques. Comment pouvons-nous réellement évaluer l'employabilité des données dans des contextes variés ? Cela nécessite une approche pragmatique pour comprendre comment les données peuvent être utilisées efficacement dans des applications réelles.

La précédente interrogation fait ressortir la nécessite d'améliorer les solutions existantes pour, le traitement des données tabulaires notamment. Quelles méthodes actuelles sont réplicables ou susceptibles d'être optimisées pour mieux gérer ces types de données ? L'objectif est de trouver des solutions robustes et adaptables qui facilitent leur intégration et leur utilisation dans des systèmes complexes.

Enfin, dans le contexte du monde réel, comme implémenter un agent conversationnel en utilisant le RAG, et particulièrement si celui-ci a trait à des sujets sensibles, comment raffiner l'approche et ajuster le modèle pour qu'il puisse être à la hauteur des défis du traitement des sujets délicats et adopte une tonalité appropriée dans des scénarios du monde réel nécessitant de la nuance et pas juste une véracité comparativement au corpus ?

Contribution

2.1 Gestion de projet en IA et employabilité des données

Un premier pan de mon travail axé sur la gestion de projets d'intelligence artificielle avait pour objectif principal de contribuer à l'élaboration d'un référentiel dédié, destiné à être utilisé dans chaque mission menée par ALLONIA. Ce référentiel vise à standardiser les meilleures pratiques et à structurer l'approche de projet d'IA.

Cette expérience m'a conduite à approfondir plusieurs questions clés :

— La suffisance des données et la génération de données de synthèse, comme illustré par un *proof-of-concept* dans le domaine de l'assurance maritime, en partenariat avec le Hub France IA.

— La qualité des données et la gestion des données manquantes, particulièrement critiques dans le cadre d'un projet de rénovation de bâtiments assisté par IA pour un constructeur.

— La définition d'indicateurs de succès pour quantifier l'avancement des projets.

— La création d'indicateurs permettant de hiérarchiser les différentes approches selon des critères préétablis et d'orienter ainsi les décisions stratégiques.

Le processus a débuté par une revue approfondie de dix projets en cours, en m'appuyant sur les restitutions de réunions et les présentations archivées. Cette revue m'a permis de poser les bases du référentiel. Par la suite, j'ai été impliquée dans trois projets déjà initiés à mon arrivée, ce qui m'a offert une opportunité concrète d'affronter les défis liés à la gestion de projets d'IA générative pour plusieurs clients :

Client	**Description et conclusion**
Région Île-de-France	Conception d'un assistant virtuel à destination des Franciliens. Allonia a été choisie par la région Île-de-France pour développer une solution permettant aux Franciliens de se renseigner sur les aides dont ils peuvent bénéficier, par exemple en matière de rénovation énergétique ou de mobilité douce. La genèse de cette mission précédait mon arrivée ; toutefois, mon implication m'a permis d'acquérir une pleine maîtrise des différentes facettes, et j'ai eu l'opportunité de présenter la solution au salon VivaTech [1].
	Le corpus était constitué de documentation administrative existante. Un point d'attention était l'écart qu'il pouvait y avoir entre le jargon administratif et les termes susceptibles d'être employés par les utilisateurs cibles. Par exemple, là où un utilisateur s'intéresserait aux vélos électriques, les documents administratifs ne les mentionnent que sous le nom de véhicules à faible émission.
Meetrisk	Insurtech maritime accompagnée dans le cadre du Pack IA du Hub France IA. Utilisation de rapports de navigation et d'incidents pour établir une base de données structurées grâce aux LLMs, en vue d'une exploitation pour inférence et tarification d'assurance maritime.

Suite à la page suivante

1. Ma présentation à VivaTech a fait l'objet d'un article de presse disponible à cette adresse : `https://www.iledefrance.fr/toutes-les-actualites/des-ia-innovantes-100-franciliennes`

Client	Description et conclusion
	Le principal défi pour ce client était la rareté des données et sa capacité du client à les annoter. Un premier prototype a permis de montrer que les modèles pouvaient être utilisés efficacement pour extraire des données structurés des documents. Il n'a pas été possible de déboucher sur une solution commercialisable dans le temps imparti mais uniquement de fournir des outils pour des prestations de services.
RenovAIte	Consortium mené par le groupe VINCI qui propose de déployer l'IA pour la rénovation de bâtiments.
	Le principal obstacle était la difficulté d'accès à certaines informations, dont certaines nécessitaient des visites sur place, alors que le lot de logements précurseurs des solutions, soit 5000 logements pour débuter, s'étend sur une zone géographique incluant La Réunion, l'Occitanie et l'Île-de-France. Les méthodes d'inférence se sont ainsi révélées limitées par la proportion de données manquantes : sur les 300 variables par bâtiment, certaines n'étaient renseignées que pour 30% des bâtiments. Il a été nécessaire de restreindre la liste des variables employables et donc les applications possibles.

