3D Smart Factory - Mohammadia

Projet du stage de la fin d'année

NLP pour l'analyse des articles scientifiques

Business Intelligence & Analytics

Realisé par :
Amin BENALI

Encadré par :
Mr.Thierry Bertin Gardelle

Année universitaire 2023/2024

NLP pour l'analyse des articles scientifique

Table des matières

Table des figures

Liste des tableaux

List of abbreviations

Abbreviations

AI	Artificial Intelligence
NLP	Natural Language Processing
RAG	Retrieval-Augmented Generation
LLM	Large Language Model

Remerciements :

En premier lieu, je tiens à remercier toute l'équipe de la société 3D Smart Factory pour m'avoir accepté en tant que stagiaire au sein de leur entreprise. En second lieu, je tiens à exprimer ma sincère gratitude à M. Thierry Bertin, le CEO de cette société, ainsi qu'à mon encadrant, pour sa disponibilité, ses précieux conseils, et son accompagnement tout au long de la réalisation de ce projet.

Je remercie également toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce travail, ainsi que les membres du jury qui ont accepté d'évaluer ce projet.

Résumé :

Ce rapport détaille les étapes suivies pour la réalisation de mon projet de stage : NLP pour l'analyse des articles scientifiques. Il consiste à construire et évaluer un système de génération augmentée par la récupération (RAG).

Ce projet a pour objectif de simplifier la recherche d'informations dans la littérature scientifique en s'appuyant sur les dernières avancées en traitement du langage naturel (NLP), notamment les grands modèles de langage (LLM), les modèles d'embeddings et les bases de données vectorielles. Ces composants interagissent entre eux pour générer des réponses aux requêtes des utilisateurs, en se basant à la fois sur les données d'entraînement du LLM, mais surtout sur une base de données externe contenant des informations plus récentes ou plus spécifiques. Ainsi, les utilisateurs peuvent extraire très rapidement toute information recherchée à partir d'une base de données alimentée par des articles scientifiques.

Abstract :

This report presents the steps taken in the completion of my internship project : NLP for analyzing scientific articles. It involves building and evaluating a Retrieval-Augmented Generation (RAG) system.

The goal of this project is to simplify the retrieval of information from scientific literature by leveraging the latest advancements in Natural Language Processing (NLP), including large language models (LLMs), embedding models, and vector databases. These components interact to generate answers to user queries, relying both on the LLM's training data and, more importantly, on an external database containing more recent or specific information. As a result, users can quickly extract any sought-after information from a database filled with scientific articles.

Introduction Générale

Les domaines scientifiques évoluent à un rythme exponentiel. Et les avancées scientifiques génèrent une quantité énorme et cruciale d'informations. Cependant, cette masse d'informations pose un défi aux étudiants et aux professionnels qui cherchent des informations précises et spécifiques dans la littérature scientifiques.

Une solution proposée par l'intelligence artificielle générative est l'utilisation des grands modèles de langage (LLM), qui permettent de générer des réponses à partir de vastes bases de données d'entraînement. Néanmoins, ces modèles rencontrent encore des difficultés à fournir des réponses pertinentes en raison de leurs réponses non spécifiques, de l'obsolescence progressive de leurs bases de données et du problème de l'« hallucination ».

Dans ce contexte, une nouvelle approche est apparue pour surmonter les limites des LLM : les systèmes de génération augmentée par la récupération (RAG). Ce processus de génération simplifie la recherche d'informations en analysant automatiquement une grande quantité de données externes sans la nécessité de reconstruire un LLM. Ce projet a pour objectif de créer une application bureau intégrant un système RAG avec une interface utilisateur, afin de faciliter la recherche d'informations dans des corpus textuels scientifiques riches.

Ce rapport présente tout d'abord l'état de l'art des différentes composantes ainsi que les objectifs du projet. Ensuite, il discute l'analyse et la conception, en détaillant les exigences fonctionnelles et non fonctionnelles, l'architecture du système et les choix technologiques de l'implémentation. Enfin, il traite la phase de la réalisation suivie par l'évaluation des performances de ce système RAG développé.

Chapitre 1

Contexte général du projet

Introdution

Ce chapitre aborde le contexte général du projet. Il commance par présenter l'organisme d'accueil et ses activités principales. Il met en lumière la problématique à laquelle le projet cherche à répondre, ainsi que ses objectifs. Ensuite, il traite la recherche webographique et bibliographique. Enfin, le plan du projet suivi pour compléter ce projet.

1.1 Présentation de l'organisme d'accueil

1.1.1 Présentation du 3D Smart Factory

3D SMART FACTORY est une entreprise marocaine innovante, fondée à Mohammedia en 2018 et dirigée par Khalil Lbbarki en tant que directeur. Elle se concentre sur la recherche et le développement de technologies 3D.

L'objectif principal de 3D SMART FACTORY est d'accompagner les jeunes entrepreneurs dans la concrétisation de leurs projets, en leur fournissant les outils nécessaires pour transformer leurs idées en réalité.

En tant qu'entreprise orientée vers la recherche et le développement, 3D SMART FACTORY s'engage à soutenir les entrepreneurs à chaque étape de leur parcours, depuis la phase de recherche jusqu'à la production, avec la vision d'avoir un impact positif sur le développement économique et social.

En plus de ses activités de R&D, l'entreprise se spécialise également dans la conception d'appareils multitechnologiques pour le secteur médical, illustrant ainsi sa capacité à innover et à diversifier ses domaines d'expertise.

Grâce à son engagement envers l'innovation et l'accompagnement des jeunes entrepreneurs,

3D SMART FACTORY a contribué de manière significative à la croissance et au progrès technologique au Maroc.

1.1.2 Fiche technique

Dénomination sociale	3D Smart Factory
Siège social	Mohammedia, Mohammedia
Fondée en	2018
Type	Société civile/Société commerciale/Autres types de sociétés
Téléphone	0523300446
Email	3dsmartfactory@gmail.com
Adresse	Villa N 75 lotissement la gare, Mohammedia, Mohammedia 20800, MA

Table 1.1 – Fiche technique de l'entreprise

1.2 La Problématique et les Objectifs du Projet

1.2.1 La Problématique

L'analyse des articles scientifiques est un enjeu majeur dans les domaines académiques et de recherche. Les utilisateurs, qu'ils soient étudiants, chercheurs ou professionnels, sont souvent confrontés à la nécessité de traiter un volume important de littérature scientifique pour en extraire des informations importantes, ce qui peut s'avérer compliqué et chronophage. Cette complexité souligne le besoin d'outils capables d'automatiser et d'améliorer l'extraction d'informations. L'émergence des techniques de traitement du langage naturel (NLP), des modèles de langage avancés (LLM) et de la technologie de génération augmentée de récupération (RAG) ouvre de nouvelles méthodes pour relever ce défi en optimisant le processus d'analyse documentaire.

Dans ce contexte, ce projet vise à mettre en œuvre une solution élaborée à l'aide de ces technologies. Cette solution doit permettre à ses utilisateurs de trouver l'information désirée en temps

réel à l'aide d'une RAG qui sera utilisée pour la récuperation d'informations, et un LLM pour la communication de ces informations.

1.2.2 les Objectifs du Projet

Dans le cadre de ce projet, l'objectif est de créer une solution complète qui simplifie l'accès aux informations recherchées dans les articles scientifiques. Les principaux objectifs à atteindre sont les suivants :

— Développer une interface utilisateur intuitive pour faciliter l'interaction avec le système.

— Mettre en œuvre une solution utilisant un système RAG (Récupération Augmentée par Génération) pour extraire les informations.

— Utiliser un grand modèle de langage (LLM) pour communiquer ces informations d'une manière cohérente.

— Assurer la capacité du système à fournir des réponses en temps réel.

1.3 État de l'art

1.3.1 Théories et concepts clés

1.3.1.1 Traitement Automatique des Langues (NLP)

Le traitement automatique des langues, en anglais natural language processing (NLP), est un domaine multidisciplinaire qui combine la linguistique, l'informatique et l'intelligence artificielle. Son objectif principal est de créer des outils capables de traiter le langage naturel pour diverses applications, allant de la traduction automatique aux chatbots en passant par l'analyse de texte et la synthèse vocale.[1]

Le traitement automatique du langage naturel (NLP) a débuté dans les années 1950, principalement avec des projets de traduction automatique motivés par la guerre froide. Depuis cette période, divers développements et évolutions ont eu lieu dans ce domaine, mais ils n'ont pas toujours été pleinement satisfaisants. Cependant, depuis les années 2010, le NLP a fait des progrès significatifs grâce aux avancées en intelligence artificielle et en apprentissage automatique. En 2018, des modèles développés par Microsoft et Alibaba ont surpassé les humains dans des tests de lecture et de compréhension. Dans cette même année, Google a lancé BERT, un modèle de langage pré-entraîné bidirectionnel, et en 2020, OpenAI a annoncé GPT-3, un modèle avec

175 milliards de paramètres. Ces innovations ont considérablement amélioré la capacité des machines à comprendre et générer du langage naturel, ouvrant la voie à des applications avancées comme les chatbots sophistiqués et l'analyse de grandes quantités de données textuelles. [1]

1.3.1.2 Modèles de langage de grande taille (LLM)

Un grand modèle de langage (LLM, pour large language model en anglais) est un modèle de langage possédant un grand nombre de paramètres, généralement de l'ordre d'un milliard ou plus. Ces modèles sont des réseaux de neurones profonds entraînés sur de grandes quantités de texte non étiqueté en utilisant l'apprentissage auto-supervisé ou semi-supervisé. Les LLM ont émergé vers 2018 et sont utilisés pour des applications telles que les agents conversationnels. Les LLM excellent dans une variété de tâches, car ils ne sont pas spécifiquement entraînés pour une seule tâche, mais pour prédire une suite probable à une entrée donnée. La qualité de leur sortie dépend de la quantité de ressources (taille des paramètres, puissance de calcul, données) et de la qualité des données fournies. Ces modèles capturent une grande partie de la syntaxe et de la sémantique du langage humain démontrant une connaissance générale considérable.[2]

Les grands modèles de langage ont plusieurs applications : [3]

— **Récupération d'informations** : Ils sont utilisés par des moteurs de recherche comme Bing ou Google pour fournir des réponses en récupérant et résumant des informations pertinentes.

— **Analyse des sentiments** : Ils permettent aux entreprises d'analyser le sentiment exprimé dans les données textuelles.

— **Génération de textes** : Des modèles peuvent créer du texte à partir de requêtes spécifiques, telles que rédiger un poème dans un style particulier.

— **Génération de code** : Ils peuvent également générer du code en comprenant des schémas de programmation.

— **Chatbots et IA conversationnelle** : Ils permettent aux chatbots et aux systèmes d'IA de communiquer avec les clients, d'interpréter leurs demandes, et de fournir des réponses appropriées.

Avec un tel éventail d'applications, les grands modèles de langage se retrouvent dans une multitude de domaines : [3]

— **Technologie** : Ils permettent aux moteurs de recherche de répondre aux requêtes, aident les développeurs à écrire du code, et offrent de nombreuses autres utilités.

