UNIVERSITÉ PARIS-DAUPHINE – PSL ET INSA ROUEN

Double diplôme Master Ingénierie Statistique et Financière (Université Paris Dauphine) et Ingénierie Mathématiques Appliquées (INSA Rouen)

Mémoire de stage de fin d'études

Amélioration d'un pipeline de Retrieval-Augmented Generation pour un chatbot d'assurance

Stage effectué au sein de :
Allianz France – Direction Big Data and AI
Paris La Défense

Période : Avril 2025 – Octobre 2025

Réalisé par :
LOUZI Idriss

Encadrants :
Tuteur en entreprise : COUSIN Louis – Data Scientist, Allianz France
Tuteur en entreprise : FRACSO Samy – Lead Data Scientist, Allianz France
Tuteur académique : RIVOIRARD Vincent – Université Paris-Dauphine
Tuteur académique : FORCADEL Nicolas – INSA Rouen

Remerciements

Je souhaite tout d'abord exprimer ma profonde gratitude à Louis Cousin, mon tuteur de stage chez Allianz, pour son accompagnement constant, son encadrement attentif et le temps consacré à la supervision de mes travaux.
Je remercie également Samy Fracso pour sa disponibilité et ses conseils, ainsi que l'ensemble de l'équipe Communications pour leur accueil chaleureux, leur accompagnement et leurs échanges constructifs tout au long du stage.
Mes remerciements s'adressent aussi à l'équipe Big Data pour son accueil, aux autres stagiaires, ainsi qu'à Stéphane Ollivier pour son suivi et son soutien durant cette expérience.
Je tiens également à remercier l'INSA Rouen Normandie et l'Université Paris Dauphine – PSL pour les enseignements théoriques et humains que j'ai pu y acquérir, qui ont constitué une base solide pour la réalisation de ce stage.
Je tiens aussi à remercier Katia Meziani pour l'acompagnement qu'elle a pû dispenser tout au long de l'année ainsi que sa disponibilité et ses conseils.
Je remercie enfin mes tuteurs académiques, Vincent Rivoirard à l'Université Paris Dauphine et Nicolas Forcadel à INSA Rouen, pour leur disponibilité et leurs réponses toujours précises à mes questions.
Enfin, des remerciements particuliers vont à ma mère, à ma sœur et à l'ensemble de ma famille pour leur soutien moral et leur encouragement constant tout au long de ce parcours.

Résumé

Ce mémoire présente les travaux réalisés dans le cadre de mon stage de fin d'études chez Allianz, portant sur l'amélioration d'un pipeline de *Retrieval-Augmented Generation* (RAG). L'objectif principal est l'amélioration de la qualité des réponses d'un chatbot interne, à travers la combinaison de plusieurs approches issues de l'état de l'art en intelligence artificielle et en recherche d'information. Des graphes d'information ont été exploités afin d'optimiser la récupération de contenus pertinents par les modèles de langage. Un jeu de données annoté a été conçu pour l'entraînement d'un modèle RoBERTa chargé d'évaluer la pertinence des passages récupérés. Ce modèle constitue la base d'un futur entraînement par *Group Relative Policy Optimization* (GRPO) visant à améliorer le reranking. En complément, une API de traitement documentaire et un système d'envoi automatique de KPI ont été développés afin de faciliter le suivi du pipeline. Ces travaux contribuent à renforcer la qualité et la modularité du système tout en préparant l'intégration de méthodes d'apprentissage par renforcement.

Mots-clés : Intelligence artificielle, RAG, graphe de connaissances, recherche hybride, reranking, BERT, RoBERTa, GRPO

Abstract

This thesis presents the work conducted during my final-year internship at Allianz, focusing on the improvement of a *Retrieval-Augmented Generation* (RAG) pipeline. The main objective is to enhance the quality of responses produced by an internal chatbot by combining several state-of-the-art approaches in artificial intelligence and information retrieval. Knowledge graphs were leveraged to improve the retrieval of relevant information by large language models (LLMs). An annotated dataset was created to train a BERT-based model responsible for assessing the relevance of retrieved passages. This model serves as a foundation for future training using *Group Relative Policy Optimization* (GRPO) to refine reranking. Additionally, a document processing API and an automated KPI delivery system were developed to facilitate pipeline monitoring. The work carried out strengthens the system's robustness and modularity while preparing for the integration of reinforcement learning methods.

Keywords: Artificial Intelligence, RAG, knowledge graph, hybrid search, reranking, BERT, RoBERTa, GRPO

Contents

1 Lexique

- RAG: Retrieval Augmented Generation
- LLM: Large Language Model
- ABFS: Azure Blob File Storage
- RL: Reinforcement Learning
- GRPO: Group Relative Policy Optimization
- API: Application Programming Interface
- OCR: Optical Character Recognition
- BERT: Bidirectional Encoder Reprensentations from Transformers
- RoBERTa: Robustly optimized BERT approach
- KPI: Key Progress Indicator

2 Introduction

L'IA générative prend de plus en plus de place dans notre quotidien, ainsi que dans celui des entreprises. Afin de tirer partie de ces avancées technologiques, de nombreuses organisations cherchent à intégrer de l'IA générative dans leurs processus métiers, notamment pour automatiser des tâches destinées aux collaborateurs, les aider dans leur quotidien et pour améliorer l'expérience client. Cependant, malgré des performances remarquables, l'IA générative, sous la forme de modèles de langages, présente une limite majeure: ces modèles ne possèdent pas un accès direct à des données spécifiques ou récentes, rendant leurs réponses imprécises ou inexactes lorsqu'ils sont confrontés à ce type d'informations. Les organisations ont alors intégré un ensemble d'architectures permettant de s'affranchir de cette limitation des modèles de langages. Parmi ces pratiques, le RAG (Retrieval Augmented Generation) reste la plus populaire, de par les résultats qu'elle a sû démontrer, au prix d'un coût de calcul très faible.
Allianz, un des leaders mondiaux du secteur de l'assurance, s'inscrit également dans cette dynamique d'innovation. C'est pour cela qu'Allianz en France en particulier dispose d'une équipe nommée Data et IA, composée de plus de 60 data scientists, data engineers et développeurs, spécialisée dans le développement et le déploiement de solutions variées orientées Data Science et intelligence artificielle. C'est dans cette équipe et plus précisément au sein de l'équipe Communication, qui cherche à développer des solutions conversationnelles internes fiables destinées aux agents franchisés, que le stage a été effectué.
L'équipe Communication développe actuellement un chatbot destiné aux agents franchisés, sous le nom d'Ask-Me. Derrière ce chatbot réside l'architecture de type RAG évoquée précédemment. La problématique centrale de ce mémoire peut alors être formulée ainsi:

Comment améliorer les performances d'une pipeline RAG afin d'optimiser la qualité des réponses générées par un chatbot d'assurance ?

Afin de répondre à cette problématique, les travaux réalisés durant le stage et présentés dans ce mémoire s'articulent autour de quatre axes majeurs:

- **Industrialisation du traitement documentaire** : conception d'une API REST permettant de rendre le processus de découpage (*chunking*) de documents accessible en tant que service, facilitant ainsi sa réutilisation par d'autres équipes ;
- **Amélioration du retrieval** : construction de graphes de connaissances reliant les chunks selon leur proximité sémantique, la détection de communautés par l'algorithme SLPA (Speaker-Listener Label Propagation Algorithm), et l'ajout de résumés de communautés comme métadonnées contextuelles ;
- **Optimisation du reranking** : expérimentation d'une méthode de reinforcement learning nommée Group Relative Policy Optimization (GRPO), permettant d'ajuster les scores de pertinence du reranker à partir d'un signal de récompense relatif ;
- **Automatisation du suivi des performances** : mise en place d'un système d'envoi automatique hebdomadaire de KPI (Key Performance Indicators) pour assurer un suivi continu de la qualité du pipeline.

3 Travaux réalisés durant le stage

Tous les travaux nécessaires à la réponse de la problématique et listés dans le plan de réponse en introduction ont été réalisés. Également, les travaux listés dans la convention de stage ont tous été effectués à l'exception du benchmarking de modèles et de l'exploration de l'utilisation d'Agents IA pour permettre à un utilisateur de charger un document externe au corpus documentaire d'Allianz et de l'interroger. Ces tâches n'ont pas été réalisées car jugées non prioritaires dans le planning des tâches à réaliser par l'équipe.

4 Présentation de l'entreprise et de l'équipe

4.1 Le groupe Allianz

Le groupe Allianz a été fondé en 1890 à Munich par Carl Von Thiem et Wilhelm Finck. Dès sa création, l'entreprise s'est positionnée comme un acteur innovant dans le secteur de l'assurance. Elle a débuté avec des produits d'assurance liés au transport et aux accidents puis s'est rapidement diversifiée avec des offres couvrant les incendies. Aujourd'hui, Allianz est présent dans plus de 70 pays, compte 155 000 collaborateurs et plus de 126 millions de clients dans le monde, avec un chiffre d'affaires à 179.8 milliards d'euros. Allianz opère dans plusieurs domaines d'activité:

- Assurance: offre complète pour les particuliers et les entreprises couvrant un large éventail de besoins, allant de l'assurance automobile, habitation et santé à des produits plus spécialisés tels que l'assurance vie ou l'assurance dommages aux biens
- Allianz Global Corporate and Specialty (AGCS) : spécialisée dans les assurances industrielles et grands risques
- Allianz Trade : leader mondial de l'assurance crédit, aide les entreprises à se prémunir contre les impayés
- Gestion d'actifs : via Allianz Global Investors (AGI), le groupe propose des solutions d'investissement durables et performantes
- Assistance : par l'intermédiaire d'Allianz Partners, la société intervient dans les domaines de la mobilité, santé, voyage et services aux particuliers.
- Investissement immobilier : avec PIMCO Prime Real Estate

4.2 Allianz en France

Allianz en France est organisée autour de huit grandes unités:

- Assurances de bien
- Assurances de personnes
- Distribution
- Finance
- Investissement
- Service client
- Transformation
- Data, Engagement, Marketing, Stratégie (UDEMS)

C'est dans cette dernière, qui pilote notamment la stratégie de marque, la connaissance client et les projets Data, que le stage a été effectué.

La direction Leads et Data, intégrée à l'UDEMS, est chargée de la génération de leads commerciaux et de la manipulation et l'analyse de données clients. Elle se divise en quatre sous-directions: Leads, Analytique client et distribution, Parcours client et enfin Big Data et AI.
Cette dernière sous-direction a pour objectif de valoriser les données au service des référents métiers via des projets d'intelligence artificielle et de data science. C'est là que s'est déroulé le stage.
Cette sous-direction utilise la méthode Agile dans la livraison des différents projets. Les équipes la constituant sont aussi appelées "scrums".

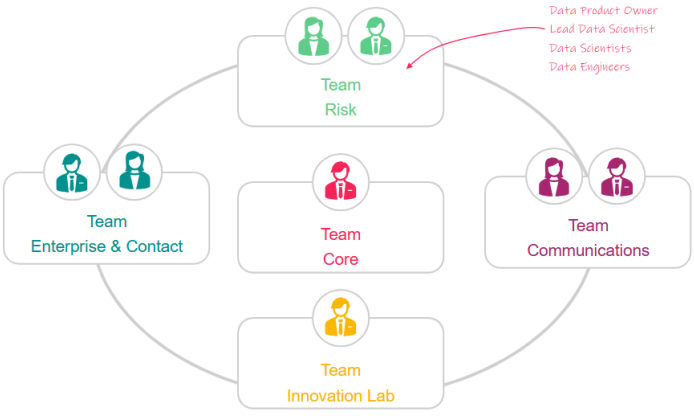


Figure 1: Organisation de l'équipe Big Data and AI

Comme nous pouvons le voir dans la figure précédente, la sous-direction Big Data and AI est composée de 5 équipes:

- Le scrum Core: chargé du développement et de la maintenance des systèmes techniques centraux
- Le scrum Risk: qui veille à la gestion, l'analyse et la modélisation des risques liés à l'activité d'Allianz France
- Le scrum Entreprise: qui pilote les solutions technologiques à destination des clients professionnels
- Le scrum Innovation Lab: dédié à l'exploration et à l'expérimentation de nouvelles technologies
- Le scrum Communications: composé de 11 data scinetists et 2 data engineers, est responsable du développement de solutions IA et data orientées communication avec les clients. C'est là qu'à été effectué le stage.

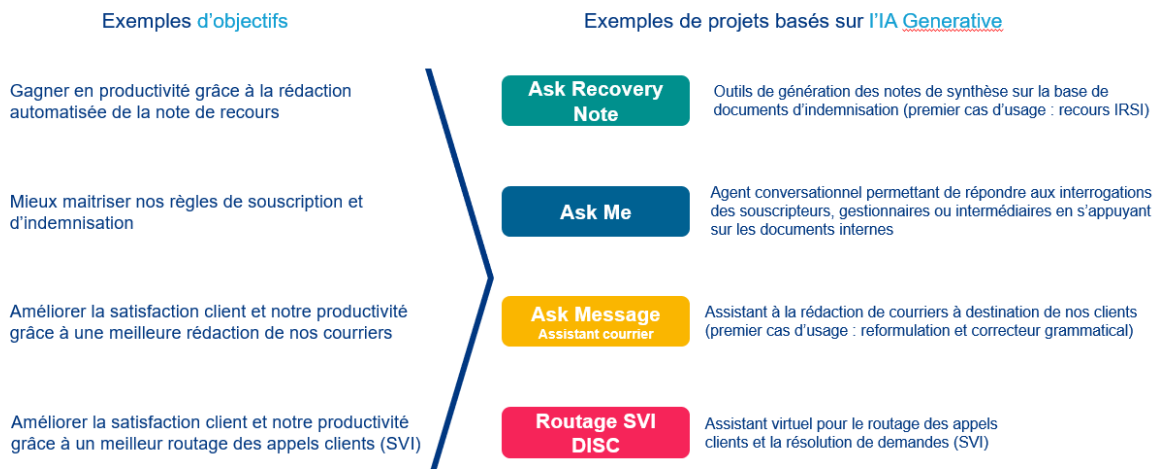


Figure 2: Projets sous la responsabilité du scrum Communications

Le projet sur lequel auront lieu les interventions du stage est le projet Ask-Me, qui est un chatbot destiné aux agents.
Un agent (ou agent général d'assurance) est un intermédiaire indépendant qui représente Allianz auprès des clients afin de les conseiller, leur proposer des contrats d'assurance ou gérer leur portefeuilles.
Ask-Me a alors pour objectif d'aider ces agents dans leur prise de décision quant aux différents produits d'assurances d'Allianz.
Le chatbot sera présenté plus en détail dans la section Environnement de travail et stack technologique.