Ces projets m'ont confrontée à des attentes clients parfois en décalage avec les capacités de modélisation limitées par les données disponibles, mettant en lumière les problématiques de qualité et d'employabilité des données. Ces défis sont cruciaux pour la réussite des modèles IA implémentés, dont la performance dépend en grande partie de la qualité des données utilisées, comme expliqué dans la section sur les *transformers* (voir section 1.1).

À travers cette responsabilité, j'ai eu le privilège de pouvoir m'appuyer sur les retours d'expérience d'experts reconnus, notamment Françoise Soulié-Fogelman, avec qui j'ai collaboré dans le cadre de l'accompagnement de certains clients auprès du Hub France IA où elle exerce actuellement en tant que conseillère scientifique. Ces interactions ont considérablement élargi ma perspective. Ce faisant j'ai substantiellement enrichi le référentiel de gestion de projet en IA au fil des missions, en identifiant les zones d'ombre et en intégrant des questions et vérifications cruciales qui n'avaient pas été anticipées dans la version initiale, notamment la validation des données, la création d'un référentiel d'évaluation des *outputs* des modèles, ou encore l'adhérence à la réalité métier et aux exigences d'avoir une rentabilité concrète à exposer.

Ce travail m'a permis de développer une expertise concrète dans la gestion de projet appliquée à l'IA, et de comprendre ce qui peut faire défaut dans les projets industriels par rapport aux *"toy problems"* de la recherche.

L'expérience acquise met en lumière des points essentiels, notamment les contraintes spécifiques liées à la qualité et à la disponibilité des données, ainsi que la capacité à évaluer efficacement les solutions.

En outre, cette première implication a mis en évidence un axe d'amélioration prioritaire pour ALLONIA : le traitement des données tabulaires en entrée des modèles LLM. Cet axe vise à optimiser les outils existants et à étendre leur utilisation à un plus grand nombre de clients.

2.2 Etude de la vectorisation des données tabulaires en entrée des *LLM* en vue d'un enrichissement des outils existants

2.2.1 Problématique

Les fichiers souvent distribués au format PDF présentent une variété de mises en page, incluant des données textuelles séquentielles, tabulaires et visuelles. Ce format pose un défi important pour l'exploitation de l'information non seulement pour le contenu, mais surtout pour la structure sous-jacente qui organise ces informations. L'intégration des données tabulaires, comme des images de texte en entrée des *LLMs* revêt donc un intérêt particulier et pose des défis complexes.

L'ambition d'ALLONIA est de compléter son offre d'agents conversationnels pour permettre à l'utilisateur d'initialiser un agent conversationnel, charger un corpus de documents PDF, word ou CSV contenant ou non des tableaux de différentes tailles, ou encore des images de tableaux et de permettre à l'utilisateur d'utiliser ces éléments comme une mémoire externe dont se sert son agent conversationnel.
Ce projet conséquent nécessite de travailler de façon méthodique, et la première étape consiste à atteindre une maîtrise complète par l'agent conversationnel d'un corpus constitué d'un unique document comportant exclusivement un tableau. L'objectif est de permettre à l'agent de recourir à ce tableau comme mémoire externe dans 100% des cas, quel que soit le tableau.

Les outils de *LLMs* montrent actuellement des limites lorsqu'il s'agit de traiter certaines données tabulaires. Cette limitation est liée à plusieurs facteurs, parmi lesquels on peut citer l'architecture séquentielle qui est optimisée pour des données textuelles linéaires, ou encore comme le souligne [2] l'absence de biais inductifs qui rend difficile l'exploitation des concepts tels que les relations clés-valeurs par exemple, et qui limite la capacité des *LLMs* à comprendre les logiques sous-jacentes des données tabulaires. Dans le cas d'un tableau dont le nombre de lignes est de l'ordre de quelques dizaines et dont les colonnes ne sont pas imbriquées l'une dans l'autre les performances et la capacité du modèle RAG d'ALLONIA à répondre sont excellentes. Toutefois, ces situations idéales ne sont pas les seules auxquelles les modèles peuvent être confrontés.

Ainsi, à ce stade l'offre d'ALLONIA ne couvre pas encore la totalité des besoins éventuels des clients. À titre d'exemple, un rapport financier dont les tableaux comprennent souvent du contenu aux colonnes imbriquées peut mettre en difficulté les outils actuels d'ALLONIA. Pour citer un deuxième exemple, un tableau particulièrement volumineux, tel que la base de données des communes de France et de leur population composée de 35000 lignes et 6 colonnes met également en évidence les limites de la chaîne de RAG actuelle d'ALLONIA : selon la question, le nombre de *chunks* à récupérer nécessaire aux opérations peut être trop important pour que le modèle soit à même de répondre correctement.

