UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

Faculté des Sciences - Fès

Département d'Informatique

UNIVERSITÉ SORBONNE PARIS NORD

Partenariat International

RAPPORT DE PROJET

Assistant Intelligent RAG Multimodal

Réalisé par :

Lakehal Salma
Bouizdouzene Bilal

Formation :

Master Web Intelligence et Data Science

Année Universitaire 2025-2026

10 août 2025

Table des matières

CHAPITRE 1

Introduction

1.1 Contexte et objectifs du projet

Avec l'évolution rapide des technologies d'intelligence artificielle, les systèmes de dialogue automatisés ont gagné en importance, notamment dans les applications d'assistance virtuelle, support client et traitement intelligent de l'information. Ce projet vise à développer un chatbot intelligent basé sur la méthode *Retrieval-Augmented Generation* (RAG), capable de traiter des questions posées en texte, audio ou fichiers, et d'exploiter une base documentaire riche pour fournir des réponses précises et contextualisées.

L'objectif principal est de concevoir un système multimodal qui intègre la reconnaissance vocale (via Whisper), la recherche documentaire sémantique (via FAISS et embeddings HuggingFace), et la génération de texte avancée (via le modèle Gemini 2.5 Pro), tout en assurant la gestion d'un historique conversationnel pour une meilleure continuité.

1.2 Présentation générale du système

Le système développé fonctionne selon un pipeline structuré : les données entrantes (texte, audio ou fichiers) sont d'abord prétraitées et transformées en format exploitable. Les contenus textuels extraits sont indexés dans une base vectorielle FAISS après avoir été segmentés sémantiquement et vectorisés grâce à un modèle de type `sentence-transformers`.

Lorsqu'une question est posée, le système effectue une recherche de documents pertinents dans l'index FAISS, applique un reranking pour prioriser les meilleurs extraits, puis génère une réponse enrichie en tenant compte de l'historique récent de la conversation, garantissant ainsi une interaction fluide et cohérente.

CHAPITRE 2

Architecture du système

2.1 Description générale

L'architecture du chatbot RAG multimodal est conçue pour intégrer plusieurs composants clés permettant le traitement efficace de différentes formes d'entrée utilisateur, ainsi que la génération de réponses précises et contextualisées.

Le système est organisé autour d'un pipeline principal où les données entrantes (texte, audio, fichiers) sont prétraitées, indexées, et utilisées pour la recherche d'informations augmentée par génération (RAG).

2.2 Architecture du système

Cette section décrit en détail l'architecture du système, depuis la réception des entrées utilisateur jusqu'à la génération de la réponse finale.

2.2.1 Entrées utilisateur (texte, audio, fichiers)

Le système accepte plusieurs types d'entrées utilisateur :

— **Texte** : Questions saisies directement par l'utilisateur via l'interface.

— **Audio** : Enregistrements vocaux transcrits en texte grâce au modèle Whisper.

— **Fichiers** : Documents (PDF, DOCX, images) uploadés et dont le contenu textuel est extrait.

Ces différentes modalités permettent une interaction riche et flexible avec l'assistant.

2.2.2 Prétraitement des données

Avant toute utilisation, les données brutes subissent plusieurs étapes de prétraitement :

— **Extraction de texte** : Utilisation de bibliothèques spécialisées pour extraire le contenu textuel des fichiers uploadés (PyPDF2, python-docx, Tesseract OCR, Whisper).

— **Nettoyage et normalisation** : Suppression des caractères inutiles, correction de l'encodage, découpage en passages cohérents (chunking sémantique) pour faciliter la recherche.
— **Création de métadonnées** : Attribution d'identifiants uniques aux documents, enregistrement des titres et sources pour un suivi précis.

2.2.3 Indexation et base documentaire (FAISS)

Les documents extraits sont vectorisés à l'aide d'un modèle d'embeddings pré-entraîné (sentence-transformers/all-MiniLM-L6-v2) puis stockés dans une base de données vectorielle FAISS pour des recherches efficaces.

— **Chargement et sauvegarde** : L'index FAISS est chargé au démarrage de l'application et sauvegardé après chaque modification.
— **Gestion dynamique** : Possibilité d'ajouter ou de supprimer des documents à la volée, avec contrôle des doublons grâce aux identifiants uniques.