5 Contexte scientifique et état de l'art

Dans cette partie seront fournies les différentes briques scientifiques nécessaires à la compréhension du travail effectué.

5.1 IA générative et modèles de langage larges

Les LLM (Large Language Models) sont des modèles de deep learning entraînés sur de très grands corpus textuels dans le but de comprendre et générer du langage naturel. Ils représentent la pierre angulaire des applications modernes d'IA générative: chatbots, assistants virtuels, traduction, résumé, génération de code, etc. Leur entraînement est généralement non supervisé sur la plupart des données (pré-entraînement), puis éventuellement affiné (finetuning) pour des tâches spécifiques. Avant d'être traités par un modèle de langage, les textes doivent être convertis en une forme numérique compréhensible par le réseau de neurones. Cette transformation repose sur plusieurs étapes fondamentales:

- **La tokenisation** consiste à découper un texte en unités élémentaires appelées **tokens**. Un token peut correspondre à un mot, une sous-partie de mot ou même un caractère selon la méthode de segmentation utilisée (par exemple, *Byte Pair Encoding* ou *WordPiece*). Cette étape permet de gérer efficacement les langues à vocabulaire large et d'éviter les problèmes liés aux mots rares ou inconnus.

- Chaque token est ensuite converti en un vecteur numérique dense appelé **embedding**. Ces embeddings capturent la signification sémantique des tokens où la proximité entre vecteurs reflète la similarité de sens entre les mots correspondants. Ainsi, deux mots ayant des significations proches auront des représentations vectorielles similaires.

La majorité des LLM repose sur une architecture appelée Transformer [Vaswani et al., 2017] qui a profondément impacté le traitement du langage naturel en remplaçant les architectures séquentielles classiques par un mécanisme d'attention permettant de capturer efficacement les dépendances à longue portée entre éléments de la séquence (tokens). Voyons de plus près de quoi il s'agit

5.1.1 Le méchanisme d'attention

Les modèles séquentiels présentent une difficulté majeure : ils doivent encoder l'intégralité d'une séquence source dans un vecteur de dimension fixe. Cela conduit souvent à une perte d'information lorsque la séquence est longue. Le mécanisme d'attention, introduit par [Bahdanau et al., 2014] puis généralisé dans les Transformers, permet de pallier cette limitation. L'idée centrale est que, pour générer un élément de sortie, le modèle n'a pas besoin de se baser sur l'intégralité de la séquence source uniformément, mais peut se concentrer davantage sur certaines parties pertinentes.
Concrètement, l'attention calcule une combinaison pondérée des représentations de la séquence source, où les poids reflètent l'importance relative de chaque élément.

5.1.2 Définition formelle

Représentation mathématique. On considère une séquence d'entrée composée de n éléments (ou tokens), représentés sous forme de vecteurs dits d'embedding regroupés dans

la matrice :

$$X = [x_1, x_2, \ldots, x_n] \in R^{n \times d_{model}},$$

où d_{model} désigne la dimension du vecteur d'embedding.

À partir de cette matrice X, le modèle calcule trois projections linéaires distinctes à l'aide de matrices de poids apprises durant l'entraînement :

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

où :

- $Q \in R^{n \times d_k}$ représente les **requêtes (queries)**, c'est-à-dire ce que chaque token cherche à comprendre dans le contexte ;
- $K \in R^{n \times d_k}$ représente les **clés (keys)**, qui décrivent ce que chaque token peut "offrir" comme information ;
- $V \in R^{n \times d_v}$ représente les **valeurs (values)**, c'est-à-dire les informations transportées par chaque token.

Ces matrices de projection W_Q, W_K et W_V sont des paramètres du modèle, appris automatiquement pendant la phase d'entraînement, sans aucune intervention humaine.

Calcul des poids d'attention. Pour chaque paire *(requête, clé)*, on calcule un score de similarité mesurant leur compatibilité via un produit scalaire. On obtient ainsi la matrice des scores :

$$S = \frac{QK^\top}{\sqrt{d_k}} \quad \in R^{n \times n},$$

où le facteur $\sqrt{d_k}$ sert à normaliser les valeurs et à stabiliser l'entraînement.

Ces scores sont ensuite transformés en probabilités à l'aide de la fonction *softmax*, définie pour chaque ligne i et colonne j de la matrice S par :

$$A_{ij} = \frac{\exp(S_{ij})}{\sum_{j'=1}^{n} \exp(S_{ij'})},$$

de sorte que chaque ligne de A représente la distribution des poids d'attention d'un token vers les autres.

Combinaison pondérée. Enfin, la sortie du mécanisme d'attention est obtenue en combinant les valeurs V pondérées par ces coefficients d'attention :

$$\text{Attention}(Q, K, V) = A \cdot V.$$

Ainsi, chaque token de la séquence est représenté par une combinaison pondérée des représentations de tous les autres tokens, les poids indiquant leur importance contextuelle.

Le mécanisme d'attention présenté précédemment ne permet de modéliser qu'un seul type de relation entre les éléments de la séquence. Or, dans le langage naturel, les dépendances peuvent être de nature très variée : syntaxiques (par exemple, sujet-verbe), sémantiques (relations de sens), ou encore contextuelles (ton, temporalité, etc.).

5.1.3 Extention: Multi-Head attention

Pour capturer cette diversité de relations, les Transformers introduisent le concept de **Multi-Head Attention**, c'est-à-dire plusieurs têtes d'attention fonctionnant en parallèle où chaque tête apprend à se concentrer sur un aspect différent du contexte.

Principe. Au lieu d'appliquer une seule attention sur les représentations Q, K et V, le modèle crée H versions projetées dans des sous-espaces distincts grâce à des matrices de poids apprises :

$$\text{head}_h = \text{Attention}(QW_h^Q,\, KW_h^K,\, VW_h^V),$$

où $W_h^Q, W_h^K, W_h^V \in R^{d_{\text{model}} \times d_k}$ sont des matrices de projection spécifiques à chaque tête h. Ainsi, chaque tête apprend une vue partielle et complémentaire des dépendances entre tokens.

Combinaison des têtes. Les sorties de ces H têtes sont ensuite concaténées, puis repassées à travers une projection linéaire commune :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O,$$

où $W^O \in R^{Hd_v \times d_{\text{model}}}$ est une matrice de poids finale qui réunit les informations issues des différentes têtes.

On peut alors interpréter chaque tête comme un *observateur indépendant* qui se focalise sur un type particulier de relation. Par exemple :

- une tête peut apprendre à relier les verbes à leurs sujets ;
- une autre peut se concentrer sur les relations d'adjectif à nom ;
- une autre encore peut suivre les co-références (par exemple, savoir que "il" renvoie à "le client").

5.1.4 Mise à jour des matrices de poids W_Q, W_k, W_v

Après le calcul de l'attention (ou de la Multi-Head Attention), chaque token de la séquence possède un vecteur contextuel $z_i \in R^{d_{\text{model}}}$ résumant l'information combinée des autres tokens. Ces vecteurs subissent des transformations supplémentaires (nommons h_i les vecteurs issus de ces transformations finales) et sont ensuite utilisés pour prédire le token suivant (dans le cadre des modèles de langage) :

$$\hat{y}_i = \text{softmax}(h_i W_{\text{vocab}} + b),$$

où $W_{\text{vocab}} \in R^{d_{\text{model}} \times |V|}$ est en général la transposée la matrice de tous les embeddings du vocabulaire, b est un biais appris, et $|V|$ est la taille du vocabulaire. La sortie $\hat{y}_i \in R^{|V|}$ représente la distribution de probabilité des tokens possibles pour la position i.

La **fonction** calculée est généralement la cross-entropy (t_i est le token à la position i) :

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{|V|} \mathbf{1}\{t_i = j\} \log \hat{y}_{i,j}.$$

Les poids sont enfin mis à jour par des techniques de minimisation de la fonction coût (descentes de gradient).

Maintenant que nous avons vu comment fonctionnaient le méchanisme de l'attention et les Transformers, intéressons nous aux architectures BERT et RoBERTa qui dérivent de ces deux concepts.

5.2 Architectures BERT et RoBERTa

BERT : Bidirectional Encoder Representations from Transformers Proposé par [Devlin et al., 2018], BERT constitue une avancée majeure dans le domaine du traitement du langage naturel (NLP). Il repose sur l'architecture Transformer détaillée précédemment, plus précisément sur sa partie encodeur. L'idée principale de BERT est d'apprendre des représentations de texte de manière bidirectionnelle, c'est-à-dire en tenant compte du contexte de chaque mot à la fois à gauche et à droite dans une phrase.

Le modèle est préentraîné sur deux tâches principales :

- **Masked Language Modeling (MLM)** : une fraction des tokens d'entrée (environ 15%) est masquée aléatoirement, et le modèle doit prédire les tokens manquants à partir du contexte. Ce procédé permet d'apprendre des représentations contextuelles riches et bidirectionnelles.
- **Next Sentence Prediction (NSP)** : le modèle apprend à prédire si une phrase B suit réellement une phrase A dans le texte original. Cette tâche vise à améliorer la compréhension inter-phrases et la modélisation du discours.

BERT existe en plusieurs tailles (BERT-Base, BERT-Large) et a servi de base à de nombreux modèles dérivés, grâce à sa capacité à être *fine-tuné* sur des tâches variées (classification, question-réponse, extraction d'entités, etc.).

RoBERTa : (Robustly Optimized BERT Pretraining Approach) RoBERTa [Liu et al., 2019] est une amélioration de BERT proposée par Facebook AI, reposant sur la même architecture mais avec des choix d'entraînement optimisés. Les auteurs ont montré que la performance de BERT était limitée par certaines décisions d'entraînement et non par l'architecture elle-même.

Les principales différences introduites par RoBERTa sont :

- Suppression de la tâche NSP, jugée inutile pour la performance en aval ;
- Utilisation de **séquences plus longues** et de **batches plus grands** pour un entraînement plus robuste ;
- Entraînement sur un **corpus plus large** et plus diversifié (BookCorpus, Wikipedia, CC-News, OpenWebText, Stories) ;
- Utilisation d'un **dynamisme du masquage** : le masque change à chaque epoch, ce qui évite le surapprentissage des positions masquées.

Ces ajustements ont permis à RoBERTa de surpasser BERT sur de nombreux benchmarks de compréhension du langage (GLUE, SQuAD, etc.), tout en conservant la même structure d'encodeur à plusieurs têtes d'attention.

RoBERTa ne modifie donc pas la structure du modèle, mais optimise son utilisation et son efficacité à grande échelle.

Après avoir détaillé ces architectures, faisons à présent un état de l'art de certaines méthodes de recherche d'informations dans les corpus documentaires vastement employées.

5.3 Méthodes de recherche : TF-IDF, BM25, recherche sémantique et recherche hybride

Dans le cadre d'un système de recherche d'information (Information Retrieval), l'objectif est d'identifier les documents les plus pertinents par rapport à une requête utilisateur. Plusieurs approches coexistent, reposant soit sur des méthodes statistiques classiques, soit sur des représentations vectorielles apprises. Trois familles de méthodes sont particulièrement importantes dans ce contexte :TF-IDF, BM25, la recherche sémantique, et la recherche hybride.

5.3.1 TF-IDF : pondération par fréquence et rareté

Le modèle **TF-IDF** (*Term Frequency–Inverse Document Frequency*) est l'une des premières méthodes statistiques de recherche d'information. Il repose sur l'idée qu'un mot est important dans un document s'il y apparaît fréquemment (TF), mais qu'il perd de son importance s'il est fréquent dans l'ensemble du corpus (IDF).

La pondération d'un terme t dans un document D est donnée par :

$$\text{TF-IDF}(t, D) = \text{TF}(t, D) \times \text{IDF}(t)$$

avec :

$$\text{TF}(t, D) = \frac{f(t, D)}{\sum_{t' \in D} f(t', D)} \quad \text{et} \quad \text{IDF}(t) = \log\left(\frac{N}{n_t}\right)$$

où :

- $f(t, D)$ est la fréquence du terme t dans le document D,
- N est le nombre total de documents dans le corpus,
- n_t est le nombre de documents contenant le terme t.

Cependant, TF-IDF souffre de certaines limites : il ne tient pas compte de la saturation de fréquence (un mot répété dix fois n'est pas forcément dix fois plus pertinent) ni de la longueur des documents. Le modèle BM25 vient pallier à cela.

5.3.2 BM25 : un modèle de recherche statistique

Le modèle **BM25** (Best Matching 25) est une méthode de pondération issue du modèle de *probabilité de pertinence* [Robertson et al., 1995]. Il repose sur la fréquence des termes dans les documents et dans la requête pour estimer leur importance relative. La formule de BM25 est donnée par :

$$\text{score}(D, Q) = \sum_{t \in Q} \text{IDF}(t) \cdot \frac{f(t, D) \cdot (k_1 + 1)}{f(t, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

où :

- $f(t, D)$ est la fréquence du terme t dans le document D,
- $|D|$ est la longueur du document,
- avgdl est la longueur moyenne des documents de la collection,
- k_1 et b sont des paramètres de lissage (souvent fixés à $k_1 = 1.2$ et $b = 0.75$),
- IDF(t) est le poids d'inverse document frequency, mesurant la rareté du terme t dans la collection.