Ainsi dans le cas des données tabulaires l'étape de segmentation du corpus est particulièrement délicate puisque selon la taille et la structure des tableaux la récupération des *chunks* similaires à la requête ne permet pas toujours de produire la réponse à la question.

2.2.2 Étude menée

Le sujet de la prise en charge des données tabulaires dans le contexte des agents conversationnels à mémoire externe est un domaine de recherche en plein développement. Il n'existe actuellement ni consensus ni outil populaire qui excelle dans les deux situations suivantes : l'analyse d'un tableau de type rapport financier et la gestion de tableaux particulièrement volumineux.

Une première revue de la littérature a permis de faire émerger deux approches spécifiques, chacune adaptée à des circonstances bien précises. Ces deux approches très différentes m'ont permise de considérer un large panel de solutions.

La première approche qui s'est démarquée est GROBID (GeneRation Of BIbliographic Data [9]), une bibliothèque d'apprentissage automatique conçue pour extraire et traiter des données de la littérature scientifique au format PDF. L'objectif est de transformer des données PDF non structurées en données structurées sous forme de format TEI (Text Encoding Initiative) (Consortium, 2023), permettant de gérer efficacement de gros volumes de fichiers. GROBID, formé sur un vaste corpus d'articles scientifiques, permet la reconnaissance d'un large éventail d'éléments documentaires et l'extraction de données bibliographiques associées. Les fichiers TEI générés par GROBID comprennent les métadonnées du document (titre, auteurs, résumé), les sections, les tableaux, les références aux figures, la bibliographie, ainsi que le contenu lui-même. L'objectif final est de convertir ces fichiers TEI en fichiers JSON, plus facilement exploitables, tout en préservant la structure et le contenu de l'original.

La seconde approche réside dans les fonctionnalités de la librairie LangChain [7], qui combine différents *agents* pour implémenter des solutions personnalisées. Cette bibliothèque qui est populaire et solidement ancrée dans l'IA générative, s'en trouve très fortement compatible avec l'architecture des solutions déjà proposées par ALLONIA.

Les deux approches sont très différentes, non-concurrentes et difficilement comparables, ce qui a été un atout pour permettre à l'analyse de ces deux approches de dégager des pistes prometteuses pour l'implémentation d'une solution adaptée aux besoins spécifiques de la gestion des données tabulaires dans le cadre des agents conversationnels.

2.2.3 Premières implémentations résultant de la recherche

Première piste

La première piste qui a fait l'objet d'une implémentation est l'intégration du tableau volumineux et de sa transposée dans la base de données vectorielle afin que tous deux soient segmentés et vectorisés.
Ainsi, si la question porte sur une colonne de 1000 lignes, en faisant l'hypothèse d'une dimension de 20 caractères par chunk, l'embedding de la transposée permet de travailler avec 50 *chunks* là où le tableau non transposé nécessiterait d'exploiter 1000 *chunks* ! Cet exemple simple illustre l'attrait pour cette méthode pour les tableaux volumineux et la motivation derrière son implémentation.

Observation après implémentation :
La solution implémentée est pertinente mais présente des limites en ce qu'elle ne permet pas d'automatiser de manière agnostique la transposition. En effet, il est nécessaire d'indiquer a minima le libellé de la colonne qui servira d'en-tête. Par exemple, pour les communes de France, il semble pertinent que les en-têtes de la transposée soient les noms des communes. Or, cela ne correspond pas nécessairement à la première colonne. Ainsi, cette solution nécessite une connaissance minimale

de la structure du tableau. Dans cette situation, une application massive sur des tableaux au contenu non connu n'est pas encore fonctionnelle, elle nécessiterait de demander à l'utilisateur des informations supplémentaires au moment de l'*upload* éventuellement. Les fonctionnalités de la librairie Langchain, notamment le composant *Langsmith*, pourraient être mises à contribution pour permettre de créer un agent pour aider dans ce sens.

Seconde piste

La seconde piste ayant fait l'objet d'une implémentation consiste à complexifier l'architecture de la solution existante en intégrant une pipeline qui gérerait les données via SQL. Dans le cas où le tableau est trop grand mais sans cellules imbriquées, un outil a été développé pour répondre à des questions en utilisant SQL pour effectuer les opérations de recherche ou de calcul de l'information recherchée, puis générer une phrase restituant la réponse à l'utilisateur. Dans ce paradigme, le choix est fait d'employer un agent qui fournit une "compétence" à la chaîne de RAG. Le tableau est détecté et intégré à une base de données ou à un DataFrame. Si la question concerne le tableau alors la requête est aussi faite au tableau via le *LLM* qui génère la requête, le résultat est réinjecté dans le contexte pour générer la réponse.