— **Santé et science** : Ils sont capable de saisir les détailles de la médicine et ils sont utilisée dans l'amélioration des traitements et l'invention. Les LLM sont également utilisés comme chatbots médicaux pour les admissions de patients ou des diagnostics de base.

— **Service client** : Utilisés sous forme de chatbots ou d'IA conversationnelle dans divers secteurs pour améliorer le service client.

— **Marketing** : Les équipes marketing exploitent les LLM pour réaliser des analyses de sentiments, générer des idées de campagnes, des textes comme des synopsis, et bien plus encore.

— **Domaine juridique** : Ils aident à la recherche dans de grands ensembles de données textuelles et à la génération de jargon juridique pour les avocats, assistants juridiques, et le personnel juridique.

— **Services bancaires** : Les LLM sont utilisés pour aider les institutions bancaires à détecter les fraudes.

Traditionnellement, les réseaux de neurones récurrents étaient la norme pour le traitement des données séquentielles, mais leur capacité limitée à paralléliser les calculs posait des difficultés. En 2017, l'introduction des transformeurs a marqué une avancée majeure. Cette nouvelle architecture a permis une parallélisation efficace des calculs et a considérablement augmenté le nombre de paramètres des modèles, entraînant des gains de performance notables. Aujourd'hui, la majorité des grands modèles de langage reposent sur cette architecture.

1.3.1.3 Transformateurs

Le Transformeur, introduit en 2017, est un modèle d'apprentissage profond qui excelle dans l'apprentissage du contexte et du sens en analysant les relations dans des données séquentielles, telles que les mots dans une phrase ou les images dans une vidéo. Sa caractéristique principale est le mécanisme d'attention, qui lui permet de se concentrer sur différentes parties d'une séquence d'entrée lors de la réalisation de prédictions. Structuré sous la forme d'une architecture encodeur-décodeur, le modèle Transformer se compose d'un encodeur qui mappe une séquence d'entrée en représentations continues et d'un décodeur qui génère une séquence de sortie basée sur la sortie de l'encodeur et ses sorties précédentes. [4]

Différentes classes de grands modèles de langage ont émergé basées sur le Transformeur, chacune étant adaptée à des cas d'utilisation spécifiques [5] :

— Les modèles uniquement encodeurs, comme BERT (Bidirectional Encoder Representations from Transformers), sont bien adaptés aux tâches nécessitant la compréhension du langage, telles que la classification et l'analyse des sentiments.

— Les modèles uniquement décodeurs, comme GPT-3 (Generative Pretrained Transformer 3), excellent dans la génération de langage et de contenu, les rendant appropriés pour des tâches comme la rédaction d'histoires et la génération de blogs.

— Les modèles encodeur-décodeur, tels que T5 (Text-to-Text Transformer), combinent les composants encodeur et décodeur pour comprendre et générer du contenu. Cette architecture est particulièrement efficace pour des tâches comme la traduction et le résumé.

1.3.1.4 RAG

Les LLMs sont particulièrement performants dans la création des réponses textuelles basées sur un vaste ensemble de données d'entrainement. Cependant, les LLMs génèrent parfois des affirmations fausses qui ne semblent pas être justifiées par leurs données d'apprentissage, on parle alors d'« hallucination » [6]. En plus, les informations utilisées pour générer ces réponses sont limitées à ces données. Ces données peuvent être obsolètes et le processus de construction d'un nouveau LLM généralisé est long et coûteux.

Dans ce contexte, le besoin pour la génération augmentée de récupération (RAG) est apparu. La RAG permet d'optimiser les résultats d'un LLM en lui fournissant des informations ciblées. Ces informations peuvent être plus récentes que celles du LLM et spécifiques à un secteur particulier. Ainsi, le système d'IA générative peut fournir des réponses contextuellement appropriées aux invites, en se basant sur des données plus récentes et plus précises.

La RAG a vue le jour après la publication de « Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks », un article de Patrick Lewis et une équipe de Facebook AI Research en 2020. Depuis que le concept de la RAG a été adopté, la valeur des systèmes d'IA générative a augmenté considérablement.[7]

Le flux de travail de la génération augmentée par la récupération (RAG) comprend les étapes suivantes :[8]

- Récupérer : La requête de l'utilisateur est utilisée pour récupérer un contexte pertinent à partir

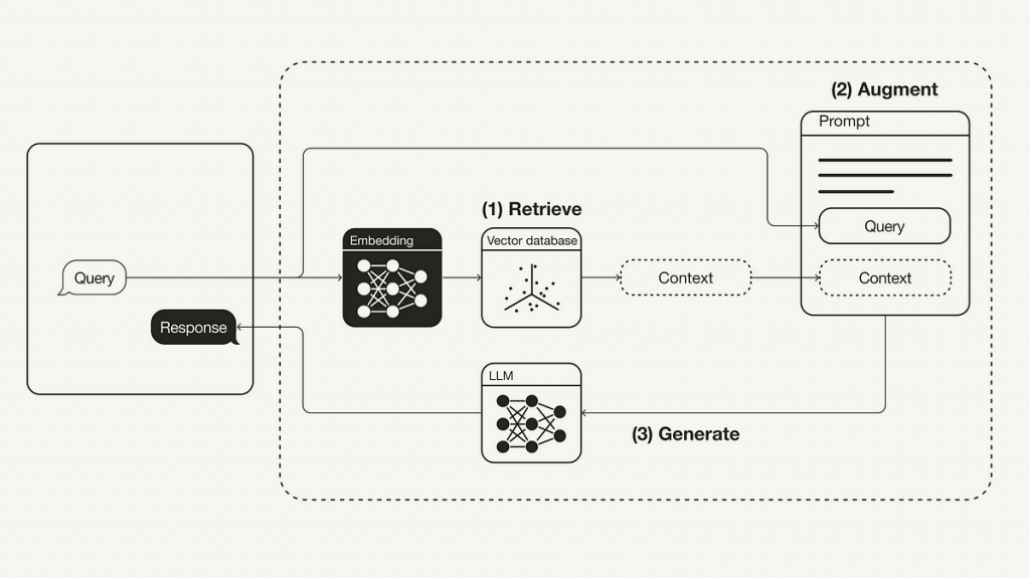


FIGURE 1.1 – Illustartion du flux de travail de la génération augmentée par la récupération (RAG). [8]

d'une source de connaissance externe. Pour cela, la requête de l'utilisateur est transformée en vecteur à l'aide d'un modèle d'embedding dans le même espace vectoriel que le contexte supplémentaire contenu dans la base de données vectorielles. L'embedding est une méthode de représentation d'objets tels que du texte, des images et de l'audio sous forme de points dans un espace vectoriel continu, où la position de ces points est sémantiquement significative pour les algorithmes de machine learning [10]. Cela permet d'effectuer une recherche de similarité (Distance Cosinus, Euclidienne, ...), et les k objets de données les plus similaires dans la base de données vectorielles sont retournés.

- Augmenter : La requête de l'utilisateur et le contexte supplémentaire récupéré sont intégrés dans un prompt.

- Générer : Enfin, le prompt est transmis au modèle de langage (LLM) pour générer une réponse.

1.3.1.5 les types de RAG

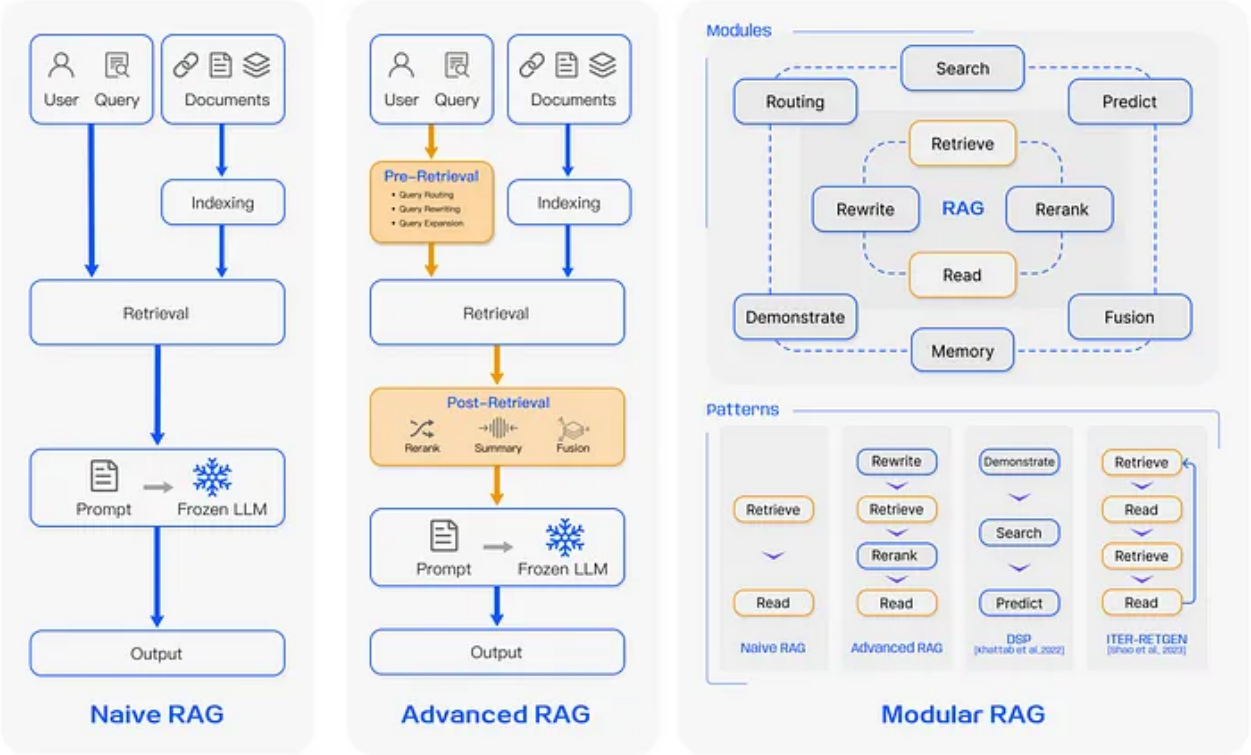


FIGURE 1.2 – Différents types de RAG [11]

RAG Naïve

Le RAG naïf c'est la première méthodologie adoptée pour répondre au problème des données d'entrainement obsolètes et générales. Le RAG naïf suit un processus comprenant l'indexation, la récupération, l'augmentation et la génération de réponses. Voici les étapes en détail :

— **Indexation** : Préparation des données en extrayant et nettoyant les informations de sources comme des fichiers ou des URLs, puis en les convertissant en texte brut et en les divisant en morceaux plus petits. Ces morceaux sont transformés en vecteurs à l'aide de modèles d'embedding.

— **Récupération** : Encodage de la requête utilisateur en vecteur et recherche des morceaux les plus similaires dans les embeddings de données, utilisant des méthodes comme la similarité cosinus, le produit scalaire, etc.