BM25 est efficace pour la recherche lexicale, c'est-à-dire lorsque la correspondance entre les mots-clés de la requête et ceux des documents est explicite. Cependant, cette approche échoue lorsque des synonymes, paraphrases ou reformulations apparaissent, car elle ne capture pas la similarité sémantique entre les mots. La recherche sémantique permet de capturer ce genre de similarités.

5.3.3 Recherche sémantique

La **recherche sémantique** repose sur des représentations vectorielles du texte, apprises à l'aide de modèles de langage comme BERT ou ses variantes spécialisées (Sentence-BERT, MiniLM, etc.). Chaque texte — qu'il s'agisse d'une phrase, d'un paragraphe ou d'une requête — est projeté dans un espace vectoriel de dimension d via un encodeur, de manière à ce que des textes sémantiquement proches aient des vecteurs proches selon une mesure de similarité (souvent la similarité cosinus) :

$$\text{sim}(q, d) = \frac{\langle \mathbf{q}, \mathbf{d} \rangle}{\|\mathbf{q}\| \|\mathbf{d}\|}$$

où $\mathbf{q}$ et $\mathbf{d}$ représentent respectivement les vecteurs de la requête et du document.

Cette approche permet de retrouver des documents sémantiquement proches même en l'absence de correspondance lexicale directe. Elle est particulièrement efficace dans les contextes où la reformulation du langage est fréquente, comme les systèmes de questions-réponses ou les chatbots basés sur RAG.

5.3.4 Recherche hybride

La **recherche hybride** combine les avantages des deux approches précédentes : la robustesse lexicale de BM25 et la compréhension sémantique des modèles d'embedding.

Mathématiquement, on peut combiner les scores lexicaux (s_{bm25}) et sémantiques (s_{sem}) à l'aide d'une pondération :

$$s_{\text{hyb}} = \alpha \cdot s_{\text{bm25}} + (1 - \alpha) \cdot s_{\text{sem}}$$

où $\alpha \in [0, 1]$ permet d'ajuster l'importance relative des deux composantes.

Cette approche améliore significativement la précision de la récupération de documents, notamment dans les systèmes complexes tels que les pipelines RAG, où la qualité des passages extraits influence directement la qualité de la génération finale.

Ces méthodes de recherche documentaire sont une étape essentielle du RAG (Retrieval-Augmented Generation), que nous présentons dans la sous-section suivante.

5.4 Retrieval-Augmented Generation (RAG)

L'approche **Retrieval-Augmented Generation** (RAG), introduite par [Lewis et al., 2020], constitue une avancée majeure dans la conception de systèmes de génération de texte informés par des sources externes. Elle repose sur l'idée d'enrichir la génération de texte d'un LLM à l'aide de documents récupérés dynamiquement depuis une base de connaissances externe. Le but est d'améliorer la précision et la factualité des réponses produites, tout en réduisant la dépendance du modèle à la seule information contenue dans ses poids lors de son pré-entraînement.

5.4.1 Principe général

Un pipeline RAG se compose de deux modules principaux :

- un **retriever**, chargé d'identifier dans une base documentaire les passages les plus pertinents par rapport à une requête utilisateur ;
- un **generator**, qui exploite ces passages pour produire une réponse cohérente et contextualisée.

Concrètement, lorsqu'une requête q est posée, le retriever sélectionne un ensemble de documents $D = \{d_1, d_2, \ldots, d_k\}$ pertinents à l'aide d'un modèle de recherche (par exemple BM25, recherche sémantique ou hybride). Ces documents sont ensuite injectés dans le modèle génératif, qui produit une réponse y conditionnée à la fois sur la requête et sur les documents récupérés :

$$p(y|q) = \sum_{d \in D} p(y|q, d)\, p(d|q)$$

où $p(d|q)$ correspond à la probabilité de récupération du document d par le retriever, et $p(y|q, d)$ à la probabilité de génération de la réponse donnée la requête et le document.

Le modèle peut alors produire des réponses précises même lorsqu'il n'a pas été explicitement entraîné sur le contenu demandé, en exploitant les documents externes comme source de vérité.

5.4.2 Intérêts et limites du RAG

Le principal avantage du RAG réside dans sa capacité à **améliorer la factualité et la mise à jour des connaissances** sans nécessiter de réentraînement complet du modèle de langage. Il permet ainsi de combiner la puissance des grands modèles génératifs (capables de produire un texte fluide et cohérent) et la précision des systèmes de recherche documentaire.

Cependant, plusieurs défis persistent :

- la **qualité de la récupération** influence directement celle de la réponse : un mauvais retriever mène souvent à une génération incomplète ou erronée
- la **fusion de contextes** peut induire des contradictions dans la réponse générée
- la **gestion de la longueur du contexte** (limite de tokens) impose une sélection judicieuse des passages injectés

5.4.3 Applications

Les pipelines RAG sont aujourd'hui utilisés dans de nombreux domaines nécessitant des réponses précises et contextualisées : assistants conversationnels, moteurs de recherche internes, chatbots spécialisés (juridiques, médicaux, assurantiels), ou encore génération augmentée de rapports. Cette approche constitue une étape essentielle vers des modèles de langage plus fiables et alignés sur des bases de connaissances externes.

5.5 Le reranking dans les pipelines RAG

Afin d'améliorer la qualité de la récupération d'informations dans un pipeline RAG, un module de **reranking** est souvent ajouté après la phase de recherche initiale. Ce composant vise à **réordonner** les passages récupérés par le retriever selon un score de pertinence plus précis, en affinant ainsi la sélection des documents transmis au modèle génératif.

Dans un pipeline RAG classique, la récupération des documents repose sur des méthodes de similarité vectorielle ou lexicale, telles que BM25 ou la recherche sémantique par embeddings. Cependant, ces méthodes peuvent introduire des passages peu pertinents ou redondants parmi les résultats les mieux classés, car elles évaluent la similarité de manière indépendante entre la requête et chaque document, sans véritable compréhension contextuelle fine.

Le **reranker** est introduit pour pallier cette limitation : il permet de calculer un score représentant la **pertinence contextuelle** entre la requête et chaque passage de manière conjointe, en exploitant un modèle plus puissant, souvent basé sur des architectures de type Transformer.

Les documents sont ensuite triés selon ces nouveaux scores et seuls les n premiers sont conservés pour la génération.
Le retrieval avec reranker s'effectue donc généralement selon une stratégie à deux étages: une première étape de *retrieval rapide* (BM25,semantic search, hybrid search...), suivie d'un reranking plus coûteux mais précis.

Maintenant que nous avons exploré ce qu'est le RAG, intéressons-nous à différentes approches d'amélioration des sorties de LLM pour les rendre plus spécifiques aux domaines dans lequels ils sont utilisés (post-training) par finetuning (sous-sections 5.6 et 5.7) ainsi que par reinforcement learning (section 5.8)

5.6 Low-Rank Adaptation (LoRA)

L'entraînement complet de modèles de langage de grande taille est souvent prohibitif en termes de coûts computationnels et de ressources mémoire. Pour pallier cette limitation, différentes approches de fine-tuning ont été proposées, permettant d'ajuster un modèle pré-entraîné à une tâche spécifique sans devoir réentraîner l'ensemble de ses paramètres. Parmi ces méthodes, la **Low-Rank Adaptation (LoRA)** [Hu et al., 2021] est reconnue comme une solution efficace et économe en ressources.

Le principe de LoRA repose sur l'idée que les mises à jour nécessaires à l'adaptation d'un modèle pré-entraîné résident dans un sous-espace de faible dimension. Plutôt que de modifier directement les matrices de poids d'un modèle, LoRA introduit des matrices d'adaptation dies à faible rang. Ces matrices sont les seuls paramètres entraînables, ce

qui réduit considérablement le nombre total de paramètres à ajuster tout en maintenant des performances comparables à un fine-tuning complet.

5.7 Gradual Unfreezing

L'unfreezing progressif (*Gradual Unfreezing*) est une technique de fine-tuning introduite par [Howard and Ruder, 2018] dans le cadre du modèle ULMFiT (Universal Language Model Fine-Tuning) . Elle vise à améliorer la stabilité et la performance de l'adaptation de modèles de langage pré-entraînés à des tâches spécifiques, en évitant la dégradation des représentations apprises lors du pré-entraînement, phénomène connu sous le nom de *catastrophic forgetting*.

Le principe repose sur une observation qui est que lors du fine-tuning, si toutes les couches d'un modèle sont mises à jour simultanément, les poids pré-entraînés des couches profondes risquent d'être altérés trop rapidement, entraînant une perte de généralisation. Pour y remédier, le Gradual Unfreezing consiste à rendre entraînables les couches du modèle progressivement, en commençant par la couche la plus proche de la sortie, puis en dégelant les couches inférieures une à une au fil des itérations d'entraînement.

Cette méthode a démontré son efficacité dans divers contextes, notamment en classification de texte et analyse de sentiments.

5.8 Reinforcement Learning et Group Relative Policy Optimization (GRPO)

Le **Group Relative Policy Optimization** (GRPO) est une variante récente des méthodes de reinforcement learning qui a pour objectif l'optimisation de la politique (ici, le LLM) lorsque celui-ci génère plusieurs complétions candidates pour une même entrée, en comparant ces complétions au sein d'un groupe. Cette méthode d'optimisation destinée à l'entraînement post-supervisé (post-training) de grands modèles de langage a été développée par le laboratoire de recherche de DeepSeek [Shao et al., 2024].

5.8.1 Intuition

Pour une même requête x le modèle génère un ensemble de m complétions (ou *completions*) $\{y_1, \ldots, y_m\}$. Chaque complétion est évaluée par un *reward* scalaire $r_i = R(y_i \mid x)$ (issu d'un modèle de récompense ou d'une métrique externe). Plutôt que d'estimer une valeur d'état/action par un critic, GRPO calcule un avantage relatif au sein du groupe, par exemple :

$$\bar{r}_G = \frac{1}{m}\sum_{i=1}^{m} r_i, \qquad s_G = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(r_i - \bar{r}_G)^2}, \qquad \hat{A}_i = \frac{r_i - \bar{r}_G}{s_G}. \tag{1}$$

Le signe et l'amplitude de $\hat{A}_i$ indiquent la qualité relative de la complétion y_i par rapport aux autres complétions du même groupe.

5.8.2 La fonction coût dans le cadre GRPO

La fonction de coût introduite dans le cadre du GRPO a été spécifiquement développée pour l'entraînement post-supervisé des grands modèles de langage. Néanmoins, son principe reste applicable de manière plus générale à tout problème d'optimisation de politique par renforcement. Elle peut ainsi s'écrire plus généralement:

$$L_{\text{GRPO}}(\theta) = E_{x,i}\Big[\min\big(\rho_\theta \hat{A}_i,\ \text{clip}(\rho_\theta, 1-\epsilon, 1+\epsilon)\hat{A}_i\big) - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot|x) \parallel \pi_{\text{ref}}(\cdot|x)\big)\Big] \quad (2)$$

où :

- $\rho_\theta = \dfrac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)}$ est le ratio de probabilité entre la politique courante et celle de l'itération précédente ;
- $\hat{A}_i = r_i - \bar{r}_G$ représente l'avantage relatif de la complétion i par rapport à la moyenne du groupe (cf. équation précédente) ;
- ϵ est un coefficient de *clipping* typiquement compris entre 0.1 et 0.3, qui limite les variations de probabilité trop fortes ;
- $D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})$ est la divergence de Kullback–Leibler, qui mesure l'écart entre la distribution de la nouvelle politique et celle du modèle de référence ;
- β est un hyperparamètre positif contrôlant l'intensité de la régularisation KL.

5.8.3 Avantages et limites

Parmi les principaux avantages de GRPO :

- **réduction de la charge mémoire** : simplifiant l'entraînement des grands modèles.
- **Signal relatif robuste** : l'avantage calculé au sein d'un groupe capture l'information comparative utile pour distinguer bonnes et mauvaises complétions.
- **Efficacité pratique** : pour des tâches de raisonnement / post-training sur LLMs où l'on génère plusieurs complétions par entrée.

Cette approche présente toutefois des limites et défis :

- **Coût de génération multiple** : il faut échantillonner plusieurs complétions par entrée, ce qui augmente la charge computationnelle.
- **Sensibilité au design du groupe** : la taille du groupe et la qualité des complétions candidates influencent fortement la qualité du signal d'avantage.

5.9 SLPA — Speaker-Listener Label Propagation Algorithm et conductance

Le **Speaker-Listener Label Propagation Algorithm** (SLPA) est un algorithme de détection de communautés, proposé par Xie, Szymanski et Liu (2011), conçu pour identifier des communautés chevauchantes dans des graphes. SLPA est une extension du label propagation classique qui autorise chaque nœud à accumuler et conserver plusieurs labels

au cours d'un processus d'interaction itératif entre "speaker" et "listener". À chaque itération, un nœud "speaker" choisit et émet un label depuis sa mémoire selon une distribution empirique. Le nœud "listener" recueille plusieurs labels émis par ses voisins et met à jour sa mémoire en ajoutant le label le plus fréquent, ce qui permet d'identifier des communautés potentiellement chevauchantes. A la fin des T itérations, nous sélectionnons les labels étant présents à une fréquence supérieure à r (T et r sont les hyperparamètres de cet algorithme).
Pour le détail de l'algorithme SLPA, veuillez vous référer à l'Annexe 1.

5.9.1 Avantages pratiques de SLPA

- **Complexité et scalabilité**: l'algorithme a une complexité proche de linéaire en pratique, ce qui le rend adapté aux graphes de grande taille.
- **Flexibilité**: le post-traitement avec les hyperparamètres (seuils, filtrage) permet d'ajuster la granularité des communautés détectées selon l'application.

Ces caractéristiques expliquent son adoption dans diverses applications de fouille de graphes et d'extraction de structures thématiques.