Observation après implémentation : Cette seconde approche produit d'excellents résultats dans l'exemple nucléaire simplifié choisi correspondant à un document unique ne contenant qu'un tableau. Elle permet de s'affranchir des difficultés liées au RAG en s'appuyant sur les performances éprouvées de SQL. Toutefois, son intégration dans la solution proposée par ALLONIA entraînerait une complexification de l'architecture du produit, un point négatif qui ne serait pas a priori insurmontable. Il faudrait en effet détecter les tableaux, les exclure, et mener des recherches tantôt entre la requête et la base de données vectorisée issue du RAG (par MIPS) pour certaines questions, et pour d'autres passer d'abord par une requête au sein de la donnée tabulaire via SQL, cela pourrait passer par la création de méta-données concernant le tableau.

TABLE 2.2 – Synthèse comparative des deux approches originales qui ont émergé

	Segmentation conjointe du tableau initial et de sa transposée	**Architecture intégrant une pipeline SQL pour traiter les opérations sur tableau.**
Avantages	Permet d'adapter la constitution des *chunks*, facilitant les opérations sur l'ensemble des colonnes ou des lignes	Élimine les difficultés liées au découpage des données, avec une méthode fiable grâce à l'identification efficace des colonnes d'intérêt.
Inconvénients	La solution actuelle n'est pas universelle, nécessitant que l'utilisateur fournisse des informations sur la structure du CSV ou du tableau, notamment la colonne à utiliser comme en-tête dans la version transposée.	L'intégration de cet outil dans l'architecture existante complexifie la solution proposée au client et nécessite d'implémenter une agentification du traitement des tableaux.

2.3 Réalisation de mission pour répondre aux besoins d'un client d'envergure : SPI

Pour synthétiser mon expérience et solidifier les compétences acquises j'ai eu l'opportunité d'appliquer ces principes à un cas concret et d'être positionnée dès la genèse d'une nouvelle mission en tant que cheffe de projet et data scientist. Cette mission est particulièrement originale puisque le client est une institution religieuse : le Service pour les Professionnels de l'Information du Diocèse de Paris (SPI).

Cette mission a été l'occasion de développer des solutions visant à être déployées chez le client et d'aborder toute la chaîne de la création d'outils d'IA générative. Elle a été particulièrement gratifiante puisque loin de se limiter aux seuls aspects de l'implémentation, j'ai pu jouer un rôle dans les échanges avec le client.

2.3.1 Contexte et objectif

Les acteurs de cette mission sont d'une part le Service pour les Professionnels de l'Information du Diocèse de Paris (SPI), représenté par deux interlocuteurs privilégiés, et d'autre part l'entreprise ALLONIA, avec une équipe opérationnelle de 3 personnes : un directeur de projet, un autre ingénieur et moi-même.

Le besoin du client SPI pour un agent conversationnel

SPI a approché ALLONIA avec une problématique originale, celle de développer un "Chat GPT pour l'Église catholique française", qui doit être présenté à la Conférence des Evêques de France en novembre 2024.
Cette demande est formulée dans un contexte de forte influence du contenu en langue anglaise dans l'entraînement des modèles, par une institution consciente de différences philosophiques entre la conception chrétienne catholique qu'elle représente et les conceptions chrétiennes d'autres courants (pentecôtiste, baptiste, protestant, anglican, etc.). Afin de se prémunir d'une éventuelle hégémonie de courants non-catholiques, le SPI souhaiterait développer un outil basé sur l'IA générative. Cet outil pourrait permettre au clergé d'avoir accès à une connaissance digitale centralisée qui fait références aux textes, il servirait également d'assistant pour certaines questions pendant le catéchisme afin d'apporter des réponses sourcées et nuancées qui ne se basent pas uniquement sur l'opinion de l'individu en charge, il jouerait un rôle de communication à l'attention des journalistes et pourrait permettre aux curieux de se renseigner.

Éléments sensibles à prendre en compte

Par ailleurs, un fort travail de *guardrailing* est nécessaire, puisque de nombreux sujets brûlants et d'actualité sont susceptibles de heurter le public : nommément les questions d'interruption de grossesse, de fin de vie, d'orientation romantique, de caractère 'subalterne' d'un des conjoints dans le contexte du mariage, ou encore de la légitimité des guerres et des crimes commis au nom du christianisme, pour n'en citer que quelques-uns. Il est important pour notre client que la solution développée soit à même de faire preuve de nuance et d'empathie, conformément à la position de l'Église, qui diffère actuellement sur de nombreux points de certains préceptes datés figurant dans des textes religieux.

Objectif pour ce client :

L'objectif est de développer un outil utilisant les technologies commercialisées par ALLONIA pour proposer des solutions taillées sur mesure et simples d'utilisation.
Une technologie basée sur le RAG a été considérée pour commencer. Cette approche permet de combiner des capacités de recherche avec des modèles de génération de texte, offrant ainsi des réponses plus pertinentes et contextualisées.