— **Génération** : Fusion des morceaux récupérés avec la requête utilisateur et les instructions, puis génération de la réponse par le LLM.[9]

Inconvénients de la RAG Naïf

Le RAG naïf présente des inconvénients, tels que la faible précision lors de la récupération des morceaux pertinents, la présence de données obsolètes, et la réponse générée peut ne pas être basée sur le contexte supplémentaire fourni.[9]

RAG Avancée

Le RAG avancé a été développé pour surmonter les défauts de la RAG naïve. En plus des étapes du RAG naïve, il inclut deux techniques en plus :

— **La Pré-récupération** : Amélioration de la qualité du contenu indexé en supprimant les informations non pertinentes, en mettant à jour les documents obsolètes, et en ajoutant des métadonnées appropriées aux morceaux.

— **La Post-récupération** : Techniques comme le *re-ranking* (réorganisation des morceaux selon leur similarité contextuelle) et la compression des *prompts* (réduction du bruit et mise en avant des passages importants).[9]

RAG Modulaire

Le RAG modulaire se distingue du RAG naïf par l'introduction de fonctionnalités améliorées. Il intègre un module de recherche pour la récupération de similarités et adopte une approche de *fine-tuning* dans le processus de récupération. Les techniques comprennent :

— **Recherche Hybride** : Intégration de recherches basées sur des mots-clés, sémantiques et vectorielles pour une récupération cohérente et contextuelle.

— **Récupération Récursive et Moteur de Requêtes** : Acquisition de petits morceaux au début pour capturer les significations sémantiques clés, fournissant ensuite des morceaux plus larges avec plus d'informations contextuelles.

— **Approche StepBack** : Encouragement du LLM à raisonner autour de concepts plus larges, améliorant les performances dans les tâches d'inférence.

— **Sous-requêtes** : Utilisation de diverses stratégies de requêtes comme les requêtes en arbre ou les requêtes vectorielles selon le scénario.

— **Embeddings de Documents Hypothétiques** : Création d'un document hypothétique en réponse à une requête et utilisation de son embedding pour récupérer des documents

réels similaires.[9]

1.3.2 Travaux antérieurs

Des travaux sur les systèmes de questions-réponses basés sur le traitement du langage naturel (NLP) et la génération augmentée par récupération (RAG) ont été implémentés dans divers secteurs. Un article intitulé "Automated Evaluation of Retrieval-Augmented Language Models with Task Specific Exam Generation" examine l'utilisation des systèmes RAG, qui combinent la récupération d'informations et la génération de réponses par des grands modèles de langage (LLM).

L'objectif principal de cet article est d'optimiser le système RAG pour des tâches complexes comme les questions à choix multiples dans des domaines spécialisés. Les auteurs utilisent des données provenant de secteurs variés, tels que les documents financiers et les questions techniques, pour l'évaluation en combinant différents mécanismes de récupération (modèles denses, sparses, hybrides) et des modèles LLMs comme Mistral et Llama2.

Les résultats montrent que la performance varie selon la combinaison des modèles de récupération et des LLMs. Par exemple, dans les domaines techniques, les systèmes hybrides ont surpassé les autres. Un modèle qui a un accès direct aux documents originaux, est resté le plus performant, soulignant l'importance de la qualité des informations récupérées dans les sytèmes de RAG.

En conclusion, cet article vise à optimiser un système RAG en évaluant différentes combinaisons de technologies de récupération d'informations et de génération de texte pour des tâches spécifiques.

1.4 Le plan du projet

La réalisation de ce projet comprend six phases. La durée du projet est de deux mois, allant du 15 juillet 2024 au 16 septembre 2024. Le tableau ci-dessous présente les différentes tâches du projet, leurs dates de début et de fin, ainsi que leur durée :

ID	Nom	Date de début	Date de fin	Durée
1	Compréhension et Définition des objectifs du projet	15 juillet 2024	22 juillet 2024	8 jours
2	États de l'art et Recherche bibliographique	23 juillet 2024	1 août 2024	10 jours
3	Analyse fonctionnelle et Architecture du système	2 août 2024	12 août 2024	11 jours
4	Les Choix d'Implémentations	13 août 2024	19 août 2024	7 jours
5	La réalisation	20 août 2024	5 septembre 2024	17 jours
6	L'évaluation	6 septembre 2024	16 septembre 2024	11 jours

La figure suivante montre le diagramme de Gantt du projet :

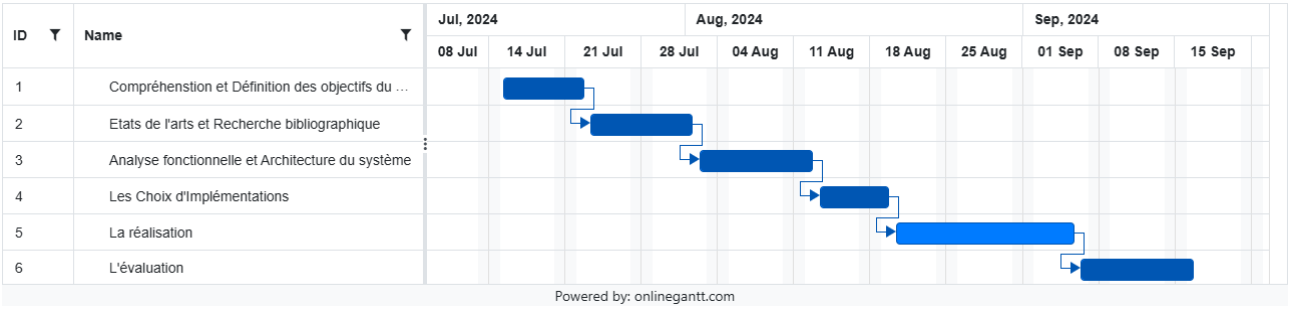


FIGURE 1.3 – diagramme de Gantt

Conclusion

En conclusion, ce chapitre a permis d'établir une base solide pour comprendre le cadre dans lequel ce projet s'inscrit. En identifiant la problématique et les objectifs associés, ainsi qu'en explorant les concepts théoriques et les travaux antérieurs. Le plan du projet présenté à la fin de ce chapitre servira de guide pour les phases à venir.

Chapitre 2

Analyse et Conception

Introdution

Ce chapitre de l'analyse et la conception commence par une analyse fonctionnelle qui permet de définir les exigences fonctionnelles et non-fonctionnelles du projet. Ensuite, l'architecture globale du système qui servira comme un guide dans la réalisation. Et se termine par les choix technologiques effectués pour l'implimentation ainsi que la dataset employés pour l'évaluation de ce projet.

2.1 Analyse Fonctionnelle

Nous cherchons à créer une application conçue en tant que MVP (Minimum Viable Product), c'est-à-dire une version minimale mais fonctionnelle, offrant juste assez de fonctionnalités pour attirer l'intérêt des utilisateurs et se différencier des autres applications similaires. Cette application doit respecter des exigences fonctionnelles et non fonctionnelles qui seront expliquées dans cette partie.

2.1.1 Exigences Fonctionnelles

1. **Collecte et Prétraitement des Données :** Les données à collecter sont des données textuelles extraites des fichiers PDF qui représentent nos articles scientifiques.
2. **Construction de la Base de Connaissances :** La base de connaissances sera construite en utilisant des techniques d'embedding pour convertir les données textuelles en vecteurs.
3. **RAG (Retrieval-Augmented Generation) :** La mise en œuvre d'une approche RAG pour récupérer les données à partir de la base de données textuelle en fonction des requêtes des utilisateurs.

4. **LLM (Large Language Model) :** Intégration d'un modèle LLM pré-entrainé pour générer des réponses cohérentes basées sur les informations récupérées par le système RAG.
5. **Construction de l'Interface Utilisateur :** Développement d'une interface utilisateur pour le chatbot permettant aux utilisateurs de poser des questions et de recevoir des réponses générées. L'interface doit être intuitive et facile à utiliser.

2.1.2 Exigences Non-Fonctionnelles

1. **Performance :** Le système doit être capable de traiter efficacement un volume important de documents afin d'assurer la pertinence des réponses, et de générer ces réponses dans un délai raisonnable.
2. **Précision :** Les réponses générées doivent être suffisamment précises pour refléter correctement les informations des articles scientifiques.
3. **Fiabilité :** Le système doit être capable de gérer efficacement les erreurs et les exceptions, et avoir un fonctionnement stable et continue.
4. **Maintenabilité :** Le système doit être facile à maintenir, avec un code bien documenté.
5. **Utilisabilité :** L'interface utilisateur doit être conviviale, avec une navigation claire et une interaction facile pour les utilisateurs.

2.2 Architecture du Systeme

Le système doit répondre aux exigences fonctionnelles et non fonctionnelles, et le produit final doit être conçu comme un MVP (Minimum Viable Product). L'utilisateur soumet d'abord une requête via une interface, qui permet également d'afficher les réponses et de suivre l'historique des conversations. Avant de procéder à la récupération des informations, la requête est d'abord améliorée par le modèle de langage (LLM) pour reformuler la requête. Ensuite, cette requête optimisée est utilisée pour interroger la base de données vectorielles des articles scientifiques. Ces derniers sont initialement collectés dans un dossier sous format PDF et puis traités et stockés dans la base de données vectorielles. Le système de récupération sélectionne alors les phrases les plus pertinentes en fonction de la requête améliorée. Ces informations récupérées sont ensuite combinées avec cette requête pour former un prompt final, lequel est renvoyé au

LLM pour générer une réponse cohérente. Enfin, la réponse est affichée à utilisateur. Le schéma suivant illustre ces étapes :