5.9.2 Conductance dans la théorie des graphes

La conductance est une mesure utilisée en théorie des graphes pour évaluer la qualité d'une séparation ou d'une communauté au sein d'un graphe. Elle permet de quantifier le degré d'isolation d'un sous-ensemble de nœuds par rapport au reste du graphe. Intuitivement, une bonne communauté est un groupe de nœuds fortement connectés entre eux, mais faiblement connectés au reste du graphe ; la conductance fournit un indicateur de cette propriété.

Soit un graphe non orienté $G = (V, E)$ et un sous-ensemble de nœuds $S \subset V$.
On note :

- $E(S, \bar{S})$ l'ensemble des arêtes reliant un nœud de S à un nœud de son complément $\bar{S}$,
- $\text{vol}(S) = \sum_{v \in S} \deg(v)$ le volume de S, c'est-à-dire la somme des degrés des nœuds appartenant à S.

La conductance de S est alors définie par :

$$\phi(S) = \frac{|E(S, \bar{S})|}{\min(\text{vol}(S), \text{vol}(\bar{S}))}.$$

Cette quantité mesure la proportion d'arêtes qui "sortent" de la communauté S par rapport à son volume. Une valeur de conductance faible indique une communauté bien délimitée, faiblement connectée à l'extérieur, tandis qu'une valeur élevée traduit une communauté peu cohésive ou mal séparée du reste du graphe.
Elle offre un compromis entre compacité interne et séparation externe, deux critères essentiels pour caractériser la qualité structurelle des clusters dans un graphe.

5.10 GraphRAG — Retrieval-Augmented Generation avec graphes

GraphRAG [Edge et al., 2024] désigne une famille d'approches qui enrichissent le paradigme RAG en exploitant la structure relationnelle de graphes de connaissance. Plutôt que de se limiter à une simple recherche par similarité sur des chunks textuels, GraphRAG exploite des graphes de connaissance pour améliorer la couverture, l'explicabilité et la précision du retrieval, puis injecte ces connaissances structurées dans la génération. Les étapes typiques comprennent : extraction/construction du graphe, indexation graph-aware, retrieval graph-guided (par propagation de scores / random walks), et enfin génération conditionnée sur les passages ou sous-graphes récupérés.

Architectures et techniques courantes dans GraphRAG Les techniques mises en œuvre dans les systèmes GraphRAG incluent :

- **Construction du graphe** : Formation du graphe de connaissance par extraction d'entités/mentions des documents, co-occurrences... puis construction d'arêtes entre les entités extraites. Enfin, construction des résumés de communautés (ces communautés résultant d'un clustering, par exemple une détection SLPA) ;
- **Indexation et retrieval graph-aware** : Une fois la question de l'utilisateur posée, deux cas sont possibles:
  - soit la question est généraliste par rapport aux données et on utilise les résumés des communautés pour y répondre ;
  - soit la question est spécifique et on effectue une recherche basée sur des mesures de proximité à la question dans le graphe (k-hop expansion, diffusion de scores...)
- **Fusion pour la génération** : sélection de résumés (question généraliste) ou de sous-graphes/nœuds pertinents transformés en contexte textuel (question spécifique) injectés dans le modèle génératif.

6 Environnement de travail et stack technologique

6.1 Données manipulées

Les données manipulées sont en grande partie des données textuelles, issues de divers documents du corpus documentaire d'Ask-Me. Ces documents sont segmentés en unités de textes cohérentes, appelées chunks, par un algorithme de chunking intelligent foruni par le service DocumentIntelligence d'Azure.
La taille d'un chunk maximale, choisie expérimentalement, est de 700 tokens. Toutefois, il se peut que la taille d'un chunk dépasse les 700 tokens si l'algorithme de chunking intelligent détecte que le chunk dépassant les 700 tokens ne peut pas être divisé en deux sous chunks ayant du sens.

L'utilisation de chunks est très utile car elle permet aux modèles de languages d'avoir accès à des briques d'informations unitaires, ce qui a deux avantages principaux: réduire la dilution d'information ainsi qu'optimiser l'utilisation de la fenêtre de contexte (context window) lors de la génération de la réponse (au lieu d'ajouter le document en entier contenant l'information utile, nous ajoutons uniquement les chunks contenant cette information).

Il arrive également de rencontrer des images ou des structures spéciales dans les documents du corpus (schémas, logos, tableaux...). Ces images sont alors converties en texte par un algorithme d'OCR (Optical Character Recognition) natif à la stack technique utilisée.

6.2 Stack technologique

La stack technique utilisée permet d'intégrer des technologies et frameworks modernes adaptés au traitement de données massives et à l'implémentation de modèles d'intelligence artificielle avancés, tout en assurant un déploiement et une mise en production efficaces:
Les principales composantes de cette stack sont les suivantes :

- **Environnement de développement :** JupyterHub, permettant une gestion centralisée des notebooks et un accès partagé aux ressources de calcul.Il facilite également la collaboration et le prototypage rapide.
- **Langages de programmation** Python constitue le langage principal, utilisé pour toutes les tâches s'axant autour du machine learning et du traîtement de la donnée. SQL est utilisé pour extraire les données. Les modèles d'IA génératives utilisés sont des modèles issus d'Azure OpenAI. Pour le prototypage de solutions nouvelles, il est possible d'utiliser des modèles opensource tels que ceux issus d'HuggingFace ainsi que des bibliothèques opensource.
- **Gestion et traitement des données :** Le pipeline exploite des outils de prétraitement et d'indexation optimisés pour de larges corpus textuels issus des services d'Azure principalement (Azure Document Intelligence pour l'OCR, Azure OpenAI pour les modèles de langage).
- **Stockage et accès aux données :** Une base de données PostgreSQL est utilisée pour stocker le corpus documentaire ainsi que toutes les données pertinentes dans

le cas d'usage d'Ask-Me (logs d'erreurs, scores de la qualité des réponses...). Les autres données sont stockées sur le cloud Azure pour garantir un accès rapide et scalable.

- **Infrastructure et calcul :** Le développement et les expérimentations sur des modèles de langage sont effectués sur un environnement doté de ressources GPU (Tesla T4, 20 Go RAM) permettant l'accélération des calculs liés à l'entrainement et à l'inférence de modèles de grande taille.

- **Déploiement et intégration continue :** Argo CD est utilisé pour gérer le déploiement continu de l'application. Cette solution permet d'automatiser la synchronisation de l'état désiré du système avec l'état réel, garantissant ainsi des mises à jour fluides et reproductibles.
  L'architecture du système s'appuie sur des conteneurs Docker, c'est-à-dire des environnements isolés où s'exécutent les différentes applications. Ces conteneurs sont orchestrés par Kubernetes, qui en assure la gestion.
  Grâce à cette organisation, les différentes fonctionnalités du système peuvent être rendues accessibles et exposées aux utilisateurs via des interfaces frontend.

- **Gestion du code :** L'ensemble du code est versionné avec Git, assurant traçabilité et collaboration efficace au sein de l'équipe.

6.3 Projet Ask-Me

Comme indiqué lors de la présentation du scrum Communications, le projet sujet du travail du stage est Ask-Me. Il s'agit d'un chatbot destiné aux agents afin de les aider dans leurs tâches nécessitant une connaissance approfondie des différents documents d'Allianz. Ce chatbot se résume à une pipeline RAG comme suit:

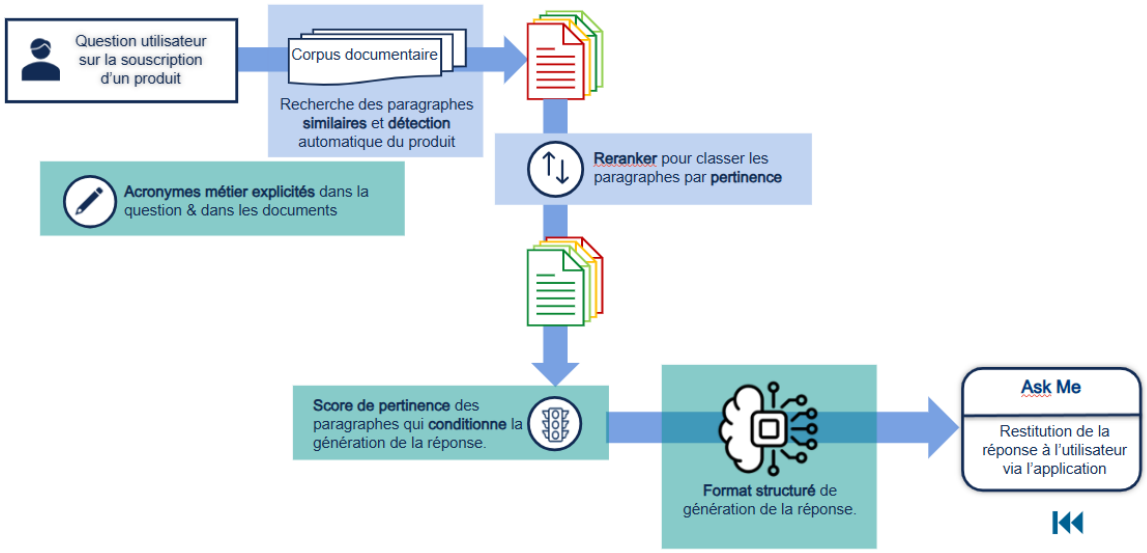


Figure 3: Architecture du RAG d'Ask-Me

- La partie Retrieval est composée de deux modules: un effectuant la similarity search (ou hybrid search) et retournant 24 chunks. Parmi ces 24 chunks, le reranker

(deuxième module) en retourne maximum 8, qui correspondent aux chunks jugés les plus pertinents pour la question de l'utilisateur. Le reranker ici est un LLM (gpt4o)

- La partie Scoring dans laquelle un score de pertinence est généré par chunk, notant sa pertinence vis-à-vis de la question.
- La partie Generation dans laquelle un LLM (gpt4o) récolte les chunks sélectionnés lors du Retrieval et les utilise comme contexte dans le prompt pour la génération de la réponse finale.

La partie qui sera sujet des travaux du mémoire est principalement la partie Retrieval.

7 Travaux réalisés

Dans cette section seront détaillés l'ensemble des trauvaux réalisés qui permettent de répondre à la problématique d'améliroation de la pipeline RAG.

7.1 Création d'une API qui expose des services de processing de texte pour des documents

L'étape du text processing de document est l'étape préliminaire à effectuer, avant de rendre cette donnée disponible dans la base de données. En effet, la pipeline RAG manipule des chunks de documents qui sont créés lors de cette étape de text processing, et il est nécessaire d'exposer cette étape en tant que service via une API.
Une API (ou Application Programming Interface) est une interface de programmation permettant à différentes applications de communiquer entre elles via le protocole HTTP, en s'appuyant sur des opérations standards (GET pour obtenir de la donnée, POST pour ajouter de la donnée, PUT pour mettre à jour de la donnée, DELETE pour supprimer de la donnée).
Transformer l'étape du chunking en API présente plusieurs avantages :
cela permet

- de rendre la fonctionnalité réutilisable et indépendante de l'environnement initial
- de faciliter son intégration dans d'autres systèmes
- de favoriser la modularité de l'architecture (en isolant cette étape comme un service distinct pouvant évoluer ou être remplacé sans impacter le reste du pipeline)

Ainsi, tout utilisateur pourra utiliser le service de text processing afin d'effectuer un chunking de ses documents. De plus, pour tout développement futur de la pipeline RAG, l'étape de text processing pourra être intégrée sans causer de conflit avec les nouvelles fonctionnalités développées.

Lorsque l'utilisateur souhaite effectuer un text processing d'un (ou de plusieurs documents), le processus se déroule comme suit:

- **Détection du format du document**: Il y a deux process distincts pour les formats pdf/word ainsi que pour les formats Excel. La manière de découper le document diffère du type de celui-ci
- **Construction des metadonnées**: Ajout de metadonnées incluant des informations diverses nécessaires pour l'identification des chunks et leur stockage en base de données (date du document, identifiant unique par hashage, détection du titre du document...)
- **Extraction intelligente de l'information des documents**: Utilisation de l'OCR d'Azure Document Intelligence pour transformer le document brut en un fichier JSON hiéarchisé de manière à refléter fidèlement la hiéarchie du document (titres, sections, sous-sections...).
- **Découpage en chunks**: Le découpage du document en chunks est réalisé sur la base du résultat de l'étape précédente, d'autres metadatas sont ensuite ajoutées

pour enrichir les chunks à l'aide d'éléments fonctionnels pour perfectionner l'étape de Retrieval, améliorer la compréhension du LLM lors de la génération de la réponse et pour monitorer les sorties de la pipeline RAG (titres de la partie/sous-partie à laquelle appartient le chunk, nombre de tokens dans le chunk...)
Une visualisation sous format pdf du découpage du document est disponible pour valider ou non les chunks obtenus

- **Sauvegarde des résultats**: Les différents chunks ainsi que la visualisation du découpage sont ensuite automatiquement sauvegardés sur les containers Azure de l'équipe pour assurer la traçabilité.

7.2 Amélioration des informations récupérées lors de l'étape du Retrieval

Un des problèmes courants dans une pipeline RAG est que certains chunks contenant des informations importantes ne remontent pas lors de la similarity search ou hybrid search. L'objectif de cette partie est de présenter les travaux réalisés pour construire une nouvelle approche d'extraction des chunks pertinents en représentant le corpus documentaire par des graphes de connaissances. Cette méthode est inspirée de celle présentée par Microsoft, intitulée GraphRAG et inroduite précédemment lors de l'état de l'art.

7.2.1 Construction des graphes

L'entièreté du corpus documentaire est divisée en périmètres d'assurances (iard-part-auto (IARD pour les automobiles) ,mrh (Multi Risques Habitations)...) . Lorsque l'utilisateur pose une question, elle est automatiquement classifiée comme appartenant à un périmètre et ce sont alors les documents de ce périmètre qui serviront à répondre à la question dans la pipeline RAG. Nous allons donc nous focaliser sur la construction d'un graphe de connaissances par périmètre au lieu d'opter pour un point de vue plus général (un graphe pour tous les périmètres) car une telle approche rendrait les calculs plus longs et perdant en granularité sans apporter d'avantage concret.