2.2.4 Recherche et reranking documentaire

Lorsqu'une question est posée, le système interroge la base FAISS pour retrouver les documents les plus pertinents selon la similarité vectorielle. Ces documents sont ensuite réordonnés (reranking) avec un cross-encoder pour améliorer la pertinence finale des résultats.

2.2.5 Gestion de l'historique conversationnel

L'historique des échanges est stocké en base de données, permettant de conserver le contexte sur plusieurs tours de dialogue. Cela garantit la cohérence des réponses et évite la perte de mémoire liée à un redémarrage ou une nouvelle session.

2.2.6 Génération de réponse avec Gemini

Le modèle Gemini 2.5 Pro est utilisé pour générer des réponses détaillées et contextualisées, en combinant les informations issues des documents retrouvés et l'historique conversationnel. Un résumé simplifié est également produit pour faciliter la compréhension.

2.3 Flux global du traitement

Le processus complet suit la séquence suivante :

1. Réception de la question (texte, audio transcrit, ou question + fichier).
2. Extraction et prétraitement du contenu des fichiers si nécessaire.

3. Ajout du contenu indexé dans FAISS avec métadonnées.
4. Recherche des documents pertinents dans FAISS et reranking.
5. Récupération de l'historique conversationnel depuis la base de données.
6. Construction du prompt combinant la question, les documents et l'historique.
7. Génération de la réponse par Gemini.
8. Enregistrement de la nouvelle interaction dans l'historique.

Cette architecture modulaire assure une grande flexibilité, évolutivité et robustesse du système.

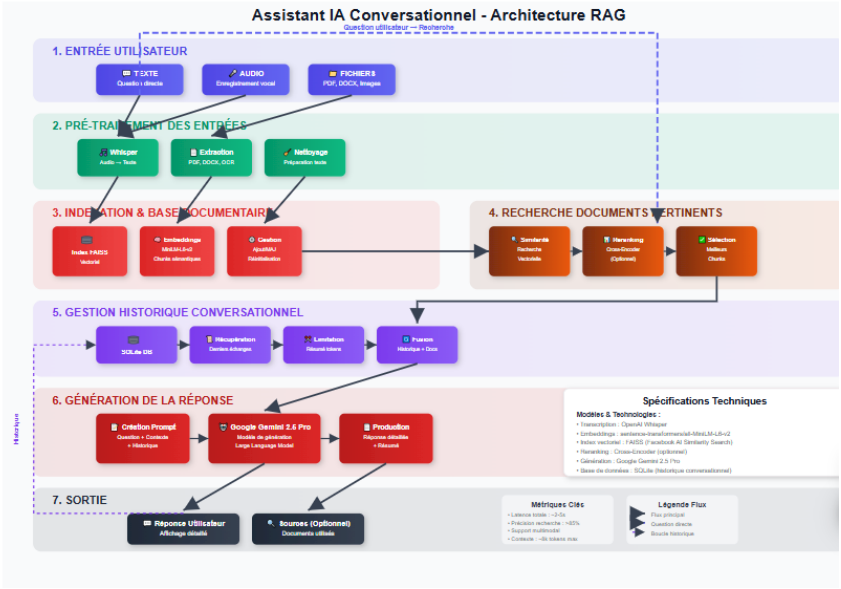


FIGURE 2.1 – Schéma de l'architecture du système montrant le flux de données du questionnement utilisateur jusqu'à la réponse générée.

2.4 Gestion de l'historique conversationnel

Le système conserve un historique structuré des échanges utilisateur-assistant, stocké dans une base de données. Cette mémoire contextuelle permet de maintenir la cohérence des dialogues, même après redémarrage du système, et d'améliorer la pertinence des réponses en tenant compte du fil de la discussion.

CHAPITRE 3

Gestion des données et entraînement

Ce chapitre décrit le processus complet de gestion des données, depuis leur acquisition jusqu'à la construction et la mise à jour de l'index FAISS utilisé pour la recherche documentaire.

3.1 Sources des données (dataset)

Les données proviennent de diverses sources, incluant :

— Documents utilisateurs : fichiers PDF, DOCX, images ou autres documents uploadés par les utilisateurs.