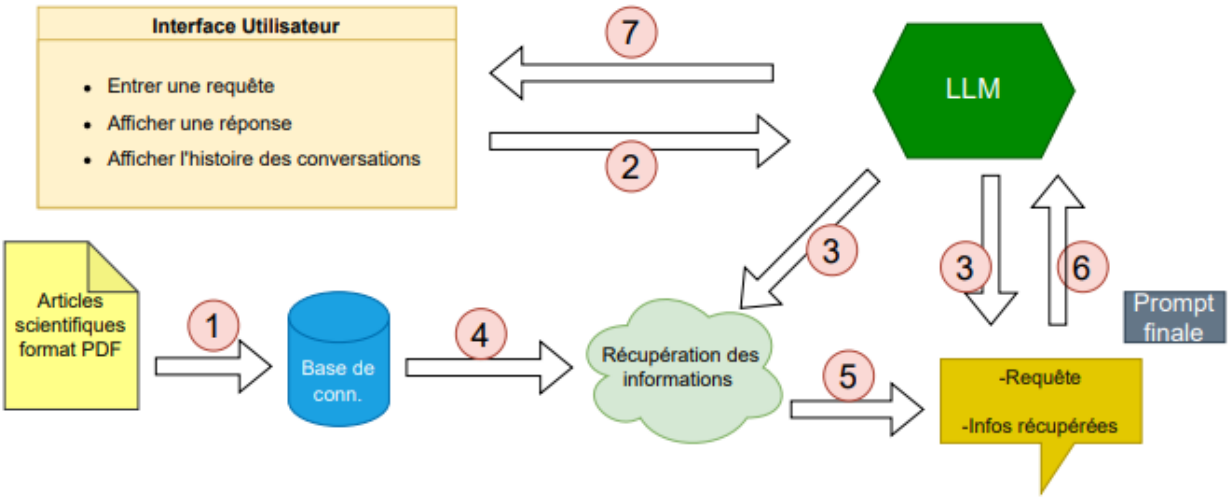


FIGURE 2.1 – Image illustrant l'architecture de l'application

2.3 Choix d'implémentation

2.3.1 Modèle d'Embedding

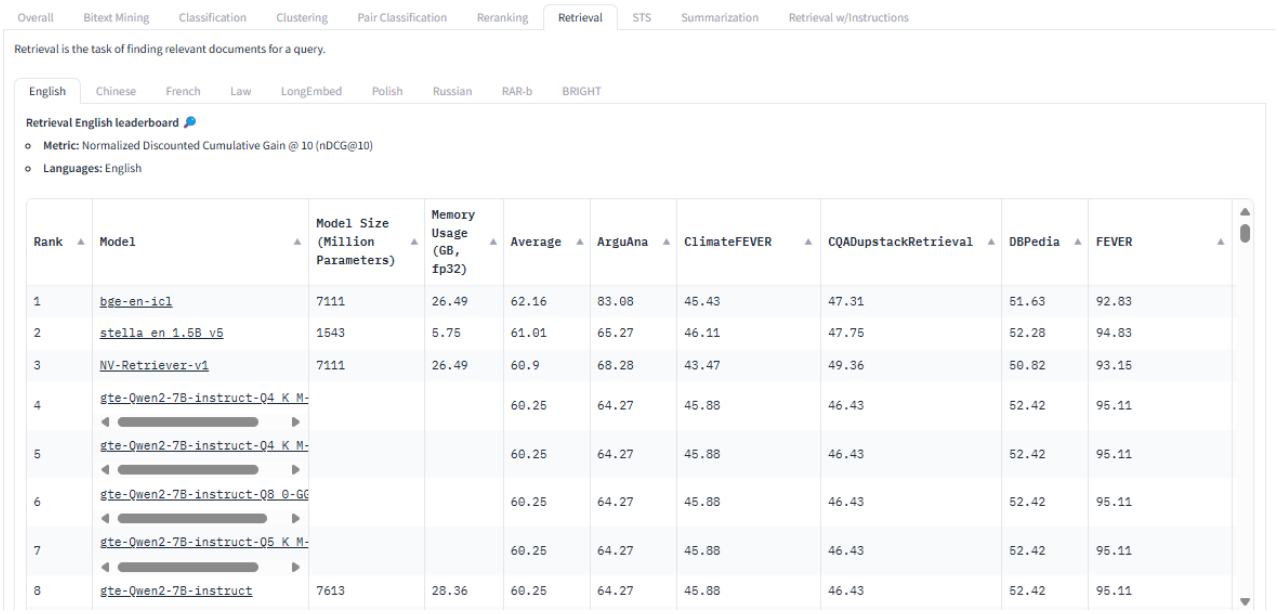
Le fine-tuning des modèles d'embeddings généraux nécessite des ressources matérielles importantes, et SciBERT est le seul modèle d'embeddings dédié aux textes scientifiques disponible. Il y a un emplacement où on peut consulter le benchmarking des différents modèles d'embedding de Hugging Face.

Hugging Face est une plateforme spécialisée dans les modèles de traitement du langage naturel (NLP) et l'apprentissage automatique. Elle offre des bibliothèques open-source comme Transformers pour travailler avec des modèles pré-entraînés tels que BERT, GPT, et T5. Les utilisateurs peuvent accéder à une large gamme de modèles via la plateforme les intégrer facilement dans leurs projets. En plus, Hugging Face propose des outils pour l'entraînement, la gestion des modèles et des API pour déployer des solutions NLP.

FIGURE 2.2 – Logo de Hugging Face

Pour l'emplacement du benchmarking des modèles d'embedding :

https ://huggingface.co/spaces/mteb/leaderboard ?ref=thepickool.com.



Overall | Bitext Mining | Classification | Clustering | Pair Classification | Reranking | Retrieval | STS | Summarization | Retrieval w/Instructions

Retrieval is the task of finding relevant documents for a query.

English | Chinese | French | Law | LongEmbed | Polish | Russian | RAR-b | BRIGHT

Retrieval English leaderboard

- Metric: Normalized Discounted Cumulative Gain @ 10 (nDCG@10)
- Languages: English

Rank	Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Average	ArguAna	ClimateFEVER	CQADupstackRetrieval	DBPedia	FEVER
1	bge-en-icl	7111	26.49	62.16	83.08	45.43	47.31	51.63	92.83
2	stella_en_1.5B_v5	1543	5.75	61.01	65.27	46.11	47.75	52.28	94.83
3	NV-Retriever-v1	7111	26.49	60.9	68.28	43.47	49.36	50.82	93.15
4	gte-Qwen2-7B-instruct-Q4_K_M-			60.25	64.27	45.88	46.43	52.42	95.11
5	gte-Qwen2-7B-instruct-Q4_K_M-			60.25	64.27	45.88	46.43	52.42	95.11
6	gte-Qwen2-7B-instruct-Q8_0-GG			60.25	64.27	45.88	46.43	52.42	95.11
7	gte-Qwen2-7B-instruct-Q5_K_M-			60.25	64.27	45.88	46.43	52.42	95.11
8	gte-Qwen2-7B-instruct	7613	28.36	60.25	64.27	45.88	46.43	52.42	95.11

FIGURE 2.3 – Benchmark des modèles d'embedding de HuggingFace

En utilisant ce benchmarking de modèles, on peut choisir le modèle d'embedding générale qui convient selon les ressources de memoire diponibles.

2.3.2 Base de données vectorielles

L'embedding des documents PDF sera enregistré dans une base de données vectorielles avec le texte correspondant à chaque vecteur et ses métadonnées. La technologie open source la plus connue est Chroma. Chroma est une bibliothèque open-source pour le stockage et la recherche de vecteurs d'embeddings. Elle permet de gérer efficacement de grandes quantités de données vectorielles, offrant des fonctionnalités pour la recherche rapide et la récupération d'informations basée sur la similarité des vecteurs.

FIGURE 2.4 – Logo de Chroma

2.3.3 LLMs

Le choix d'un LLM pour la correction de la requête de l'utilisateur et la génération de la réponse dépend de plusieurs facteurs : frais de l'utilisation, les capacités de l'ordinateur, les performances de LLM et la taille de sa fenêtre contextuelle (nombre de jetons d'entrée). En se basant sur le site web suivant : https ://www.vellum.ai/llm-leaderboard.

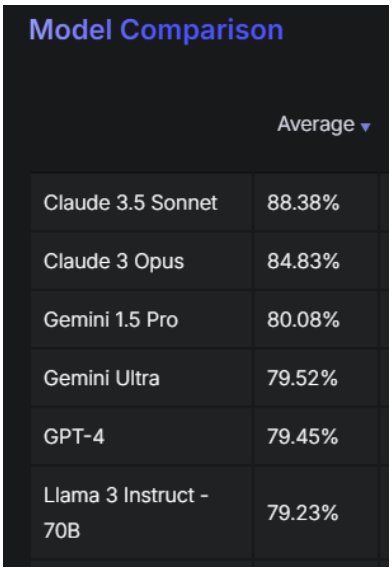


FIGURE 2.5 – La comparaison des performances des differents modèles

cette image illustre le score moyen de chaque modèle basé sur un test couvrant différentes compétances .

FIGURE 2.6 – La limite de débit de Gemini-1.5-pro.

Cette deuxième image, capturée de la documentation officielle de Google, montre la limite

de débit de l'utilisation de l'API de Gemini-1.5-pro.

On peut clairement constater que Gemini-1.5 Pro est un LLM performant, avec un débit gratuit de 32 000 jetons par minute, représentant le nombre de jetons en entrée plus ceux en sortie, ce qui équivaut à environ 24 000 mots. De plus, il permet d'envoyer deux requêtes par minute et 50 requêtes par jour. Pour la phase de test et d'évaluation, ces chiffres sont convacantes, ainsi que Gemini-1.5 Pro semble un choix raisonnable.

2.3.4 Language de programmation

Python est un langage de programmation interprété de haut niveau. Il utilise le typage dynamique et prend en charge la programmation procédurale ainsi que la programmation orientée objet. Python est utilisé dans le développement web, le développement d'interfaces graphiques, l'analyse de données, et le machine learning. [12]

FIGURE 2.7 – Python logo [12]

Ce projet implique de travailler sur diverses tâches complexes, notamment avec les nouvelles technologies telles que les grands modèles de langage (LLMs) et les modèles d'embeddings. Par conséquent, choisir Python comme langage de programmation semble plus adapté. La flexibilité de Python, sa facilité d'utilisation, et son riche écosystème de bibliothèques le rendent bien approprié. De plus, Python bénéficie d'un solide soutien communautaire, garantissant un accès à une documentation exhaustive et à de l'assistance en cas de défis rencontrés lors du développement.

2.3.5 Libraries

2.3.5.1 Langchain

LangChain est un framework open-source dédié à l'intégration des modèles de langage. Il est conçu pour simplifier l'utilisation de ces modèles dans les applications. Il inclut presque toutes les bibliothèques nécessaires à l'implémentation de systèmes RAG. [13]

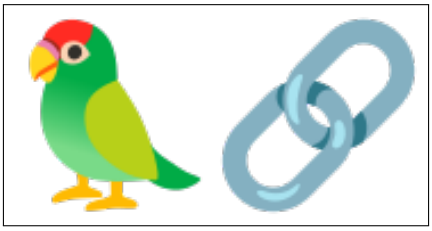


FIGURE 2.8 – Langchain logo [13]

2.3.5.2 PyQt6

PyQt6 est une bibliothèque open-source qui permet le développement d'applications graphiques en Python. Conçue pour simplifier la création d'interfaces utilisateur complexes, elle fournit des outils et des widgets pour la construction d'applications modernes.[14]

2.3.5.3 RAGAS

RAGAS (Retrieval-Augmented Generation Evaluation Toolkit) est un ensemble d'outils conçu pour évaluer la performance des systèmes basés sur la génération augmentée par récupération (RAG). RAGAS permet de mesurer l'efficacité de la récupération d'informations pertinentes et la qualité des réponses générées par les grands modèles de langage (LLM). Il fournit des métriques d'évaluation pour mieux comprendre les performances des systèmes RAG dans des scénarios réels[15].

FIGURE 2.9 – RAGAS Logo [15]

2.4 Dataset de documents

Ce projet cherche à créer un système qui permet de récupérer et de bien présenter des informations scientifiques issues de nouveaux documents et articles scientifiques. Le système à développer va fonctionner sur n'importe quel ensemble de données d'articles scientifiques constitués principalement de texte. Pour l'évaluation, et étant donné que j'explore le concept des systèmes de RAG, l'ensemble de données sera constitué de documents scientifiques partageant la même thématique, à savoir la RAG.