Description d'un nœud du graphe Nous avons fait le choix de considérer un nœud du graphe comme étant une représentation d'un chunk, et ceci pour plusieurs raisons :

- Choisir comme nœud un document en entier est trop restrictif. En effet, nous pouvons avoir des informations croisées ou des relations entre des chunks de documents différents.
  Nous devons donc modéliser ces relations qui peuvent exister entre les chunks de différents documents, ce qui n'est pas possible en considérant un document en entier comme nœud de notre graphe.
  De plus, un document en entier contient plusieurs informations variées, il serait donc difficile de le considérer comme nœud représentant une entité d'information uniforme dans notre graphe.

- Choisir comme nœud une sous-partie du chunk (phrase, proposition, mot...), tel que cela a été effectué dans le framework GraphRAG original, augmenterait la granularité de notre graphe mais demande plus de temps et de puissance de calcul.

Pour un bon compromis entre ces deux représentations, c'est à la maille du chunk que le graphe sera donc réalisé: un nœud = un chunk.
Nous choisissons de représenter un nœud dans notre graphe par son Id, son contenu ainsi que la liste des communautés auxquelles il appartient.

Description d'une arête du graphe Les arêtes du graphe ont pour objectif de modéliser les différentes relations qu'il peut exister entre les chunks :

- **Le premier type d'arête (que l'on va appeler proximité sémantique)** modélise deux chunks dont le contenu est fortement similaire : si les embeddings de ces deux chunks dépassent un certain seuil (fixé expérimentalement à 0.8), nous les considérons comme fortement similaires et les relions entre eux.

- **Le second type d'arête (que l'on va appeler proximité lexicale)** modélise deux chunks suffisamment proches d'un point de vue mots-clés.
  En effet, il se peut que deux chunks ne soient pas proches sémantiquement mais traitent de sujets similaires. Afin de prendre en compte ce cas, nous allons alors effectuer une extraction de mots clés via un modèle de NER (Named Entity Recognition) spécialisé pour le français (camembert-ner) qui va faire de l'extraction de lieux, personnes, concepts. Puis, nous allons relier entre eux les chunks qui ont suffisamment d'entités communes.
  Cette notion d'entités communes se traduit par une similarité de Jaccard :

  $$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3}$$

  où A et B sont deux ensembles. Ici A et B sont les ensembles d'entités extraites entre les deux chunks.

  Nous utilisons ce score au lieu du nombre d'entités communes car le nombre d'entités communes favoriserait les chunks les plus longs.
  En effet, un chunk plus long contient plus d'entités et a donc plus de chances d'être relié à d'autres chunks contrairement à des chunks plus courts.
  Afin d'éviter cette situation et de ne pas pénaliser les chunks plus courts, nous optons pour cette similarité de Jaccard qui "normalise" le nombre d'entités communes par la taille des ensembles considérés.
  Ainsi, tout couple de chunks dont la similarité de Jaccard dépasse un seuil défini expérimentalement sera relié.

- **Le dernier type d'arête (proximité structurelle)** modélise deux chunks liés entre eux par la structure du document : si deux chunks appartiennent à la même section/sous-section, nous les relions entre eux car ils peuvent contenir des informations complémentaires.

7.2.2 Construction et utilisation des communautés

Clustering pour formation de communautés: Une fois nos graphes par périmètre construits, nous réalisons un clustering sur chaque graphe afin de construire des communautés de chunks.
Il est important de choisir un algorithme de clustering rapide et permettant d'obtenir

des communautés recouvrantes (ie avoir la possibilité d'obtenir un chunk appartenant à plusieurs communautés à la fois). Ce type d'algorithme permettra alors de modéliser le fait d'avoir des chunks "centraux" (par exemple, des chunks parlant de restrictions, et dont la portée des restrictions atteint plusieurs chunks)
L'algorithme choisi est le SLPA.
Cet algorithme (comme la grande majorité des algorithmes de clustering avec classes recouvrantes) étant stochastique, nous perdons en reproductibilité. Pour mitiger cet effet, nous avons choisi de faire tourner l'algorithme plusieurs fois, de choisir le couple d'hyperparamètres optimaux qui ressort le plus souvent minimisant la conductance du graphe, puis de fixer une seed pour la reproductibilité. La minimisation de la conductance permet d'obtenir des petites communautés dont les chunks sont fortement reliés entre eux mais qui sont faiblement reliés aux chunks des autres communautés, ce qui correspond bien à notre objectif.

Utilisation des communautés : Une fois les communautés formées dans le graphe, les chunks appartenant à une même communauté sont passés à un modèle de langage afin de générer un résumé informatif de celle-ci. Nous disposons alors, pour chaque communauté, d'un résumé qui sera attaché aux chunks de cette dernière.
Par conséquent, lors de l'étape de génération dans la pipeline RAG, les chunks récupérés ainsi que les résumés les plus fréquents lors du retrieval seront utilisés pour générer la réponse.
Les résumés seront utilisés pour donner plus de contexte au modèle de langage mais ne sont pas considérés comme source d'information directe, par souci de traçabilité de l'information.
Le modèle de langage va ainsi utiliser l'information des chunks, mais également la contextualisation fournie par les résumés pour qu'il puisse mieux utiliser l'information disponible dans les chunks.

7.3 Amélioration du reranking dans la pipeline RAG

Dans le cadre d'une pipeline RAG, la création d'un reranker est essentielle pour garantir un retrieval de qualité. L'amélioration de cette brique constituerait donc une amélioration à la qualité de la récupération de l'information des documents en vue de répondre à la requête de l'utilisateur. Dans cette partie, nous proposons un framework qui permet de générer et d'entraîner de manière complètement automatisée un modèle de reranking. Le framework comporte trois étapes qui constitueront les 3 sous-parties suivantes et qui sont :

- la génération automatique d'un dataset dédié au finetuning
- le finetuning d'un modèle de scoring qui sera utilisé ensuite dans l'étape suivante
- l'alignement d'un modèle de reranking par reinforcement learning en utilisant la GRPO.

7.3.1 Génération automatique du dataset

La première étape consiste en la génération d'un jeu de données prévu pour du finetuning. Les données de notre dataset proviennent de fichiers qui contiennent des données sur des

conversations censées être représentatives des différentes conversations qu'un agent peut avoir avec Ask-Me

Composition du dataset : Une donnée du dataset est composée de :

- La question d'un utilisateur : Pour des raisons de qualité de la donnée, nous ne gardons que les premières questions posées dans les conversations d'Ask-Me. En effet, les questions qui sont posées ensuite ont beaucoup moins de chances d'être complètes et bien reformulées car étant des questions complémentaires qui rebondissent sur la réponse précédemment générée par le LLM ou viennent seulement approter des précisions sur la question précédente. En agissant ainsi, nous sommes sûrs de ne garder que des questions complètes.

- Un sous-ensemble de chunks : Nous sélectionnons un sous-ensemble de 8 chunks pour simuler un processus de retrieval et de reranking qui, dans le cas de notre pipeline RAG, ressort au maximum 8 chunks pertinents.
  Il est également à noter que pour chaque question de l'utilisateur, nous échantillonnons 10 de ces sous-ensembles pour créer la notion de "groupe" qui est nécessaire à l'algorithme de GRPO (dans notre cas, un groupe correspond aux 10 différents sous-ensembles de chunks).

- Une vérité générale : Elle sert de référence pour noter la pertinence du sous-ensemble de chunks dans la génération de la réponse à la question

- Un score de pertinence : Ce score (entier entre 0 et 3) permet de noter à quel point le sous-ensemble de chunks est pertinent par rapport à la question:

  - Un score à 0 indique que le sous-ensemble ne permet pas de répondre convenablement à la question
  - Un score à 1 indique que la qualité de la réponse en se basant sur ces chunks est moyenne (il manque des informations)
  - Un score à 2 indique que les chunks permettent de bien répondre à la question mais qu'il manque quelques détails
  - Un score à 3 indique que la réponse est satisfaisante et correspond bien à la vérité générale

Etapes de la génération du dataset : La génération du dataset se fait en plusieurs étapes automatisées :

- Nous commençons par spécifier les fichiers desquels nous sélectionnons les questions et les vérités générales associées.

- Vient ensuite l'échantillonnage des sous-ensembles de chunks. Pour ce faire, un ensemble de 24 chunks est extrait par similarity search (ou hybrid search) afin de simuler le processus de sélection des chunks de la pipeline RAG.
  Nous pouvons ensuite sélectionner arbitrairement 8 chunks parmi les 24 chunks mais nous optons pour un échantillonnage plus dynamique. Le but de cet échantillonnage est d'essayer d'inclure de la diversité dans les scores au sein de chaque groupe. En effet, il a été observé que lorsque nous échantillonnons de manière aléatoire, nous nous

retrouvons avec des groupes pour lesquels les différents sous-ensembles de chunks avaient le même score.
Or, pour obtenir un entraînement de meilleure qualité du modèle, il faut que nos scores soient plus divers au sein d'un même groupe, sinon le modèle apprendra à toujours scorer la question concernée par un même score.

L'échantillonnage dynamique s'adapte aux scores précédemment sortis au sein d'un même groupe. Nous commençons par diviser la population initiale des 24 chunks retournés en 3 sous-populations :

- les 8 chunks les plus proches sémantiquement de la question (population "bonne")
- les 8 chunks moyennement proches (population "moyenne")
- les 8 chunks les plus éloignés (population "mauvaise")

Ensuite, en fonction de la répartition des scores précédents, nous optons pour différentes stratégies d'échantillonnage :

- Stratégie équilibrée : Nous échantillonnons de manière équilibrée 8 chunks entre les 3 populations
- Stratégie "best heavy" : L'échantillonnage est biaisé envers la population "bonne"
- Stratégie "worst heavy" : L'échantillonnage est biaisé envers la population "mauvaise"
- Stratégie "mid heavy" : L'échantillonnage est biasé envers la population "moyenne"

Le choix parmi ces stratégies se fait dynamiquement : On commence par utiliser la stratégie équilibrée, puis, si par exemple les scores précédemment générés sont majoritairement égaux à 0, nous optons pour une stratégie d'échantillonnage biaisée envers la "bonne population" pour diversifier les scores au sein du groupe.

- Scoring des sous-ensembles de chunks : Une fois un sous-ensemble de chunks échantillonné, nous le notons à l'aide d'un modèle de langage de petite taille en fonction de la qualité de la réponse qu'il peut produire en le comparant à la vérité générale.

Le processus précédemment décrit est répété 10 fois par question (pour avoir des groupes de taille 10) et autant de fois qu'il y a de questions.
Le résultat final est un dataset d'environs 5000 individus.

7.3.2 Création et finetuning d'un modèle de scoring

L'objectif ici est de finetuner un modèle de scoring afin d'apprendre à noter des sous-ensembles de 8 chunks en se basant sur le dataset précédemment généré. C'est ce qu'on appelle de la distillation (transfert de connaissances d'un modèle (ici Gpt4o-mini) vers un modèle plus petit) .
A terme, ce modèle pourrait constituer une alternative à l'utilisation de Gpt4o-mini, étant plus petit, moins couteux (gratuit) et déterministe dans la génération des scores (un score donné à un sous-ensemble de chunks par ce modèle est reproductible, contrairement à Gpt4o-mini pour lequel nous n'avons pas la certitude d'obtenir le même score deux fois pour le même sous-ensemble).

Le modèle utilisé ici (ModernCamembert-base) est un modèle se basant sur l'architecture RoBERTa, pré-entraîné sur de nombreuses données textuelles en français, et disposant d'une fenêtre de contexte plus large que les modèles BERT basiques pré-entraînés sur des textes en français (modèles Camembert).

Présentation de la pipeline de finetuning automatisée : La pipeline est composée :

- D'un adaptateur pour charger et formater la donnée en un format acceptable par le modèle.
- D'un tokenizer, responsable du découpage des textes en tokens, adapté au modèle utilisé. Ce tokenizer va recevoir la question de l'utilisateur ainsi que les 8 chunks candidats concaténés et sortir le découpage en tokens adapté au modèle qui va les utiliser.
- D'un modèle de base. Il s'agit ici de ModernCamembert-base, importé depuis HuggingFace et enregistré localement. Ce modèle sera doté d'une tête de classification à 4 classes afin de classifier les 4 scores différents (0,1,2,3).
- D'un Trainer qui permet de finetuner et sauvegarder le modèle ainsi que de choisir parmi plusieurs hyperparamètres, rendant l'entraînement personnalisable. Une fois le dataset généré, celui-ci est chargé dans la pipeline de finetuning, nettoyé et divisé de manière homogène entre un dataset d'entraînement et un dataset de validation. Afin de respecter les exigences d'Allianz en matière de sécurité de la donnée, le modèle est chargé depuis HuggingFace puis sauvegardé en local. Toutes les manipulations, changements de poids et inférences seront effectuées sur ce modèle sauvegardé localement, nous affranchissant de toute connexion ou transfert des données avec le hub HuggingFace.

Ici, nous n'entraînons pas le modèle en entier mais uniquement certaines parties de celui-ci par gradual unfreezing ou par LoRA. Nous avons choisi la méthode LoRA pour la suite car le modèle ainsi finetuné affichait des performances similaires au gradual unfreezing pour un temps d'entraînement moindre.

Durant le finetuning, la pipeline effectue des sauvegardes du modèle au fur-et-à-mesure de l'entraînement (nommées checkpoints).
Ainsi, si l'utilisateur a la possibilité de retourner pendant l'entraînement à une sauvegarde précédente qu'il juge optimale et modifier l'entraînement s'il n'est pas satisfait par les performances du modèle après la sauvegarde actuelle.

Une fois le finetuning terminé, le checkpoint final du modèle est sauvegardé et prêt à l'inférence.