— Bases documentaires internes : collections de documents textuels pertinents pour le domaine d'application.

— Données externes : sources publiques ou privées, pouvant être intégrées selon les besoins.

Ces données sont la matière première indispensable à la constitution d'une base documentaire riche et utile pour l'assistant.

3.2 Extraction et segmentation sémantique (chunking)

Pour optimiser la recherche, les documents complets sont segmentés en passages sémantiques cohérents appelés *chunks*. Ce découpage est réalisé en tenant compte :

— Du nombre maximal de tokens par chunk (environ 500 tokens).

— Du chevauchement entre chunks (overlap de 100 tokens) pour maintenir le contexte.

— De la cohérence sémantique afin que chaque chunk contienne une idée ou un thème homogène.

Cette segmentation améliore la granularité des recherches et la pertinence des résultats.

3.3 Création des embeddings (modèle sentence-transformers)

Chaque chunk est converti en un vecteur numérique de faible dimension à l'aide d'un modèle d'embeddings pré-entraîné, ici `sentence-transformers/all-MiniLM-L6-v2`. Ce modèle encode la signification sémantique des textes, permettant de comparer efficacement la similarité entre question et documents.

3.4 Construction et entraînement de l'index FAISS

Les vecteurs d'embeddings sont indexés dans la base vectorielle FAISS, qui permet une recherche rapide et scalable dans des bases de données volumineuses.

— **Construction initiale** : Création de l'index à partir des documents existants.
— **Entraînement** : FAISS propose des algorithmes d'indexation (ex. IVF, HNSW) nécessitant parfois une phase d'entraînement pour optimiser les structures de recherche.
— **Sauvegarde** : L'index entraîné est sauvegardé localement pour réutilisation.

3.5 Mise à jour et gestion dynamique de l'index

Le système supporte l'ajout ou la suppression dynamique de documents, avec des mécanismes pour :

— Vérifier la présence d'un document via un identifiant unique afin d'éviter les doublons.
— Ajouter les nouveaux chunks vectorisés à l'index existant.
— Réentraîner ou réorganiser l'index FAISS si nécessaire pour maintenir une performance optimale.
— Sauvegarder régulièrement l'état de l'index.

Cette gestion dynamique assure que la base documentaire est toujours à jour et cohérente avec les contenus disponibles pour la recherche.

CHAPITRE 4

Technologies et outils utilisés

Ce chapitre présente les principaux langages, frameworks, modèles d'IA et outils qui composent l'architecture du projet.

4.1 Langages et frameworks

Le projet est principalement développé avec les technologies suivantes :

— **Python** : langage principal pour le backend, gestion des modèles IA, traitement des données, et orchestration du système.

— **Flask** : micro-framework web léger utilisé pour construire l'API backend et gérer les requêtes utilisateur.

— **JavaScript / HTML / CSS** : technologies frontend pour l'interface utilisateur, incluant la gestion des interactions, l'upload de fichiers, et l'enregistrement audio.

— **SQLAlchemy** : ORM utilisé pour la gestion de la base de données SQLite, facilitant la persistance de l'historique conversationnel.

4.2 Modèles d'IA (Whisper, Gemini, sentence-transformers)

Le projet intègre plusieurs modèles d'intelligence artificielle pour différentes tâches :

— **Whisper** (OpenAI) : modèle de reconnaissance vocale automatique (ASR) utilisé pour la transcription des entrées audio en texte.

— **Gemini 2.5 Pro** (Google Generative AI) : modèle de génération de langage naturel employé pour générer les réponses aux questions en combinant les informations issues des documents et de l'historique.

— **Sentence-transformers** : modèles spécialisés dans la création d'embeddings sémantiques pour la vectorisation des documents et des requêtes, facilitant la recherche par similarité.

4.3 Vectorisation et recherche sémantique (FAISS)

— **FAISS** (Facebook AI Similarity Search) : bibliothèque open-source optimisée pour la recherche rapide de vecteurs similaires dans de grandes bases de données. Elle permet :
  — L'indexation efficace des embeddings.
  — La recherche par similarité sémantique.
  — La gestion dynamique de l'index avec ajout ou suppression de documents.