Conclusion

En conclusion, ce chapitre a permis de détailler les fondations conceptuelles et techniques sur lesquelles repose le projet. L'analyse fonctionnelle a clarifié les attentes vis-à-vis du système, tandis que l'architecture du système et les choix technologiques ont mis en évidence la structure générale de l'application et les outils sélectionnés pour le développement.

Chapitre 3

Réalisation

Introdution

Ce chapitre présente la phase de réalisation du projet, dans laquelle les différentes étapes de développement du système sont détaillées. Il commence par le prétraitement des documents, qui seront ensuite utilisés pour la mise en place de la base de données vectorielle. Ensuite, il aborde la gestion de la requête utilisateur ainsi que le développement de l'interface utilisateur. Enfin, il traite la méthode d'évaluation des performances du modèle.

3.1 Data pré-processing

La source des informations à récupérer est constituée de fichiers PDF. La tâche de prétraitement consiste à extraire tout le texte de chaque PDF, à le découper en phrases complètes, puis à assigner à ces phrases des identifiants uniques. Ces identifiants permettront d'identifier les phrases dans la base de données vectorielles afin d'éviter de stocker plusieurs fois les mêmes phrases. Un identifiant aura la forme suivante :

CHEMIN_PDF :NUMERO_PAGE :PHRASE_INDICE

— CHEMIN_PDF : Il s'agit du chemin relatif du fichier PDF.

— NUMERO_PAGE : C'est le numéro de la page du PDF où se trouve la phrase.

— PHRASE_INDICE : C'est le numéro d'ordre de la phrase dans la page, en commençant par la première phrase en haut de la page.

3.2 Base de données vectorielles

La base de données vectorielles est implémenté à l'aide de Chroma. À chaque lancement du programme, le prétraitement des données est relancé. Le programme récupère les identifiants des phrases, puis les comparent avec les identifiants déjà stockés dans la base de données vectorielles. De cette façon, on peut détecter les nouveaux fichiers PDF, minimisant ainsi le temps nécessaire au calcul des embeddings et l'opération du stockage. Pour la tâche de récupération, on utilise la méthode suivante appliquée sur l'instance de connexion à la base de données vectorielles :

```
db.asimilarity_search_with_relevance_scores(query_enhanced, k = 30)
```

Avec :

— query_enhanced : la requête de l'utilisateur, améliorée à l'aide du LLM.

— k : le nombre maximal de phrases à récupérer.

Cette méthode retourne une liste de 30 tuples (score, langchain.document). Le score est calculé par défaut à l'aide de la distance cosinus, donnant ainsi un résultat compris entre 0 et 1. Ensuite, la liste est ordonnée selon le score, pour obtenir les 15 meilleurs résultats qui seront ensuite ajoutés au prompt du LLM avec la requête améliorée de l'utilisateur pour la génération de la réponse.

3.3 gérer la requête de l'utilisateur

Au moment où l'utilisateur entre sa requête, un premier prompt est construit comme suit :

```
"""
Add relevant terms to the following query to enhance its clarity and
relevance, and only give the enhanced query as response: "{user_input}"
"""
```

Ce prompt permet d'améliorer la requête de l'utilisateur en ajoutant des termes pertinents afin d'améliorer la récupération basée sur le calcul de la distance de similarité.

Ensuite, la réponse du LLM (gemini-1.5-pro) est utilisée pour récupérer les informations de la base de données vectorielles. Après cette récupération, un second prompt est construit :

```
"""
    You are an assistant for question-answering tasks.
    If you don't know the answer, just say that you don't know.
    Answer the following question concisely:

    Question: {enhanced_query}

    Use the information provided below to craft your response:

    {retrieved_infos}
    """
```

Finalement, la réponse du LLM à ce prompt sera affichée à l'utilisateur sur l'interface.

3.4 Interface Utilisateurs

L'interface utilisateur est conçue pour afficher la conversation entre l'utilisateur et le système de RAG. Elle comprend une zone de saisie pour écrire et envoyer des requêtes. Lors de l'envoi d'une requête, le système vérifie la connexion Internet. Si la connexion est détectée, le traitement de la requête continue. Sinon, un message indiquant l'absence de connexion Internet est affiché. À gauche, l'historique des conversations de l'utilisateur est affiché, avec les échanges enregistrés dans des fichiers texte. Lorsqu'une conversation est sélectionnée, son contenu est chargé et affiché dans la zone de discussion.

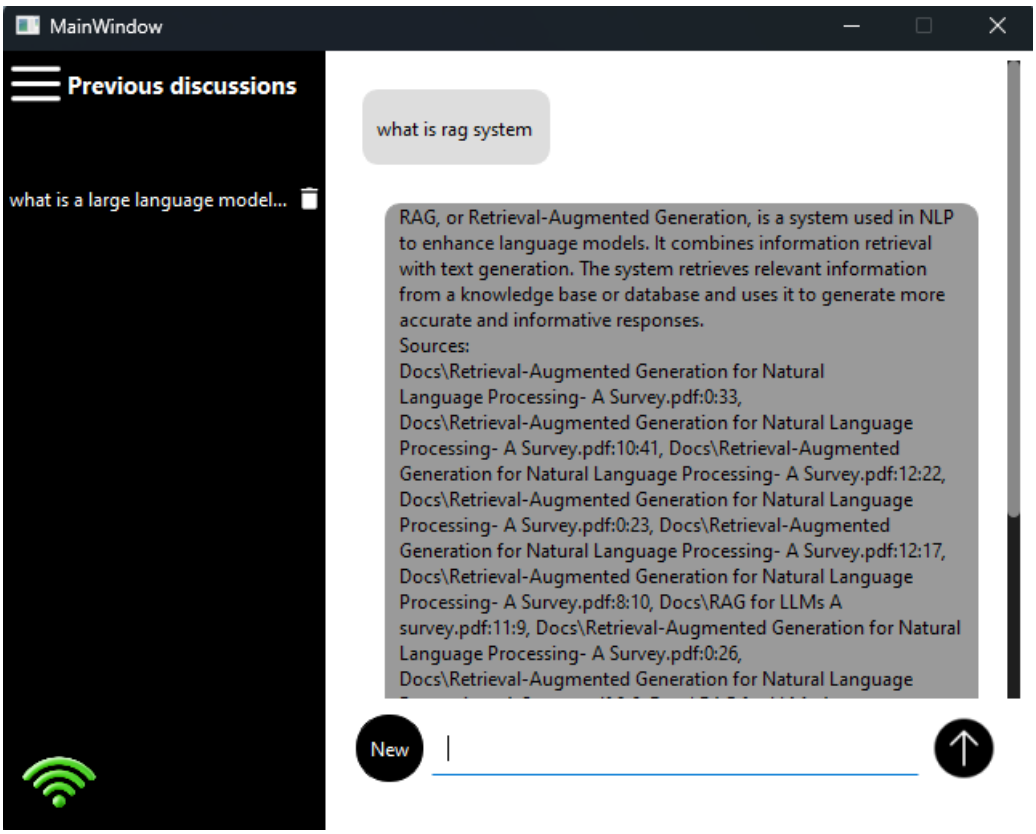


FIGURE 3.1 – Conversation de l'utilisateur avec le système.

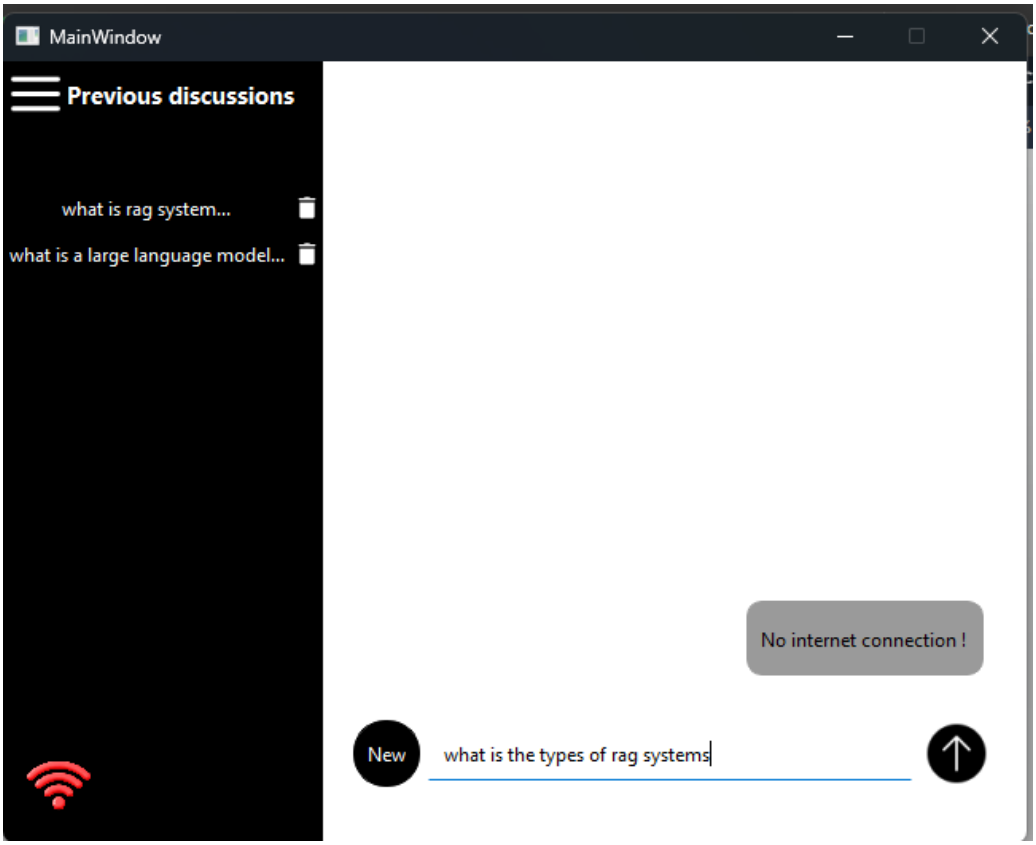


FIGURE 3.2 – L'utilisation sans connection internet du système.