Optimisation du GPU pour accélérer le temps de calcul : Afin de réduire le temps de l'entraînement, passant de 7 heures à environ 1 heure, plusieurs ajustements et optimisations ont été effectués tant au niveau du GPU que de la configuration du modèle:

- Activation du mode "fp16" (float16): L'entraînement a été réalisé en précision mixte (fp16). Cela permet de réduire la mémoire utilisée et d'accélérer les calculs sur GPU compatibles (notamment les cartes NVIDIA)

- Réduction du nombre de paramètres entraînables: En mode "gradual unfreezing" ou "LoRA", seul un ensemble de paramètres de taille faible par rapport au total des paramètres du modèle est entraînable. Cela limite le nombre de gradients à calculer et réduit le coût de backpropagation.

- Utilisation de l'accumulation de gradients (gradient accumulation steps): En accumulant les gradients sur plusieurs mini-batchs avant de faire une mise à jour, on a pu augmenter la taille effective du batch (ensemble d'exemples traités simultanément lors d'une étape d'entraînement du modèle) tout en respectant les limites de mémoire GPU. Cela améliore la stabilité et la vitesse d'entraînement.

- Désactivation du padding dans le dataset: Le padding (mettre toutes les données textuelles à la même taille) a été géré dynamiquement, ce qui évite de traiter inutilement des tokens vides et réduit la charge mémoire.

- Placement optimal des modèles: Le modèle de scoring (GroupScorer) a été placé sur le GPU uniquement pendant le fine-tuning. Lors d'utilisation ultérieures (toute utilisation en dehors de l'entraînement), il sera déplacé sur le CPU pour libérer la mémoire GPU.

- Accélération du transfert des batches : Le module de la pipeline responsable du transfert des batches a été configuré pour accélérer le transfert des batches vers le GPU.

Ces optimisations combinées ont permis de multiplier par 7 la vitesse d'entraînement sans perte significative de performance.

7.3.3 Utilisation d'une méthode inspirée par la GRPO pour l'alignement du reranker

Présentation du framework : Une fois le modèle de scoring finetuné, nous allons l'utiliser dans le cadre d'un framework de Reinforcement Learning inspiré par la GRPO, adapté à notre cas. En voici une présentation :

Notre objectif est de sélectionner le meilleur sous-ensemble de 8 chunks parmi les 24 pour répondre à une question. La politique correspond au reranker dans notre cas. Elle va attribuer une probabilité de pertinence à chaque chunk, reflétant la qualité de la réponse qu'il peut fournir à la question. La sélection des 8 éléments se fait séquentiellement : à chaque étape, la politique :

- Attribue un score à chaque chunk parmi les 24 chunks puis transforme les scores en probabilités via une softmax

- Echantillonne un chunk selon ces probabilités et le retire

- Met à jour les probabilités pour l'étape suivante et réitère le processus jusqu'à avoir sélectionné 8 chunks (ce qui va correspondre à un sous-ensemble de chunks optimaux selon la politique pour répondre à la question)

L'opération précédente va ensuite être répétée 10 fois pour obtenir un groupe de 10 sous-ensembles de 8 chunks.

Le scorer précédemment obtenu joue le rôle de modèle de récompense. Ce modèle attribue un score numérique reflétant la pertinence globale du lot et peut être vu comme une "vérité" ou une référence.

Nous calculons ensuite **les avantages normalisés**:
Pour chaque sous-ensemble de 8 chunks, nous calculons l'écart entre la récompense fournie par le modèle de scoring et la moyenne des récompenses par groupe puis nous divisons par l'écart-type de ces récompenses. L'avantage ainsi calculé nous indique si la politique a fait mieux ou pire que prévu dans ses choix.

La politique est ensuite mise à jour pour **minimiser la fonction coût de la GRPO** :
Les lots avec un avantage positif voient leur probabilité augmentée, ceux avec un avantage négatif voient leur probabilité diminuée, permettant ainsi à la politique d'apprendre à préférer les lots mieux notés.
Une pénalité est également ajoutée, conformément à la fonction coût de la GRPO, en utilisant la divergence de Kullback-Lieber pour limiter l'écart avec une politique de référence stable (ici, la politique de départ avant entraînement) pour éviter que la politique entraînée dérive trop vite ou sur-apprenne.

Limitations computationnelles : Malgré tout le soin apporté pour gérer de manière efficace et optimisée la mémoire sur le GPU, le débordement de mémoire n'a pas pû être évité, cela étant causé par la taille assez conséquente des chunks. L'entraînement n'a donc pas pû aboutir. Toutefois, l'équipe a été satisfaite du framework de GRPO adapté dans sa forme actuelle, qui pourra être utilisé une fois les ressources en GPU améliorées.

7.3.4 Automatisation de l'envoi des Kpis d'Ask-Me

Ask-Me dispose de différentes métriques stockées en base de données reflétant des performances du projet (nombre de questions, nombre de feedbacks, nombre de réponses bien/mal scorées, nombre de timeouts...). L'objectif ici sera de développer une solution qui permettra de transmettre des Kpis aux référents métiers des périmètres de l'outil afin qu'il puissent monitorer l'état du projet.

Pour cela, nous avons développé un workflow d'envoi de mails automatique tous les débuts de semaine contenant des Kpis cumulés du début de l'année jusqu'au jour précédent la semaine d'envoi.
Les étapes de la génération de ces Kpis sont les suivantes :

- Premièrement, nous effectuons une série de requêtes SQL à la base de données afin de récupérer les différentes données nécessaires brutes
- Ensuite, nous formatons ces données tel qu'exigé par les référents métiers
- Ces données sont ensuite automatiquement envoyées par email aux responsables sous format excel et sauvegardées automatiquement sur Azure, et ce, chaque début de semaine, grâce à un déploiement en utilisant un cronjob sur ArgoCD. Un cronjob est une tâche crée à l'aide de cron, qui est un programme qui permet aux utilisateurs d'éxécuter une tâche automatiquement à une date et heure spécifiée à l'avance.

Il est à noter que le Workflow mis en place empêche la génération multiple d'un même fichier de KPIs. Avant même de procéder à la collecte des Kpis, une étape de vérification s'assure que les fichiers déjà envoyés et sauvegardés ne soient pas recréés. Ainsi, si un fichier existe déjà, il ne sera ni regénéré, ni renvoyé, ni sauvegardé à nouveau.

8 Résultats et Analyse

8.1 RAG et graphes de connaissances

Afin d'obtenir un clustering idéal par l'algorithme SLPA, nous avons exécuté cet algorithme plusieurs fois et choisi les hyperparamètres optimaux (qui minimisent la conductance) les plus fréquents (dans la figure qui suit illustrant ceci, nous avons choisi par conséquent les couples t=80 et r=0.1).

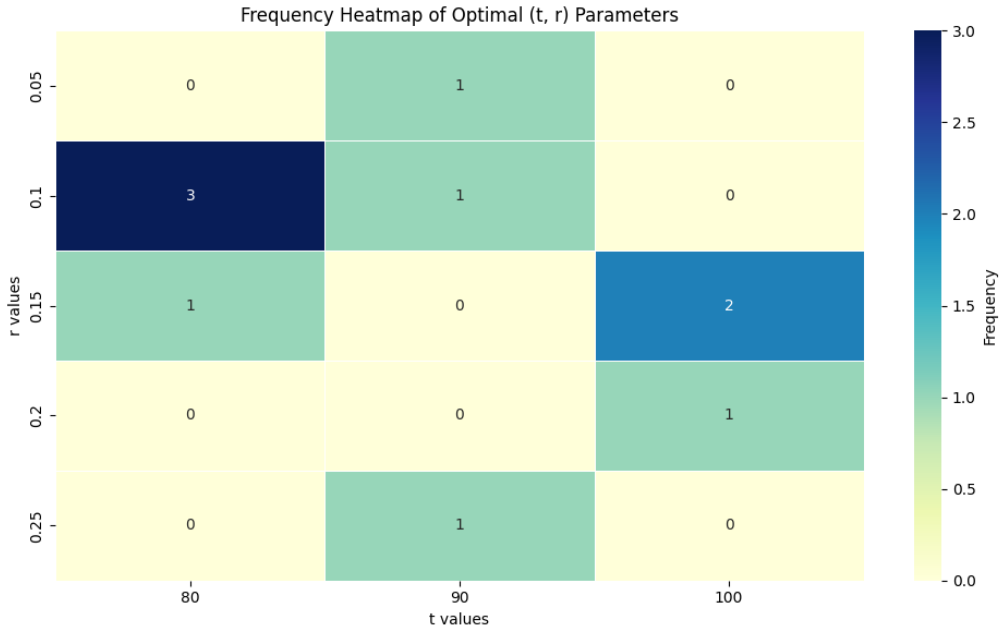


Figure 4: Heatmap des hyperparamètres minimisant la conductance

Une fois les valeurs des hyperparamètres fixées, nous réalisons le clustering en communautés recouvrantes par l'algorithme SLPA avec ces hyperparamètres optimaux.

Nous pouvons visualiser la forme du graphe construit, comme on peut le voir dans le graphe ci-après, représentant un graphe de connaissances sur le périmètre MRH (multi-risques habitation) :

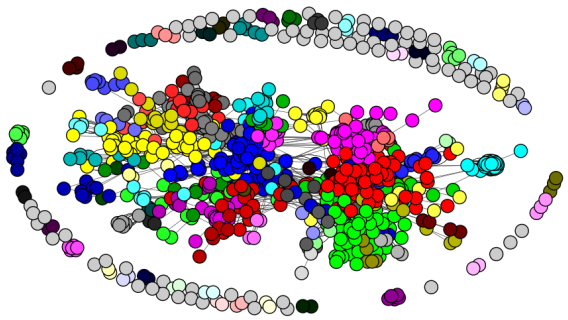


Figure 5: Représentation du corpus MRH par un graphe

Nous pouvons tirer plusieurs informations de ce graphe :

- Premièrement, nous avons des chunks qui n'appartiennent à aucune communauté (gris clair, aux extrémités de la représentation). L'algorithme considère donc ceux-ci comme étant trop différents du reste des chunks du périmètre MRH pour être catégorisés dans une communauté
- Également, nous remarquons la présence de communautés de grande taille, suggérant donc la présence de chunks "centraux" qui sont en relation avec plusieurs autres chunks
- Finalement, nous remarquons la présence de communautés de tailles différentes et variées, indiquant la présence d'informations spécifiques dans les documents (petites communautés) ainsi que d'informations plus générales (communautés de grandes tailles)

Ceci est d'autant plus visualisable dans la figure suivante :

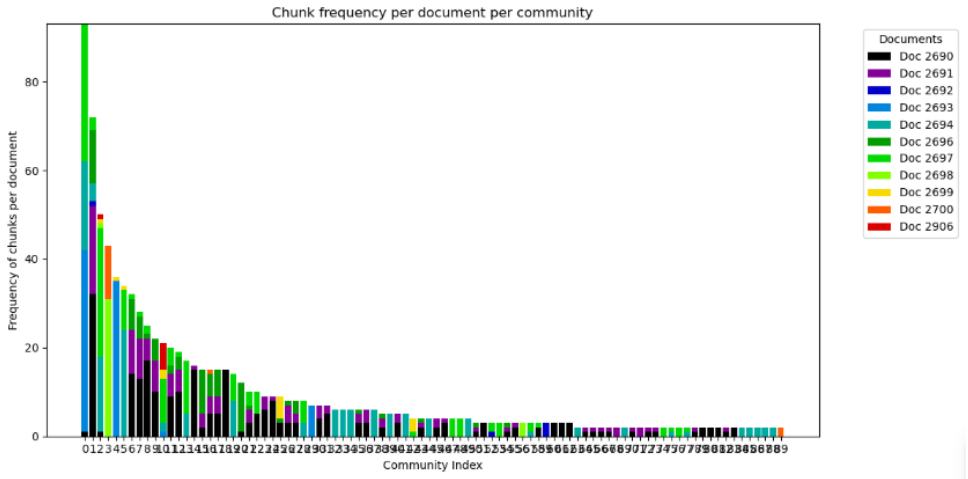


Figure 6: Histogramme et composition des communautés

Nous pouvons observer ici la présence de communautés de taille élevées (100 chunks pour la communauté 0) contre d'autres communautés à 2 ou 3 chunks. Nous remarquons également que dans les grandes communautés, nous avons une présence de nombreux documents, contrairement aux communautés plus petites qui souvent, présentent des chunks d'un unique document, renforçant cette idée que les grandes communautés abritent de l'information transverse aux documents alors que les petites communautés abritent de l'information plus spécifique.

Nous pouvons également vérifier si notre construction des liens est pertinente. Pour cela, nous avons réalisé un graphe dans lequel nous pouvons visualiser les embeddings des chunks en 3 dimensions, coloriés par appartenance aux communautés:

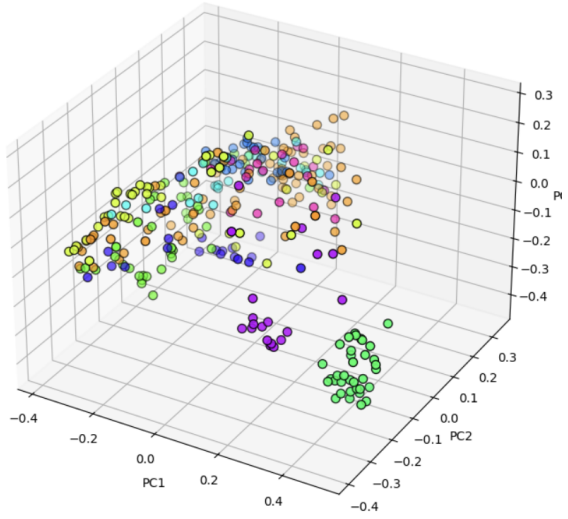


Figure 7: Projection des embeddings sur les 3 axes principaux, coloriés par communautés

Nous observons ici que nous avons des communautés fortement définies par la proximité sémantique des chunks (par exemple la communauté verte en bas à droite). Ceci est observable par un fort regroupement au sein d'une même communauté des points représentant les embeddings projetés de ces chunks.
Au contraire, nous observons également des communautés (comme celle en orange) pour lesquelles les embeddings des chunks ne sont pas regroupés, indiquant que ce critère de proximité par embeddings n'a pas joué et que ce sont les deux autres critères qui sont le plus représentatifs pour sa construction.
Finalement, nous pouvons également remarquer des communautés pour lesquelles les projections des embeddings sont à la fois regroupées et éclatées (la communauté en violet), indiquant l'importance de plusieurs critères de proximité dans sa construction.