4.4 Base de données et gestion de l'historique

Pour assurer la continuité des conversations et le suivi des interactions utilisateur, le projet utilise :

— **SQLite** : base de données légère embarquée pour stocker l'historique des échanges.
— **SQLAlchemy** : outil ORM facilitant l'interaction avec SQLite, permettant la récupération et l'insertion des messages dans la table historique.
— **Gestion de sessions et threads** : chaque conversation est associée à un identifiant de session et de thread pour organiser l'historique de manière cohérente et multi-utilisateurs.

CHAPITRE 5

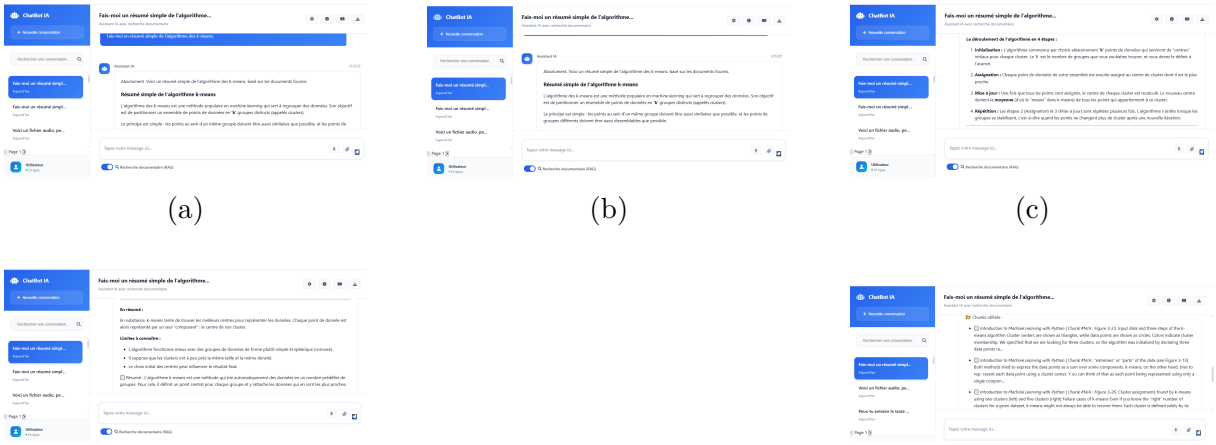
Résultats et démonstration

Ce chapitre illustre la performance et les fonctionnalités principales du système à travers des exemples concrets et des visualisations.

5.1 Exemples de questions et réponses

Pour valider la qualité des réponses générées, plusieurs questions variées ont été posées au système, portant sur des documents préalablement indexés ou des questions directes. Voici quelques exemples :

5.1.1 Questions générales

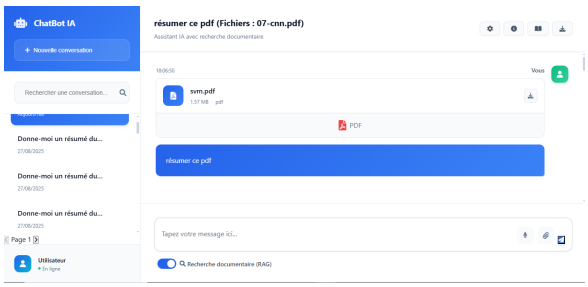


(a) (b) (c)

(d) (e)

FIGURE 5.1 – chat normal avec RAG.

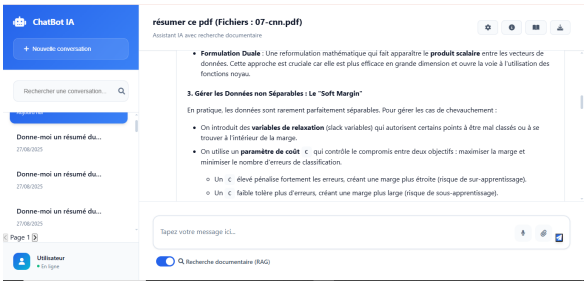
5.1.2 Questions multimodales avec PDF (upload de fichiers)



(a)

(b)

(c)



(d)

(e)