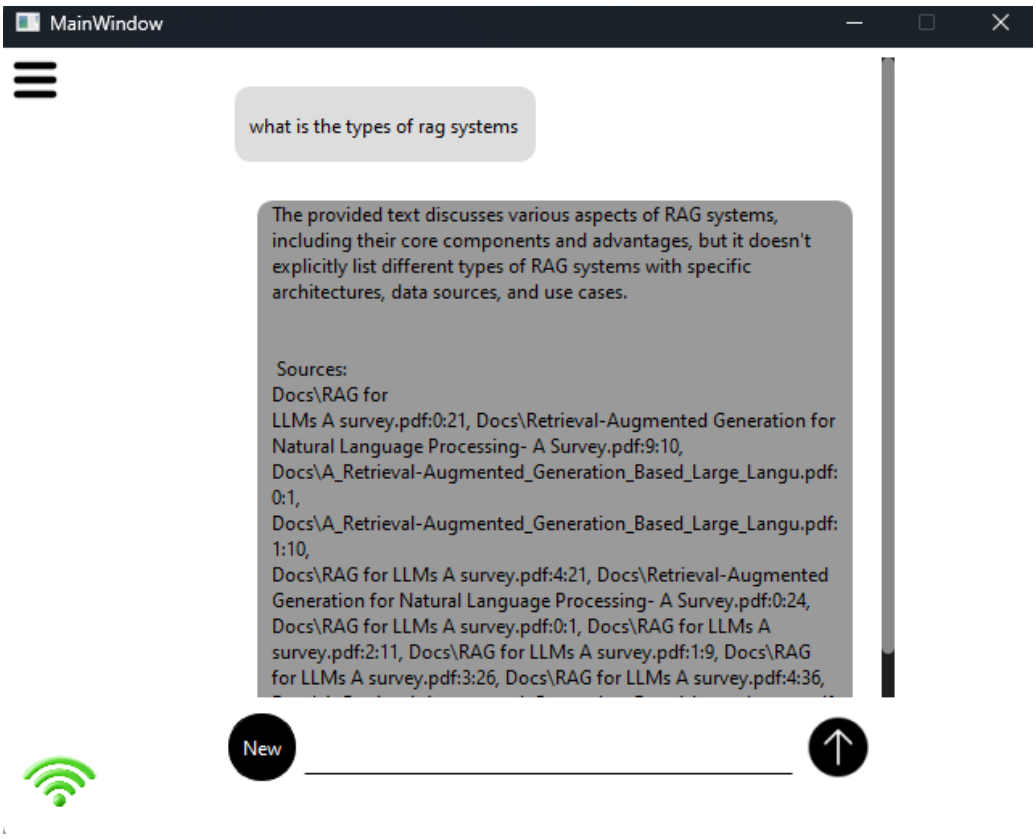


FIGURE 3.3 – Barre des discussions cachée

3.5 Méthode d'évaluation

L'évaluation des systèmes de RAG est une étape compliquée et peu claire. Les performances des systèmes de RAG dépendent principalement du modèle d'embeddings et du LLM qui génère la réponse finale. Malheureusement, l'utilisation de la librairie RAGAS, conçue pour évaluer les systèmes de RAG, n'était pas possible. J'ai rencontré une exception "Exception raised in Job[X] : TimeoutError()" pour laquelle je n'ai pas trouvé de solution.

J'ai donc appliqué une autre méthode d'évaluation qui consiste à préparer un ensemble de questions et leurs réponses correctes à partir des fichiers PDF, puis à générer des réponses à partir du système RAG. Ensuite, utiliser le LLM pour évaluer la similarité entre la réponse du système RAG et la vérité, ainsi que la pertinence des informations récupérées par rapport à la vérité. Le prompt utilisé est le suivant :

```
"""
How similar is the answer to the ground truth? Respond with: not similar,
similar, or very similar.
```

```
How relevant are the contexts to the ground truth? Respond with: not
relevant, relevant, or very relevant.
ground_truth: "{ground_truth}"
answer: "{answer}"

contexts: "{contexts}"

Your answer should be in this format: [similarity, relevance]
"""
```

Les résultats obtenus sont une liste de [similarité, pertinence] qui acceptent les valeurs suivantes :

— Similarité : not similar, similar, very similar.

— Pertinence : not relevant, relevant, very relevant.

Comme j'ai choisi la RAG comme thème pour la dataset, j'ai préparé une vingtaine de questions et leurs réponses sur 13 documents PDF qui traitent cette thématique.

Conclusion

Ce chapitre a permis de décrire les étapes concrètes de la réalisation du projet. Il couvre le traitement des documents de la base de données, la gestion de la requête de l'utilisateur, le développement de l'interface de l'application, et enfin, une présentation de la méthode d'évaluation.

Chapitre 4

Résultats et Evaluation

Introduction

Dans ce chapitre, je présente les résultats de l'application de la méthode d'évaluation sur des systèmes RAG. Ces systèmes RAG diffèrent au niveau du modèle utilisé pour les embeddings. Les résultats obtenus sont ensuite analysés afin de conclure sur les performances de ces systèmes.

4.1 l'application de la methode d'évaluation

J'ai appliqué la méthode d'évaluation sur deux modèles d'embedding :
`sentence-transformers/all-MiniLM-L6-v2` (modèle 1) : produit des vecteurs avec 384 dimensions.
`allenai/scibert_scivocab_uncased` (modèle 2) : produit des vecteurs avec 768 dimensions.
Le temps nécessaire pour remplir la base de données vectorielles avec 11 691 phrases extraites de 13 documents PDF est le suivant :

Modèle	**Temps en secondes (s)**	**Temps en minutes (min)**
Modèle 1	485.03 s	8.08 min
Modèle 2	2697.60 s	44.96 min

TABLE 4.1 – Temps de remplissage de la base de données vectorielles pour chaque modèle.

4.2 Résultats de l'évaluation

Histogrammes de la similarité

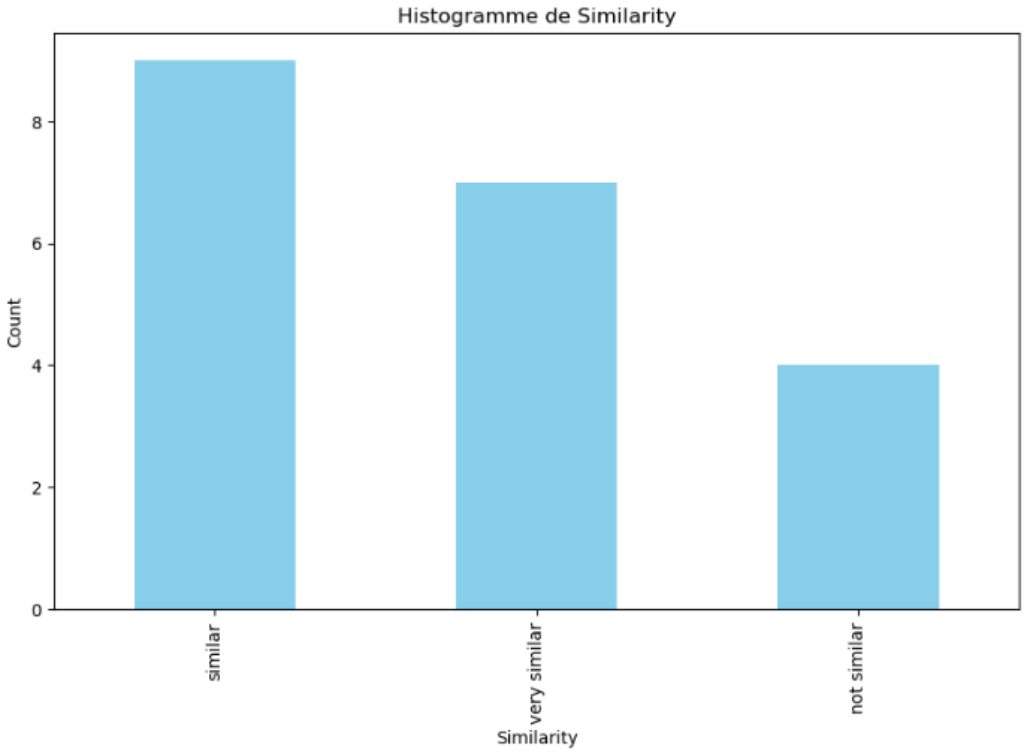


FIGURE 4.1 – Histogramme de la similarité du modèle 1

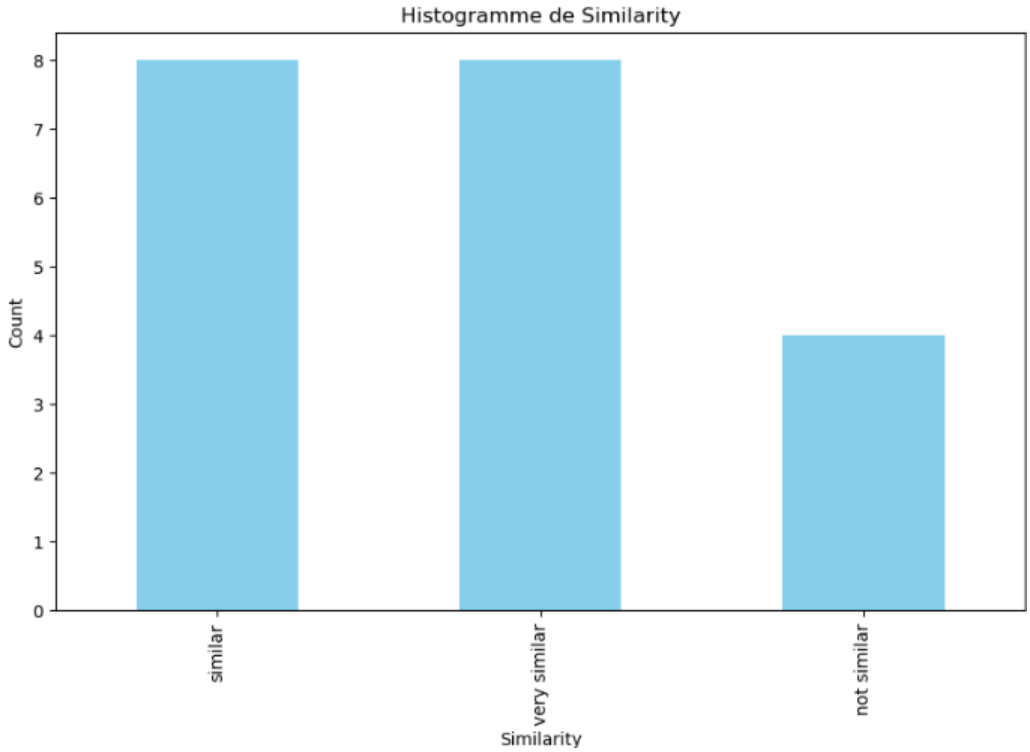


FIGURE 4.2 – Histogramme de la similarité du modèle 2

Le premier histogramme du modèle 1 montre que la majorité des réponses sont classées comme "similar" et "very similar," avec 9 et 7 occurrences respectivement. Un nombre plus faible de réponses, 4 réponses, est classé comme "not similar."

Le deuxième histogramme du modèle 2 montre une distribution similaire, avec une légère différence. On trouve 8 réponses classées comme "similar" et 8 autres classées comme "very similar" et 4 classées comme "not similar."

En comparant les deux modèles, on constate que leurs résultats sont globalement similaires. Toutefois, la différence entre les occurrences de "similaire" et "très similaire" entre les deux modèles n'est que d'une seule réponse. Cette légère variation peut être négligeable, mais peut s'explique aussi probablement par les performances supérieures du modèle 2, qui génère des vecteurs de 768 dimensions, lui permettant de mieux capturer le sens des phrases. Dans l'ensemble, les résultats sont très proches, ce qui indique une précision comparable pour les deux modèles.