Comme expliqué précédemment, les trois types de relations ont donc leur importance dans la construction d'arêtes pour la création du graphe. L'extraction de mots-clés étant une étape importante dans la construction des arêtes par mots-clés, observons les mots-clés obtenus pour vérifier la qualité de leur extraction:

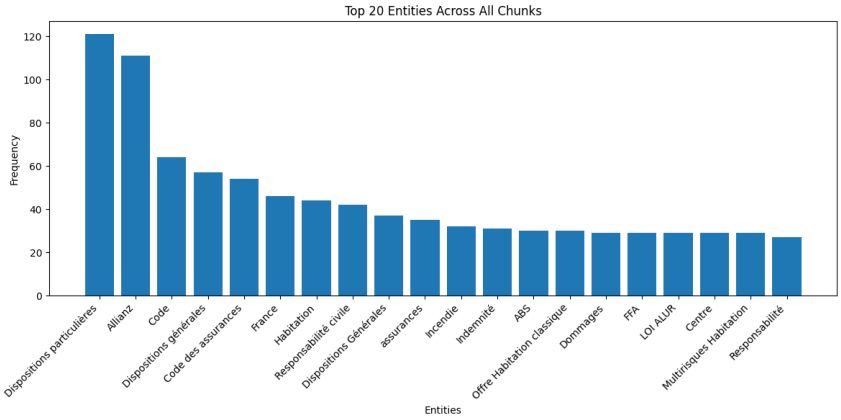


Figure 8: 20 mots-clés les plus fréquents dans le cadre du périmètre MRH

Nous remarquons ici la présence de mots-clés liés au multi-risque habitations (tels que:

habitation, Multirisques Habitations, Offre habitation classique), ce qui est encourageant car nous indique que l'extraction de mots-clés a bien saisi la thématique du périmètre.
Egalement, nous remarquons la présence de mots-clés liés à des chunks dont le contenu pourrait affecter plusieurs chunks (par exemple, dispositions générales), nous indiquant alors que nous arrivons à détecter des informations cruciales présentes dans des chunks et qui ont effet sur d'autres chunks, ce qui est également encourageant.
Nous remarquons enfin la présence de mots clés génériques tels qu'Allianz ou France, ainsi que de mots-clés répétés car pas sous la même forme(tels que Dispositions générales). Ceci nous indique la nécessité d'un post-traitement des mots-clés utilisés, nottamment pour supprimmer les mots-clés génériques et pour normaliser les mots-clés afin d'éviter des doublons. Ce post-traitement a alors été intégré dans la solution actuelle développée.

Minimiser la conductance est essentiel à l'obtention de communautés adéquates (chunks d'une même communauté fortement liés entre eux et faiblement liés aux chunks des autres communautés).
Sans ceci nous obtiendrons des communautés peu représentatives, comme nous pouvons le voir dans la figure ci-après :

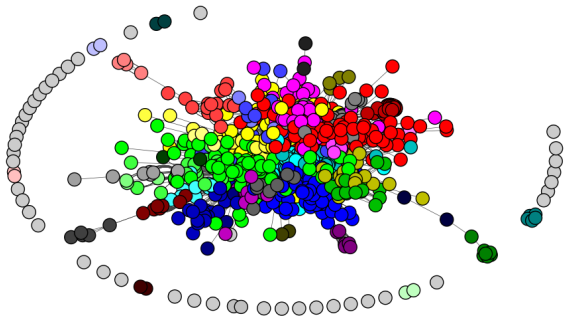


Figure 9: Représentation du corpus MRH par graphe, sans conductance minimisée

En effet, nous remarquons ici que les communautés formées sont de plus grande taille que celles formées précédemment, rendant leur interprétation plus complexe et la tâche de les résumer moins évidente, étant donné le nombre important de chunks qui y sont présents, au risque de perdre de l'information.

Également, nous pouvons observer sur la figure suivante que l'écart de taille entre les grandes communautés et les plus petites est beaucoup plus prononcé que lorsque la conductance est minimisée, ce qui réduit l'interprétabilité des grandes communautés et indique un clustering qui s'éloigne de ce que nous souhaitons obtenir.

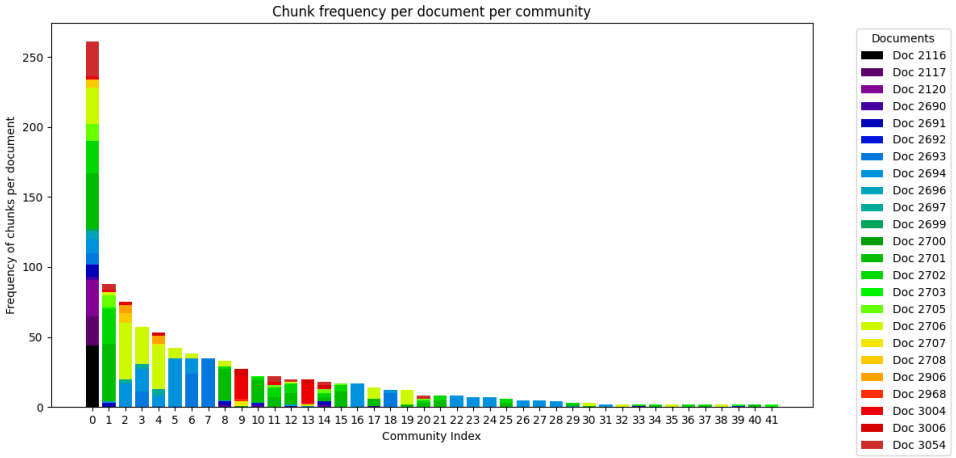


Figure 10: Histogramme et composition des communautés, sans conductance minimisée

Afin de conclure sur les résultats obtenus, nous pouvons affirmer que notre construction de graphes et notre détection de communautés sont satisfaisantes. Les résumés obtenus (voir Annexe 2 pour un exemple) ont aussi été jugés par l'équipe comme étant de bonne qualité et très informatifs, nous encourageant à considérer la piste de GraphRAG comme prometteuse pour une future mise en production.
L'approche GraphRAG apporte donc une amélioration par rapport au RAG classique, et ceci, sans ralentir le chatbot (les résumés étant générés et stockés en avance, pas lorsque l'utilisateur pose une question).

8.2 Amélioration du reranking

Nous présentons ici les résultats de la pipeline de génération automatique du dataset ainsi que du finetuning du modèle de scoring. En ce qui concerne la génération du dataset, nous pouvons afficher la distribution des scores attribués aux chunks :

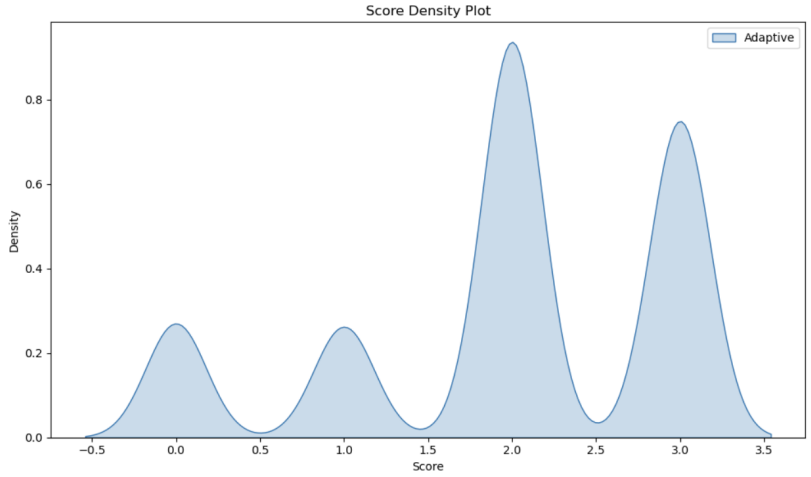


Figure 11: Distribution des scores dans le dataset

Nous remarquons que les classes 0 et 1 restent sous représentées par rapport aux classes 2 et 3 malgré notre effort en établissant une stratégie d'échantillonnage adaptative.
Les résultats précédents nous indiquent qu'il est nécessaire d'effectuer un découpage train/test stratifié et de pondérer la fonction coût du finetuning afin de prendre en compte le déséquilibre dans les classes.

Une fois le finetuning réalisé la pipeline d'entraînement permet d'obtenir et d'afficher des métriques indicatrices de la qualité de celui-ci:

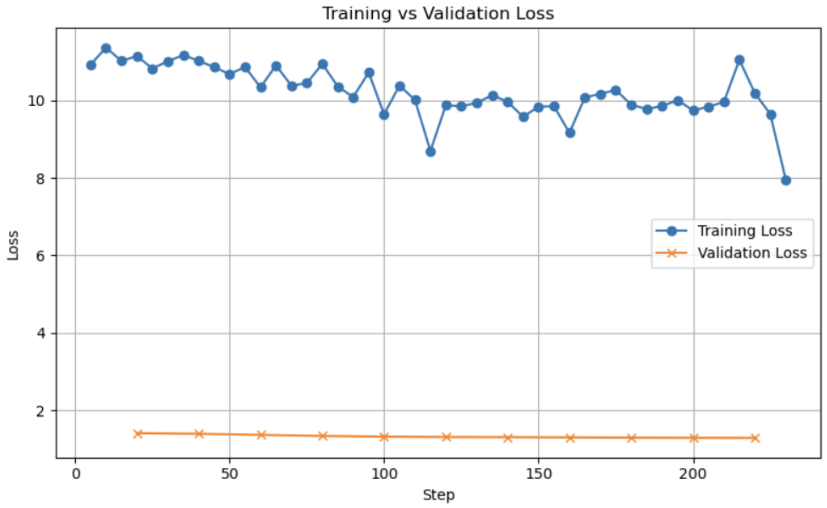


Figure 12: Train loss et eval loss

Ici, nous observons que la fonction coût lors de la validation est nettement plus faible que la fonction coût lors de l'entraînement. Cela est dû à l'activation du dropout (mise de poids à 0 aléatoirement) lors de l'entraînement et pas lors de la validation. Également, nous remarquons que la fonction coût continue à décroître, ce qui indiquerait qu'avec un entraînement plus long, nous pourrions avoir une fonction coût en entraînement qui décroît davantage.

Observons la fonction coût lors de l'évaluation:

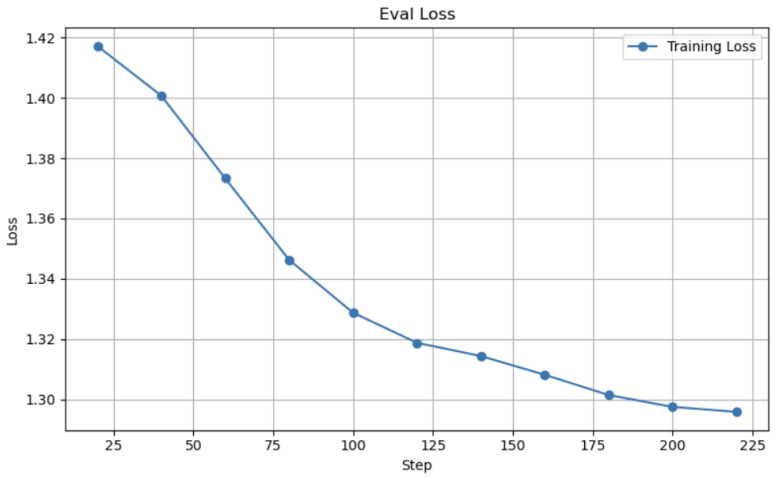


Figure 13: Eval loss

En observant la fonction coût lors de l'évaluation de plus près, nous remarquons qu'elle décroît, indiquant une bonne capacité de généralisation. Toutefois, les résultats précédents nous montrent qu'il faudrait également prolonger l'entraînement, ce qui n'a pas été possible avec les ressources actuelles.

Analysons à présent les performances du modèle sur des métriques spécifiques à la classification :

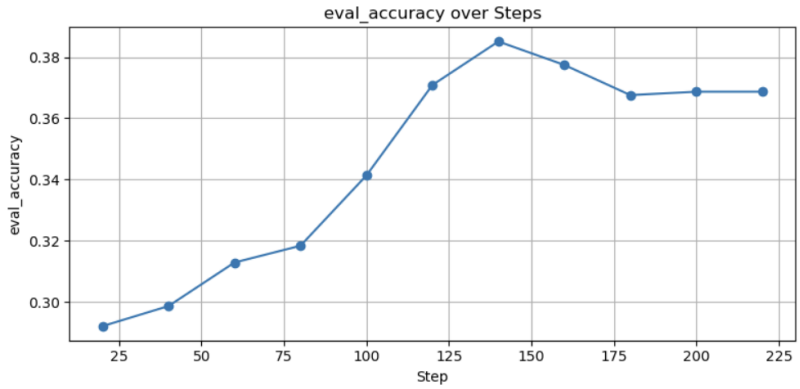


Figure 14: Accuracy lors de la validation

Nous remarquons qu'ici l'accuracy augmente au fur-et-à-mesure de l'entraînement, ce qui est encourageant. Elle atteint un pic à 0.38. Les valeurs restent cependant assez faibles mais supérieures à 0.25, qui est l'accuracy d'un modèle qui classifie complètement aléatoirement les groupes de chunks.