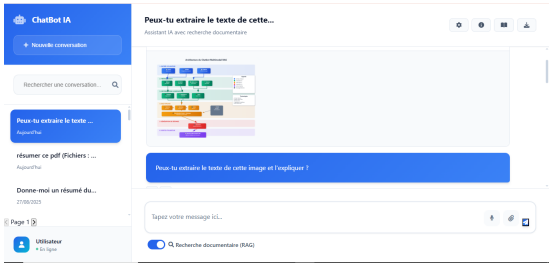
(f)

(g)

(h)

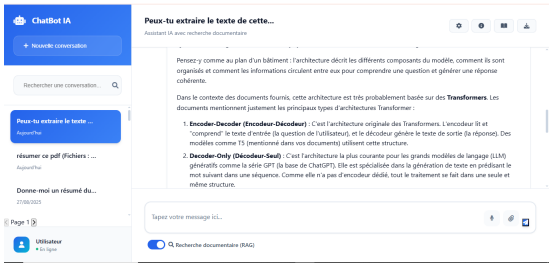
FIGURE 5.2 – Question avec PDF

5.1.3 Questions multimodales avec image (OCR)



(a)

(b)

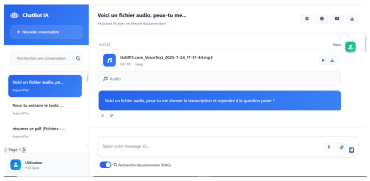


(c)

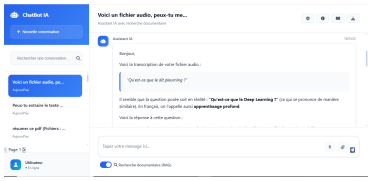
(d)

FIGURE 5.3 – Question avec image

5.1.4 Questions multimodales avec audio (Whisper)



(a)



(b)

(c)

(d)

(e)

FIGURE 5.4 – Question avec audio

Ces exemples démontrent la capacité du système à comprendre et exploiter différentes sources d'information pour fournir des réponses pertinentes et contextualisées.

5.1.5 Visualisation du contexte documentaire utilisé

Le système met en évidence les extraits documentaires ayant servi à la génération des réponses. Cette transparence est assurée par :

— L'affichage des titres des documents et des indices des chunks (segments) utilisés.
— Un aperçu textuel (quelques centaines de caractères) de chaque chunk exploité dans la réponse.
— La possibilité pour l'utilisateur de vérifier l'origine des informations fournies, augmentant la confiance dans la réponse.

Cette fonctionnalité est cruciale pour les applications professionnelles où la traçabilité des sources est essentielle.

5.1.6 Gestion de l'historique et continuité de la conversation

Le projet intègre un module de gestion d'historique conversationnel performant qui permet :

— La sauvegarde des échanges entre l'utilisateur et l'assistant dans une base de données SQLite, garantissant la persistance même en cas de redémarrage du système.
— La récupération des derniers tours de dialogue, permettant au modèle de conserver le contexte et d'offrir des réponses cohérentes dans la continuité des échanges.
— La prise en compte de l'historique dans la formulation des prompts, améliorant la qualité des réponses et évitant la perte de contexte.
— La gestion multi-utilisateurs et multi-sessions, avec organisation par identifiants de session et de thread.

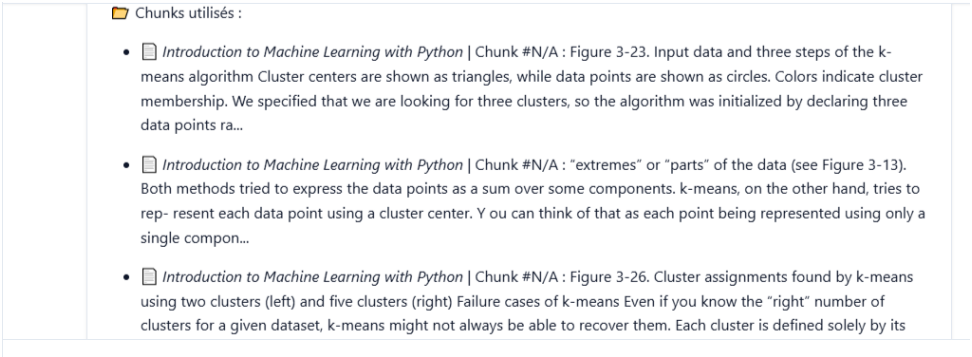


FIGURE 5.5 – Exemple des chunks utilisées.