TABLE 2.3 – exemple de questions pouvant être posées par un utilisateur

1	Quelle est la position de l'Église concernant les personnes baptisées qui ont divorcé après un mariage religieux et qui sont remariées civilement ?
2	Que signifie le lundi de Pentecôte ?

Dans un deuxième temps, une évolution vers un *fine-tuning* de type LoRA (Low-Rank Adaptation) pourrait être envisagée. LoRA est une technique d'adaptation de modèle qui permet de fine-tuner des modèles de grande taille tout en réduisant considérablement les coûts de calcul et de stockage. Contrairement aux méthodes traditionnelles de *fine-tuning* qui nécessitent de recalibrer l'ensemble des paramètres du modèle, LoRA se concentre sur l'ajustement d'un sous-ensemble réduit de paramètres, appelés "matrices de faible rang". Cette approche permet de conserver la majorité des paramètres pré-entraînés intacts tout en adaptant le modèle aux spécificités du domaine ou des données cibles sans les coûts élevés associés au *fine-tuning* complet des grands modèles.

2.3.2 Architecture de la solution proposée

Données Les données du corpus comprennent le Catéchisme de l'Église Catholique (CEC), le compendium du CEC et la Doctrine Sociale de l'Eglise. Elles représentent un volume de 745 pages au format PDF A4 et 4,2 Go.

Aperçu de la solution D'un point de vue de l'interface utilisateur la solution se présente comme un agent conversationnel. En pratique il repose sur l'implémentation d'un modèle *RAG* et sur les données qui ont été chargées.

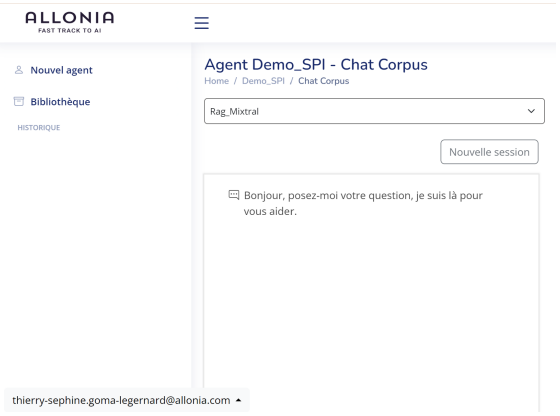


FIGURE 2.1 – Aperçu de l'interface finale après implémentation de l'agent conversationnel

2.3.3 Paramètres

Impact des paramètres de la segmentation Mon travail a permis de mettre en évidence qu'un *chunking* qui exploite les composants structuraux des documents offre des gains de performance pour la tâche de Q&A assistée par RAG. Une recherche révèle que les résultats observés sur nos données sont cohérents avec les conclusions présentées par Yepes et al. (2024)[22]. Nous n'avons pas noté de différence significative des résultats obtenus en faisant varier les paramètres clés de la segmentation.

Résultats : Évaluation des performances de l'assistant virtuel SPI

J'ai détaillé dans les sections précédentes l'implémentation et le fonctionnement de l'assistant virtuel conçu pour le *Service pour les Professionnels de l'Information* (SPI) du Diocèse de Paris. Je présente dans cette section les méthodes employées pour évaluer ses performances ainsi que les résultats obtenus.

3.1 Méthode d'évaluation de la performance du LLM développé

L'évaluation de la performance de la solution développé revêt un caractère crucial dans le cadre de la gestion d'un projet d'IA, puisqu'il est susceptible d'orienter les décisions stratégiques. Dans le cadre de la mission SPI, l'évaluation de la capacité de l'agent implémenté à répondre aux questions qui lui sont posées nécessite de définir des mesures de performances, quantitatives et qualitatives, pour évaluer la qualité des réponses en tenant compte du contexte, des données fournies et des réponses générées par le modèle.
La tâche évaluée correspond au *question answering*, où les réponses attendues et obtenues peuvent être de longues phrases, voire des paragraphes.

Pour sélectionner une liste de métriques cohérentes et scientifiquement fondées, je me suis appuyée sur la littérature existante, tel que les *benchmarks* employés pour mesurer les performances de différents modèles.