Histogramme de la pertinence

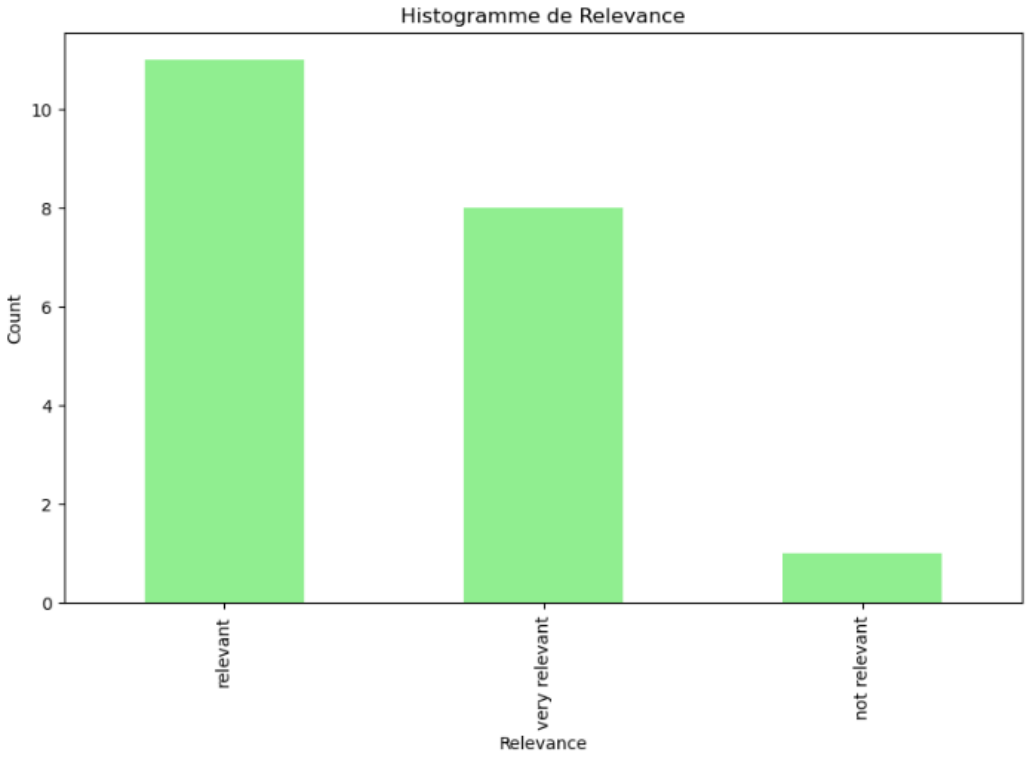


FIGURE 4.3 – Histogramme de la pertinence du modèle 1

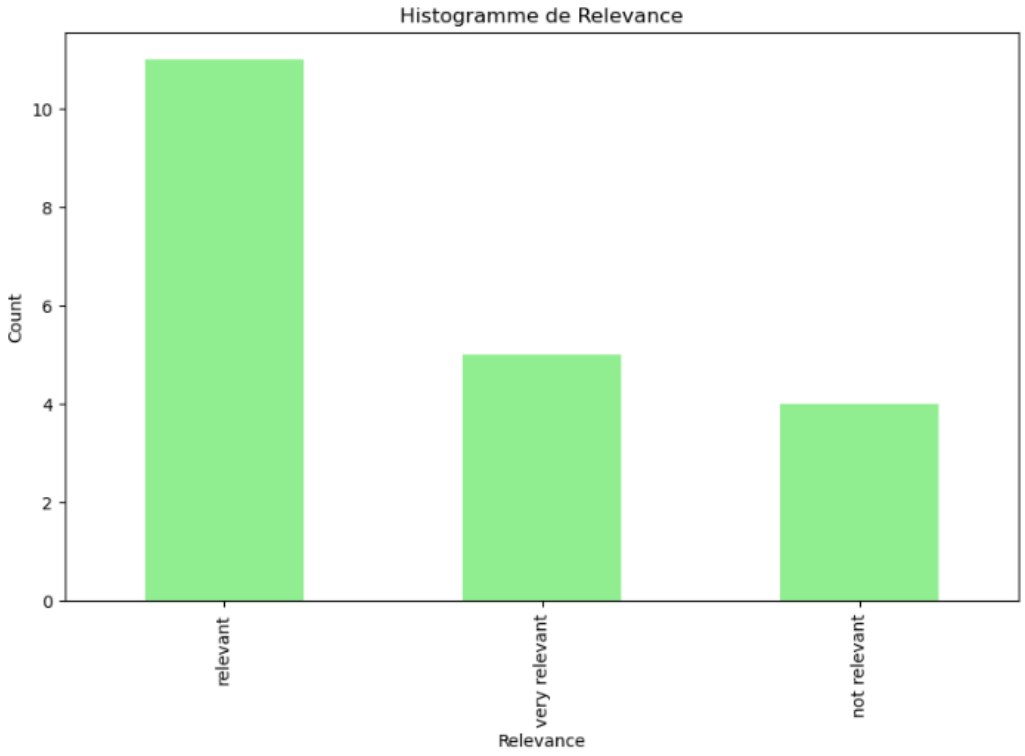


FIGURE 4.4 – Histogrammes de la pertinence du modèle 2

Le premier histogramme du modèle 1 montre que la majorité des réponses sont classées comme "relevant" avec 11 occurrences. Le reste des réponses est principalement réparti sur "very relevant" avec 8 occurrences puis "not relevant" avec une seule occurrences.

Le deuxième histogramme du modèle 2 montre une distribution similaire, avec une majorité de réponses également classées comme "relevant" (11 occurrences). Cependant, contrairement au premier modèle, le modèle 2 montre une répartition presque égale entre "very relevant" (5 occurrences) et "not relevant" (4 occurrences).

En comparant les deux modèles, on observe que le modèle 1 produit davantage de réponses dans la catégorie 'very relevant', tandis que le modèle 2 présente une proportion plus importante de réponses 'not relevant'. Cette évaluation de pertinence dépend du LLM utilisé pour le teste (Gemini-1.5-pro), qui détermine à quel point les données récupérées sont pertinentes en les comparant à la vérité. Il est possible que le LLM a commis des erreurs dans ses réponses. Cependant, affirmer que le modèle 1 est meilleur en termes de récupération reste discutable, car le modèle 2 est censé être plus performant, puisqu'il représente les phrases en 768 dimensions et a été fine-tuné sur des textes scientifiques.

Diagrammes en barre empilé

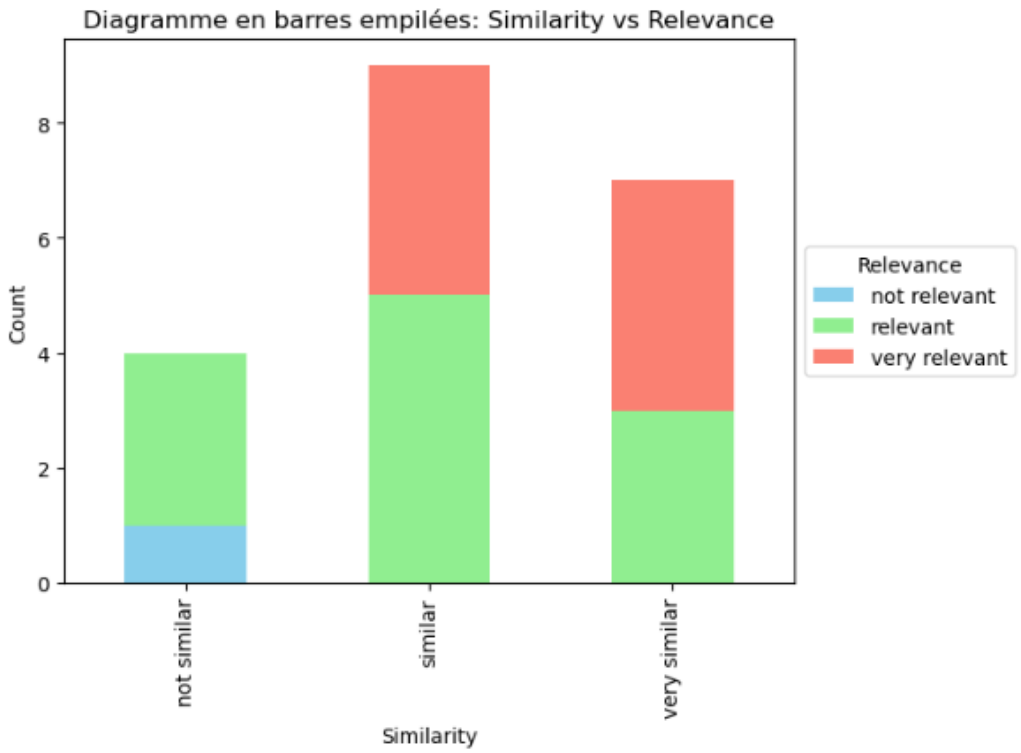


FIGURE 4.5 – Diagramme en barre empilé du modèle 1

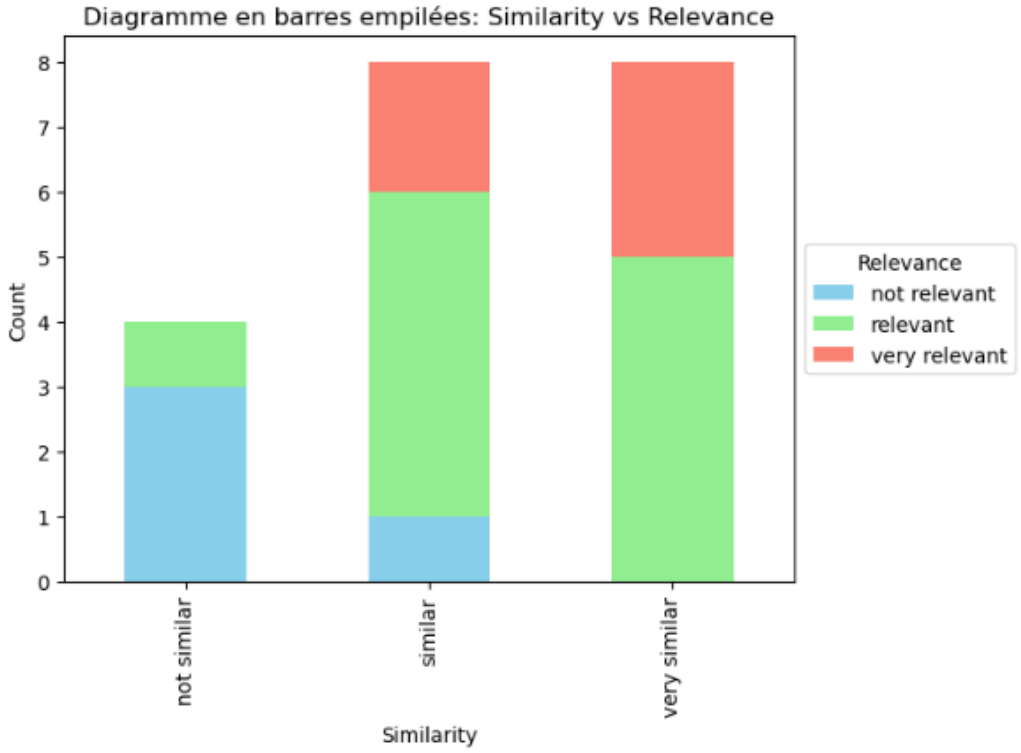


FIGURE 4.6 – Diagramme en barre empilé du modèle 2

Dans le premier diagramme du modèle 1, les barres montrent la répartition des catégories de pertinence ('not relevant', 'relevant', 'very relevant') en fonction de la similarité des réponses

('not similar', 'similar', 'very similar'). On observe que, pour la catégorie 'very similar', les occurrences de 'very relevant' sont plus dominantes que celles de 'relevant'. À l'inverse, pour la catégorie 'similar', les occurrences de 'relevant' sont plus fréquentes. Ce qui est intéressant, c'est l'absence de 'not relevant' dans ces deux catégories.