Grâce à ce système, l'expérience utilisateur est fluide, et la conversation apparaît naturelle et contextualisée, comparable à un échange humain.

CHAPITRE 6

Limites et perspectives d'amélioration

Ce chapitre analyse les principales limites rencontrées dans le projet ainsi que les pistes envisagées pour son amélioration et son évolution.

6.1 Limites actuelles du système

Malgré ses nombreuses fonctionnalités, le système présente plusieurs limitations importantes :

— **Capacité limitée de mémoire contextuelle :** La gestion de l'historique conversationnel est restreinte à un nombre fixe de derniers échanges, ce qui peut entraîner une perte progressive de contexte sur de longues conversations.

— **Dépendance aux performances de l'index FAISS :** La qualité et la rapidité des recherches dépendent fortement de la taille et de la qualité de l'index. Pour de très grands volumes de documents, l'index peut devenir lourd à manipuler sans optimisation supplémentaire.

— **Qualité variable des réponses :** La génération par Gemini peut parfois produire des réponses trop générales ou hors sujet, notamment lorsque les documents indexés ne couvrent pas précisément la question posée.

— **Traitement des fichiers limité :** Le système prend en charge plusieurs formats (PDF, DOCX, images, audio), mais certains types ou fichiers complexes peuvent poser problème lors de l'extraction de texte ou la segmentation.

— **Entraînement et mise à jour de l'index non automatisés :** L'index FAISS est mis à jour manuellement via l'ajout de documents. Un pipeline d'automatisation ou d'apprentissage continu serait bénéfique.

6.2 Améliorations futures possibles

Plusieurs axes d'amélioration sont envisageables pour rendre le système plus robuste, performant et polyvalent :

— **Extension de la mémoire conversationnelle :** Intégrer des techniques avancées de gestion du contexte (ex : mémoire longue, résumé dynamique) pour conserver davantage d'informations pertinentes au fil des échanges.

— **Optimisation et scalabilité de l'index FAISS :** Implémenter des stratégies de sharding, compression ou utilisation de bases vectorielles distribuées pour gérer efficacement de très grands corpus documentaires.

— **Amélioration du reranking :** Intégrer des modèles cross-encoder plus performants ou des méthodes hybrides pour améliorer la pertinence des documents sélectionnés.

— **Automatisation du pipeline d'indexation :** Développer un système d'ingestion automatique des documents avec extraction, chunking, embedding et mise à jour de l'index en continu.

— **Support étendu de formats et multimodalité :** Améliorer l'extraction de données depuis différents formats complexes et intégrer des modèles multimodaux plus avancés (ex : Vision-Language Models).

— **Interface utilisateur et expérience :** Ajouter des fonctionnalités interactives, un tableau de bord de suivi des conversations, et une meilleure gestion des erreurs pour une utilisation professionnelle.

Ces améliorations permettraient au système de mieux répondre aux exigences d'applications industrielles et de recherche avancée, tout en offrant une expérience utilisateur enrichie.

CHAPITRE 7

Conclusion

Ce projet a permis de développer un assistant intelligent reposant sur une architecture avancée de Retrieval-Augmented Generation (RAG), combinant indexation documentaire via FAISS, gestion dynamique des documents, et génération de réponses à l'aide du modèle Gemini. La prise en charge multimodale des entrées (texte, audio, fichiers) ainsi que la gestion d'un historique conversationnel ont été des points clés pour garantir une expérience utilisateur fluide et contextuelle.

L'approche modulaire adoptée facilite la maintenance et l'évolution du système, notamment par l'intégration de techniques de reranking et de segmentation sémantique pour optimiser la pertinence des résultats. Néanmoins, certaines limites liées à la mémoire contextuelle, la gestion de l'index et la qualité des réponses démontrent les opportunités d'amélioration, notamment vers plus d'automatisation et de scalabilité.

Ce travail ouvre la voie à des applications variées dans le domaine de l'assistance virtuelle, la recherche documentaire intelligente, et le traitement multimodal des données. Les futures évolutions viseront à renforcer la robustesse, la précision et la richesse fonctionnelle du système, en tirant parti des avancées récentes en intelligence artificielle et traitement du langage naturel.