À la lumière des publications relatives à l'évaluation des tâches de *question answering*, les critères que j'ai retenus sont les suivants :

- **Non-hallucination** : Ce critère se concentre spécifiquement sur l'absence d'informations inventées ou erronées dans la réponse générée par le modèle. Une hallucination se produit lorsque le modèle génère du contenu qui n'est pas supporté par les données d'entrée ou qui n'existe pas dans le texte source ou dans les connaissances disponibles.
- **Fidélité (*faithfulness*)** : La fidélité indique la capacité d'une réponse à refléter avec exactitude les informations provenant des sources de référence, sans introduire de fausses informations, d'interprétations erronées ou d'inventions. Une réponse est considérée comme fidèle si elle est véridique et cohérente par rapport aux données d'entrée ou à la question initiale. À ce titre, il est compréhensible que la non-hallucination soit incluse dans la fidélité : si le modèle introduit des hallucinations dans les réponses, la fidélité ne peut être assurée.
- **BERTScore** : Cette mesure, formalisée en 2020 [23], propose d'utiliser les *embeddings* du modèle BERT pour comparer les similarités sémantiques contextuelles entre la réponse générée et la réponse attendue. Il est à noter que cette mesure peut indirectement évaluer la fidélité.

Il peut être noté que dans le cadre de la mise en œuvre du modèle *RAG*, de nouvelles mesures m'ont parues pertinentes, notamment une *Retrieval Precision*, qui évaluerait la précision du processus de récupération des documents dans le corpus fourni. Cette précision pourrait être déterminée par le rapport entre le nombre de documents pertinents récupérés et le nombre total de documents récupérés. Toutefois, j'ai choisi de ne pas inclure ce quatrième critère dans l'évaluation de la tâche de *question answering*, préférant me concentrer sur la qualité des réponses.

Ayant mis en évidence l'inclusion de la non-hallucination dans la fidélité, deux mesures sont retenues :

— la **fidélité (*faithfulness*)**

— le **BERTScore**

Pour évaluer la fidélité, j'ai retenu la méthode d'évaluation par un LLM juge, adoptant ainsi une approche similaire à celle de Prometheus2[12].
Pour évaluer le BERTScore, l'implémentation nécessite de récupérer la couche d'*embedding* d'un modèle BERT (*Bidirectional Encoder Representations from Transformers*) pré-entraîné, permettant une conversion vectorielle de la réponse de référence et de la réponse générée par le modèle. Ensuite, une distance entre les deux réponses est calculée : elle vaut $1 - \text{cosine\_similarity}(y\_pred, y)$: plus la distance est faible, meilleure est la performance du modèle pour la tâche de *question answering*.

3.2 Données nécessaires à l'évaluation

L'évaluation des résultats nécessite une base de données contenant des paires de questions-réponses (x, y), où y correspond à la réponse attendue. Pour chaque paire, la réponse de référence y est comparée à la réponse fournie par le *LLM*, notée y_pred ou $\hat{y}$, et la distance entre ces deux éléments est mesurée.

Pour cette tâche, nous disposons d'un document correspondant au Compendium du Catéchisme de l'Église Catholique. Ce document n'a été utilisé ni au sein du corpus fourni pour la tâche de *RAG*, ni pour le *fine-tuning* du modèle. Il contient 598 paires de questions-réponses (x, y).

Un échantillon a été tiré parmi ces 598 paires pour servir de base à l'évaluation des performances du modèle.

3.3 Présentation des performances du modèle

Mon approche d'évaluation a consisté à sélectionner aléatoirement 150 questions issues du corpus décrit précédemment, et à mesurer les performances du modèle implémenté sur cet échantillon. Il est à souligner que les réponses attendues ont été revues, parfois reformulées et amendées par un expert en théologie, pour cadrer à l'attente du SPI. Cet expert a aussi procédé à une classification des question selon 3 niveaux :

— Niveau 1 : les "qui et les quoi", des questions sans beaucoup de discussion quant à la réponse. Exemple : *Qui est Joseph ?* Ou encore : *Qu'est-ce que le canon des Écritures ?*

— Niveau 2 : les questions plus complexes qui demandent d'agréger plusieurs sources du corpus ou se réfèrent à des concepts abstraits. ex : *Pourquoi professer un seul Dieu ?*

— Niveau 3 : des questions ouvertes et philosophiques. ex : *Quel rapport existe-t-il entre liberté et responsabilité ?*

Les résultats obtenus pour la *faithfulness* sont les suivants :

TABLE 3.1 – Évaluation de la fidélité du modèle d'agent conversationnel

Fidélité/ Faithfulness	1	2	3	4	5	Valeur médiane
%	19%	12%	13%	23%	33%	**4**

Les résultats obtenus pour la mesure de la *faithfulness* à l'aide de l'approche basée sur un LLM juge révèlent que 69 % des réponses reçoivent un score égal ou supérieur à 3 sur une échelle de 1 à 5. Selon le LLM juge, nos résultats sont corrects et très encourageants, avec une valeur médiane de 4 sur 5 et une majorité des réponses classées à 5 sur 5.
Cette méthode repose sur le prompt engineering, et il a été observé que les performances sont fortement sensibles à la formulation du prompt. Afin de compléter cette approche, je propose d'intégrer également une mesure quantitative plus explicable : le BERTScore.