Dans le deuxième diagramme du modèle 2, la répartition est presque similaire, avec quelques différences notables. Dans les catégories 'similar' et 'very similar', la répartition des réponses 'relevant' est dominante, bien qu'une occurrence de 'not relevant' est présente dans la catégorie 'similar'.

Ces deux diagrammes montrent une certaine cohérence dans les réponses et les informations récupérées. L'absence de 'not relevant' dans la catégorie 'very similar' est un indicateur positif du bon fonctionnement des deux systèmes de RAG. La présence de 'not relevant' uniquement dans la catégorie 'not similar', à l'exception d'une occurrence dans 'similar' pour le modèle 2, pourrait être un signe de la faible tendance à fournir des réponses éloignées de la vérité. Dans l'ensemble, les deux systèmes de RAG ont montré des résultats positifs sur la vingtaine de questions de l'évaluation.

Diagrammes en bulles

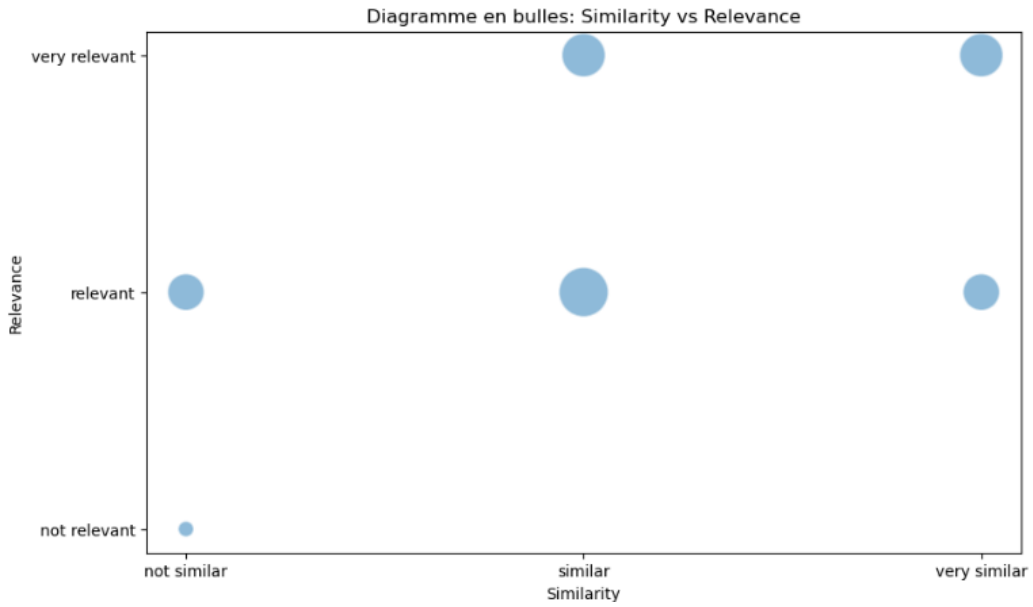


FIGURE 4.7 – Diagramme en bulles du modèle 1

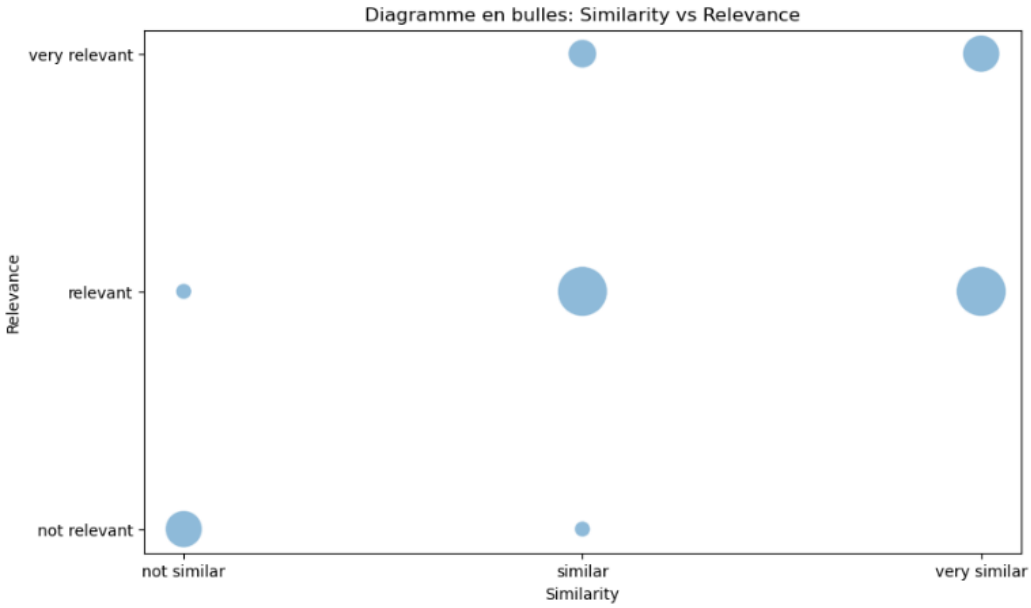


FIGURE 4.8 – Diagramme en bulles du modèle 2

Ces deux figures montrent la tendance des réponses des deux modèles. On observe que les deux ont tendance à fournir des réponses qui sont "very similar" et "similar" par rapport à la vérité, et que les informations récupérées pour ces réponses ont une plus grande probabilité d'être "relevant" ou "very relevant". Ces résultats sont des indicateurs de la bonne performance et la précision des deux systèmes de RAG.

Conclusion

Dans ce dernier chapitre, je présente les résultats de l'évaluation de deux systèmes RAG. Cette évaluation a montré l'efficacité des systèmes RAG dans l'amélioration des réponses des LLMs, car les deux systèmes évalués ont donné des résultats positifs, suggérant une bonne performance et précision.

Conclusion Générale

L'objectif de ce projet était de construire un système qui implémente le concept de la génération augmentée par la récupération pour les articles scientifiques. Le but est de faciliter, pour les étudiants et les professionnels, l'accès à une immense quantité de données textuelles scientifiques. Cela permet de consulter plusieurs documents scientifiques dans un délai raisonnable, sans consommer beaucoup du temps.

La réalisation de ce projet s'est déroulée en plusieurs phases. Initialement, je me suis concentré sur la recherche afin de bien comprendre le projet. Ensuite, il y a eu la phase d'analyse et de conception, une étape cruciale pour la mise en œuvre du système. Par la suite, la phase de réalisation qui consiste à développer ce système. Enfin, la phase d'évaluation, dans laquelle j'ai discuté les performances de deux modèles, à la fois par rapport à eux-mêmes et en les comparant entre eux.

Les résultats de l'évaluation ont été positifs. Les deux modèles ont montré de bonnes performances et une précision satisfaisante dans leurs réponses. La plupart des réponses obtenues sont considérées comme similaires et très similaires à la vérité, et la majorité des informations récupérées sont pertinentes et très pertinentes par rapport à la vérité.

Ce travail réalisé met en lumière l'intérêt des différentes avancées dans le domaine du NLP. Chaque technologie, chaque concept peut contribuer à créer des systèmes et des pipelines qui simplifient le travail avec les données. Ces dernières sont cruciales dans le monde d'aujourd'hui pour l'extraction d'informations, d'où le besoin croissant de techniques et de méthodes d'analyse capables de faciliter et d'améliorer le processus d'exploitation de ces données.

Bibliographie

[1] 24-07-2024 :
NLP (Traitement Automatique des Langues) : https://fr.wikipedia.org/wiki/Traitement_automatique_des_langues
Page Wikipédia sur le traitement automatique des langues.

[2] 24-07-2024 :
LLM (Grand Modèle de Langage) : https://fr.wikipedia.org/wiki/Grand_mod%C3%A8le_de_langage
Page Wikipédia sur les grands modèles de langage.

[3] 24-07-2024 :
Domaines d'application des LLMs : https://www.elastic.co/fr/what-is/large-language-models
domaines d'application des grands modèles de langage.

[4] 26-07-2024 :
Transformateur (Définition) : https://klu.ai/glossary/transformer
le concept de transformateur.

[5] 26-07-2024 :
Transformateur : https://www.nvidia.com/en-us/glossary/large-language-models/
Explication par NVIDIA des transformateurs.

[6] 27-07-2024 :
Hallucinations des LLM : https://fr.wikipedia.org/wiki/Grand_mod%C3%A8le_de_langage
Page Wikipédia sur les hallucinations des grands modèles de langage.

[7] 27-07-2024 :
RAG : https://www.oracle.com/fr/artificial-intelligence/generative-ai/retrieval-augmented-generationrag/#:~:text=La%20RAG%20est%20une%20technique,de%20donn%C3%A9es%20suppl%C3%A9mentaires%20sans%20r%C3%A9entra%C3%AEnement.
Page Oracle expliquant la technique de la RAG.

[8] 30-07-2024 :
Le flux de travail de la RAG : https://towardsdatascience.com/retrieval-augmented-generation-rag-from-theory-to-langchain-implementati
Explication et illustration du flux de travail RAG sur Towards Data Science.

[9] 30-07-2024 :
Types de RAG : https://blog.jayanthk.in/types-of-rag-an-overview-0e2b3ed71b8
Blog décrivant les différents types de RAG.

[10] 30-07-2024 :
Définition Embedding : https://www.ibm.com/topics/embedding
Explication des embeddings par IBM.

[11] 30-07-2024 :
types de RAG : https://medium.com/@drjulija/what-are-naive-rag-advanced-rag-
Présentation des types de RAG sur Medium.

[12] 17-08-2024 :
Python : https://en.wikipedia.org/wiki/Python_(programming_language)#Uses
page Wikipedia pour Python.

[13] 17-08-2024 :
Langchain : https://mistral.ai/
page de Wikipedia pour Langchain librairie.

[14] 17-08-2024 :
Tutoriel PyQt6 : https://www.pythonguis.com/pyqt6-tutorial/
Un guide pour apprendre PyQt6.

[15] 17-08-2024 :
RAGAS (Outil d'évaluation pour RAG) : https://docs.ragas.io/en/stable/
Documentation officielle de RAGAS.