Observons à présent les F1 scores :

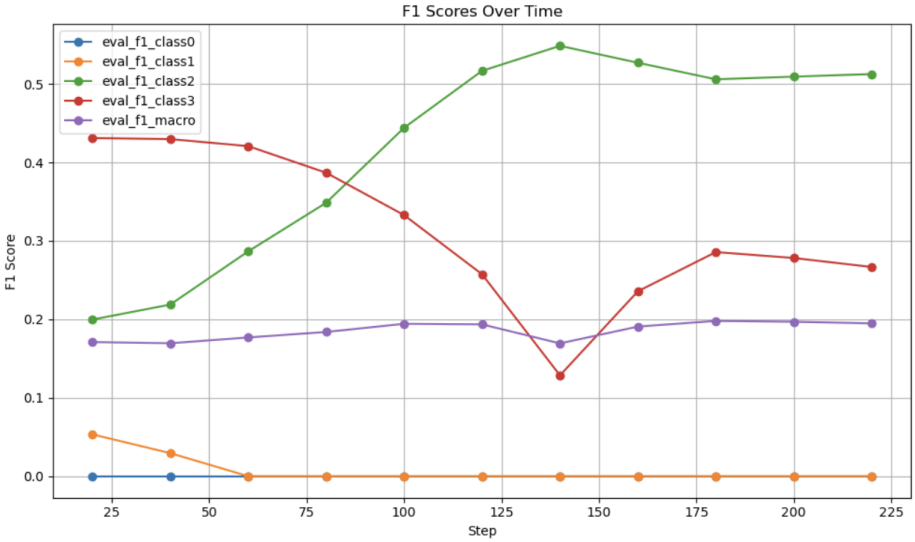


Figure 15: F1 scores lors de la validation

Nous observons que:

- Les F1 scores des classes 0 et 1 sont nuls ou presque tout au long de l'entraînement, indiquant que le modèle ne parvient pas à prédire correctmeent ces classes.
- Le F1 score de la classe 2 croît continuellement pour arriver à environs 0.5, indiquant que le modèle apprend bien cette classe
- Le F1 score de la classe 3 chute brusquement à environs 0.15 puis augmente avant de stabiliser, indiquant un déséquilibre temporaire ou un problème de surapprentissage/désapprentissage
- Enfin, le F1 score macro (La moyenne des F1 scores par classe) reste globalement constant mais faible, indiquant une ambivalence dans l'apprentissage des classes (les classes 0 et 1 sont mal apprises, contrairement aux classes 2 et 3 qui le sont mieux)

Ceci est dû au déséquilibre de notre jeu de données d'entraînement, malgré nos efforts pour nous en affranchir. Il aurait alors fallu un dataset plus large afin de voir les effets de la stratégie de sampling adaptative pour mitiger l'effet du déséquilibre.

Vérifions à présent d'autres indicateurs de la qualité de l'entraînement :

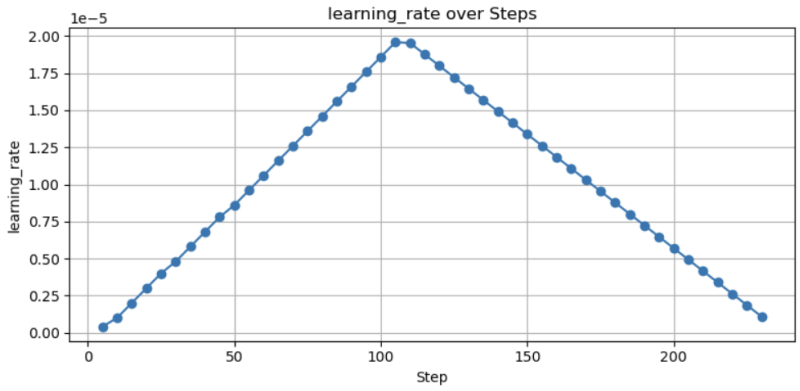


Figure 16: Evolution du learning rate lors de l'entraînement

Sur la figure ci-dessus, nous pouvons observer l'évolution du learning rate en fonction des itérations de l'entraînement. Le learning rate augmente linéairement jusqu'à la moitié de l'entraînement avant de décroître linéairement. Ce comportement, connu par le nom de learning rate linear decay, est encourageant car cela nous indique que le modèle, au départ, augmente le learning rate afin d'accélérer la minimisation de la fonction coût. Une fois proche d'un minimum de celle-ci, le learning rate diminue afin de stabiliser la convergence vers le minimum, nous indiquant que nous convergeons effectivement vers un minimum.

Inspectons à présent comment se comporte la norme du gradient lors de l'entraînement :

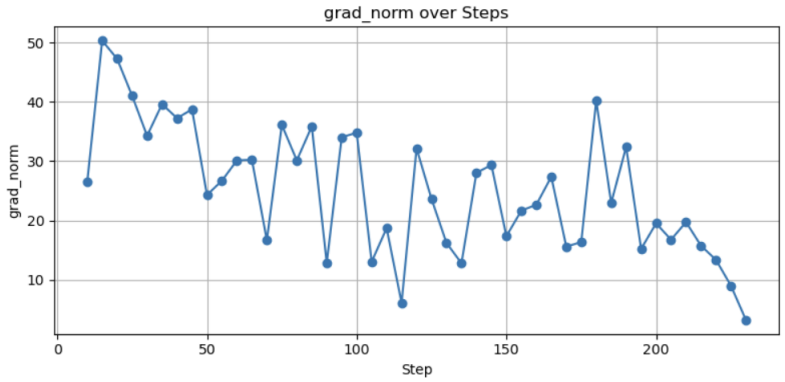


Figure 17: Evolution de la norme du gradient lors de l'entraînement

Nous remarquons que la norme du gradient reste bornée, nous indiquant qu'il n'y a pas eu durant l'entraînement de phénomènes d'explosion de gradient ou de disparition de gradient (vanishing gradient). Le modèle reste donc numériquement stable durant l'entraînement et l'optimisation reste sous contrôle : il n'y a pas de bons erratiques dans l'espace des paramètres et le modèle suit une trajectoire d'optimisation raisonnable.

Afin de conclure sur ces différentes observations, nous pouvons affirmer que le modèle ne dispose pas de performances suffisamment bonnes pour entrer en production. Ceci est principalement dû à un dataset avec peu de données et à une restriction d'un point de vue puissance de calcul, empêchant un entraînement plus long.

Toutefois, l'objectif avec l'équipe était uniquement d'effectuer un POC (proof of concept) indiquant que la piste reste envisageable et que le modèle est prometteur, avant d'investir plus de temps et de ressources dans le projet. Le fait que le modèle ait une meilleure accuracy qu'un modèle aléatoire, que notre training loss et validation loss soient continuellement décroissantes et que l'entraînement reste maîtrisé (pas d'explosions ou de disparitions de gradients, une évolution du learning rate stable) sont des arguments pour considérer l'approche encourageante et approfondir cette piste plus tard avec un dataset plus large (de l'ordre de grandeur de 100000 données) et un temps de calcul plus conséquent, d'autant plus que la pipeline actuellement développée permet la mise à l'échelle de l'entraînement.

9 Conclusion

Les travaux réalisés ont permis de mettre la lumière sur des méthodes innovantes à haut potentiel.

Les résultats de l'approche inspirée par GraphRAG quant à la construction des communautés et des résumés obtenus sont satisfaisants. De plus, le faible coût computationnel de la méthode et son impact faible sur le temps de traîtement d'une question en fait un très bon candidat d'amélioration du chatbot.

Les résultats obtenus pour essayer d'améliorer le reranking nous montrent le potentiel que peut avoir une approche de reranking sans LLM. Ils nous indiquent également la nécessité d'investir dans la construction d'un dataset de taille beaucoup plus importante afin d'observer de vrais résultats tangibles, malgré des résultats assez encourageants.

La réalisation de GRPO étant compromise par le manque de puissance de calcul, il s'agit néanmoins d'une piste intéressante à explorer lorsque la puissance de calcul ne sera plus une contrainte.
De plus, la création de la pipeline de GRPO permettra au scrum Communications d'expérimenter avec des approches similaires de Reinforcement Learning si cela est souhaité.

Les différentes automatisations réalisées, tant au niveau de l'API text processing que de l'envoi des emails automatisés aux référents métiers, a permis de rendre le projet plus automatisé et plus facile à monitorer et à présenter.

Le stage réalisé a été très enrichissant, tant au niveau technique qu'humain. Il a permis de développer des compétences de programmation respectant les standards d'un grand groupe et de mettre en pratique des avancées en intelligence artificielle pour répondre à des besoins concrets. Il a également favorisé la collaboration en équipe, par l'échange d'idées visant à améliorer des solutions existantes et la présentation du travail effectué devant toute l'équipe.

10 Bibliographie

References

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Edge et al., 2024] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., and Larson, J. (2024). From local to global: A graph rag approach to query-focused summarization. Microsoft Research Technical Report.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

[Hu et al., 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[Lewis et al., 2020] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems (NeurIPS)*.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Robertson et al., 1995] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. (1995). Okapi at trec-3. *Text REtrieval Conference (TREC)*.

[Shao et al., 2024] Shao, Z., Li, S., Li, X., Xu, Y., Liu, Z., and Sun, M. (2024). Grpo: Group relative policy optimization for fine-tuning large language models. *arXiv preprint arXiv:2402.03300*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*.

[Xie et al., 2011] Xie, J., Szymanski, B. K., and Liu, X. (2011). Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 344–349.

11 Annexes

11.1 Annexe 1: Algorithme SLPA

Entrée: Graphe non orienté $G = (V, E)$
Nombre d'itérations T
Seuil de fréquence r
Sortie : Ensemble de communautés chevauchantes $\mathcal{C}$

1. Initialisation :
foreach $v \in V$ **do**
  $M(v) \leftarrow \{label_v\}$ `// Mémoire contenant son propre label unique`
end

2. Boucle principale :
for $t = 1$ **to** T **do**
  Parcourir les nœuds dans un ordre aléatoire : shuffle(V)
  foreach $v \in$ *shuffle*(V) **do**
    `// Étape Speaker-Listener`
    foreach $u \in N(v)$ **do**
      Le speaker u choisit un label l_u de $M(u)$ selon
      $P(l_u) = \frac{\text{freq}(l_u)}{|M(u)|}$
      u envoie l_u à v
    end
    $l^* \leftarrow$ label le plus fréquent parmi $\{l_u \mid u \in N(v)\}$
    $M(v) \leftarrow M(v) \cup \{l^*\}$
  end
end

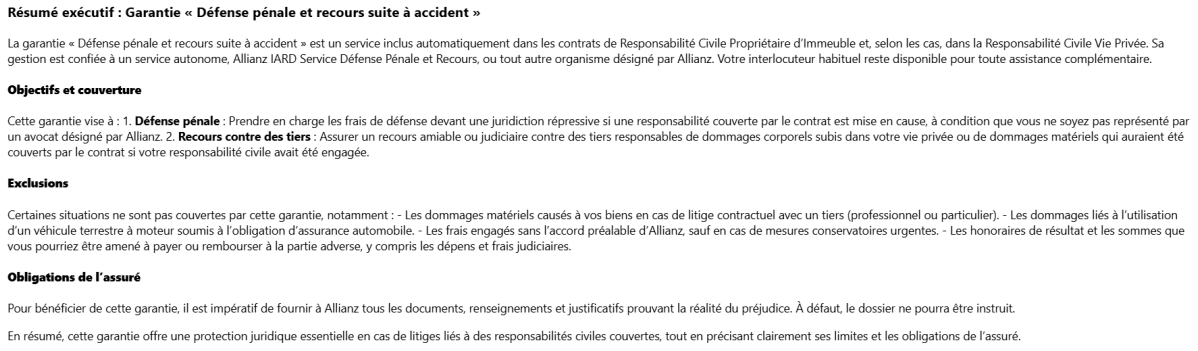
3. Post-traitement :
foreach $v \in V$ **do**
  Calculer la fréquence de chaque label dans $M(v)$
  Ne conserver que les labels dont freq $\geq r$
end

4. Construction des communautés :
foreach *label* l *conservé* **do**
  Créer une communauté $C_l = \{v \in V \mid l \in M(v)\}$
end

5. Résultat :
Retourner l'ensemble $\mathcal{C} = \{C_l\}$

Algorithm 1: SLPA — Speaker-Listener Label Propagation Algorithm [Xie et al., 2011]

11.2 Annexe 2: Exemple de résumé d'une communauté



Résumé exécutif : Garantie « Défense pénale et recours suite à accident »

La garantie « Défense pénale et recours suite à accident » est un service inclus automatiquement dans les contrats de Responsabilité Civile Propriétaire d'Immeuble et, selon les cas, dans la Responsabilité Civile Vie Privée. Sa gestion est confiée à un service autonome, Allianz IARD Service Défense Pénale et Recours, ou tout autre organisme désigné par Allianz. Votre interlocuteur habituel reste disponible pour toute assistance complémentaire.

Objectifs et couverture

Cette garantie vise à : 1. **Défense pénale** : Prendre en charge les frais de défense devant une juridiction répressive si une responsabilité couverte par le contrat est mise en cause, à condition que vous ne soyez pas représenté par un avocat désigné par Allianz. 2. **Recours contre des tiers** : Assurer un recours amiable ou judiciaire contre des tiers responsables de dommages corporels subis dans votre vie privée ou de dommages matériels qui auraient été couverts par le contrat si votre responsabilité civile avait été engagée.

Exclusions

Certaines situations ne sont pas couvertes par cette garantie, notamment : - Les dommages matériels causés à vos biens en cas de litige contractuel avec un tiers (professionnel ou particulier). - Les dommages liés à l'utilisation d'un véhicule terrestre à moteur soumis à l'obligation d'assurance automobile. - Les frais engagés sans l'accord préalable d'Allianz, sauf en cas de mesures conservatoires urgentes. - Les honoraires de résultat et les sommes que vous pourriez être amené à payer ou rembourser à la partie adverse, y compris les dépens et frais judiciaires.

Obligations de l'assuré

Pour bénéficier de cette garantie, il est impératif de fournir à Allianz tous les documents, renseignements et justificatifs prouvant la réalité du préjudice. À défaut, le dossier ne pourra être instruit.

En résumé, cette garantie offre une protection juridique essentielle en cas de litiges liés à des responsabilités civiles couvertes, tout en précisant clairement ses limites et les obligations de l'assuré.

Figure 18: Annexe 2: Exemple de résumé d'une communauté