Les résultats obtenus pour le BERTScore sont les suivants :

TABLE 3.2 – Mesure du BERTScore du modèle d'agent conversationnel

Niveau attribuée à la question	**RMSE du BERTScore**
Niveau 1	0.1496
Niveau 2	0.0618
Niveau 3	0.1035
Tous	0.1225

La mesure de la RMSE, qui quantifie l'écart quadratique moyen des distances entre les réponses attendues et les réponses proposées par le modèle que j'ai implémenté, indique un bon alignement global, avec une similarité sémantique estimée à environ 88%. De façon assez contre-intuitive, c'est sur les questions de niveau 1 que nous notons l'écart le plus important la RMSE de la distance vaut (14,96%). Ceci peut être dû à la verbosité du modèle, qui fournit des réponses parfois longues, là où, par définition, les questions de niveau 1 sont celles qui appellent des réponses succinctes et non soumises à l'interprétation.
Les performances du modèle, à ce stade, sont déjà très satisfaisantes sur la base des mesures d'évaluation retenues.

3.4 Perspectives d'enrichissement du modèle

Le modèle développé pour SPI a vocation à être employé par différents acteurs. À court terme, il ne devrait être utilisé qu'en interne au sein du diocèse de Paris, par des membres du clergé ou dans le cadre de la cathéchèse, afin de les accompagner dans l'enseignement religieux. À moyen terme, ce modèle pourrait être ouvert à un public plus large, notamment les journalistes, à qui il servirait d'outil de documentation, par exemple à l'occasion de fêtes religieuses ou d'événements spécifiques, ainsi qu'à des particuliers simplement curieux.

Dans la perspective d'une ouverture au public, il est crucial de mettre en place des garde-fous robustes, garantissant non seulement des performances quantitatives élevées, mais aussi des ajustements qualitatifs. En effet, il est impératif d'éviter une tonalité trop rigide ou orthodoxe, qui, bien que cohérente avec le corpus, pourrait être en contradiction avec les valeurs de compassion et de souplesse défendues par le SPI, notamment sur des sujets sociétaux sensibles. À ce stade, l'intégration d'un pré-prompt commun a permis d'atténuer efficacement ce problème de tonalité, les résultats obtenus sont très satisfaisants et montrent que les garde-fous sont correctement implémentés.

Néanmoins, même si les résultats sont encourageants, la solution peut encore être affinée. À titre informatif, les pistes suivantes sont envisagées :

— Poursuivre le *red-teaming* afin de mettre en évidence les failles non identifiées à ce jour. À cet égard, les techniques d'attaque récemment partagées par NVIDIA dans le cadre de la conférence *Black Hat USA* d'août 2024 [10] constituent une source précieuse.

— Traitement des données sensibles : Introduire une étape de classification des sujets ou de détection d'intention avant la génération de la réponse. Cette approche permettrait de remplacer l'utilisation d'un pré-prompt unique par des pré-prompts spécifiques, adaptés à chaque catégorie de question. La bibliothèque LangChain pourrait être exploitée pour paramétrer des agents accomplissant cette tâche de manière opérationnelle.

— Démarcation par rapport aux LLMs existants : Deux options d'évolution pour le modèle sont envisagées. La première consiste en un apprentissage partiel avec *fine-tuning* de type LoRA (Low-Rank Adaptation), qui réduit le nombre de paramètres ajustables en ne modifiant que certaines matrices à faible rang, la seconde propose le développement et l'entraînement d'un modèle entièrement depuis le début, avec une architecture potentiellement inspirée par des modèles comme Artic de Snowflakes.

Conclusion

Ce stage a été une excellente expérience pour moi car il m'a permis de mener des travaux pluridisciplinaires, s'appuyant sur mes compétences techniques et sur mon appétence à l'application de mon bagage mathématique à la résolution de problème au contact de clients.

La revue de l'état de l'art que j'ai menée m'a permis d'exercer ma capacité à apporter un regard critique sur les conclusions de papiers de recherche et à soulever différents points sur des aspects spécifiques des modèles présentés afin de parfaire une compréhension d'aspects complexes, en particulier sur l'impact du nombre de têtes du modèle Transformer. Certains aspect pourraient faire l'objet d'études complémentaires, notamment la question de la preuve de l'impossibilité de la redondance de l'information entre les têtes qui reste à éclaircir, ainsi que le sujet d'une démonstration mathématique de la transférabilité des *embeddings*

Enfin, ce stage a été l'occasion de monter en compétences sur l'implémentation d'outil de RAG et les librairies qui permettent de développer des assistants virtuels s'appuyant sur les LLMs, ainsi que de bâtir un modèle pertinent qui, bien que pouvant encore être affiné, présente à ce jour d'excellents résultats.

Par ailleurs, sur le plan personnel cette expérience m'a beaucoup apporté, elle m'a permis de satisfaire mon objectif de prendre part aux missions d'IA non pas uniquement en tant que data scientist qui conçoit des solutions et produit du code, mais en ayant également un regard de chef de projet. J'ai pu enrichir ma connaissance concrète au contact de personnes expérimentées, mathématiciens de formation qui évoluent actuellement dans des postes de direction de projet, et rencontrer des personnes intéressantes, à travers des événements comme le R.AI.SE Summit, le salon Vivatech ou encore les collaborations avec le consortium RenovAIte et le Hub France IA.

Bibliographie

[1] Joshua Ainslie et al. *GQA : Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. arXiv : 2305.13245 [cs.CL].

[2] Sercan O. Arik et Tomas Pfister. *TabNet : Attentive Interpretable Tabular Learning*. 2020. arXiv : 1908.07442 [cs.LG]. URL : https://arxiv.org/abs/1908.07442.

[3] D.S. Asudani, N.K. Nagwani et P. Singh. « Impact of word embedding models on text analytics in deep learning environment : a review ». In : *Artificial Intelligence Review* 56 (sept. 2023), p. 10345-10425. DOI : 10.1007/s10462-023-10419-1. URL : https://doi.org/10.1007/s10462-023-10419-1.

[4] Dzmitry Bahdanau, Kyunghyun Cho et Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv : 1409.0473 [cs.CL]. URL : https://arxiv.org/abs/1409.0473.

[5] Angels Balaguer et al. « RAG vs Fine-tuning : Pipelines, Tradeoffs, and a Case Study on Agriculture ». In : (2024). arXiv : 2401.08406 [cs.CL].

[6] Rewon Child et al. « Generating Long Sequences with Sparse Transformers ». In : *CoRR* abs/1904.10509 (2019). arXiv : 1904.10509. URL : http://arxiv.org/abs/1904.10509.

[7] Langchain A Library for Developing Applications Powered by Language Models. *Langchain Tabular Question Answering Tools*. Version 0.1. Accessed : 2024-08-29. 2024. URL : https://github.com/hwchase17/langchain.

[8] Jacob Devlin et al. *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv : 1810.04805 [cs.CL]. URL : https://arxiv.org/abs/1810.04805.

[9] *GROBID*. https://github.com/kermitt2/grobid. 2008–2023. swh : 1 : dir : dab86b296e3c3216e2241968f0d63b68e8209d3c.

[10] Richard Harang. *Practical LLM Security : Takeaways From a Year in the Trenches*. BlackHat USA 2024, 94 pages. Août 2024.

[11] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv : 2310.06825 [cs.CL].

[12] Seungone Kim et al. *Prometheus 2 : An Open Source Language Model Specialized in Evaluating Other Language Models*. 2024. arXiv : 2405.01535 [cs.CL]. URL : https://arxiv.org/abs/2405.01535.

[13] Olga Kovaleva et al. *Revealing the Dark Secrets of BERT*. 2019. arXiv : 1908.08593 [cs.CL]. URL : https://arxiv.org/abs/1908.08593.

[14] Mike Lewis et al. *BART : Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv : 1910.13461 [cs.CL]. URL : https://arxiv.org/abs/1910.13461.

[15] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv : 2005.11401 [cs.CL].

[16] Paul Michel, Omer Levy et Graham Neubig. *Are Sixteen Heads Really Better than One ?* 2019. arXiv : 1905.10650 [cs.CL]. URL : https://arxiv.org/abs/1905.10650.

[17] Microsoft. *NLP Recipes*. https://github.com/microsoft/nlp-recipes. Public archive. 2023.

[18] Mohammed Muqeeth, Haokun Liu et Colin Raffel. *Soft Merging of Experts with Adaptive Routing*. 2024. arXiv : 2306.03745 [cs.LG]. URL : https://arxiv.org/abs/2306.03745.

[19] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv : 1910.10683 [cs.LG]. URL : https://arxiv.org/abs/1910.10683.

[20] Noam Shazeer et al. *Outrageously Large Neural Networks : The Sparsely-Gated Mixture-of-Experts Layer*. 2017. arXiv : 1701.06538 [cs.LG]. URL : https://arxiv.org/abs/1701.06538.

[21] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv : 1706.03762 [cs.CL].

[22] Antonio Jimeno Yepes et al. *Financial Report Chunking for Effective Retrieval Augmented Generation*. 2024. arXiv : 2402.05131 [cs.CL]. URL : https://arxiv.org/abs/2402.05131.

[23] Tianyi Zhang et al. *BERTScore : Evaluating Text Generation with BERT*. 2020. arXiv : 1904.09675 [cs.CL]. URL : https://arxiv.org/abs/1904.09675.