

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ÉCOLE NATIONALE POLYTECHNIQUE



Département Génie Industriel
Entreprise : BNP Paribas El Djazaïr

Mémoire de projet de fin d'études
Pour l'obtention du diplôme d'ingénieur d'état en Génie Industriel Option : Data
Science & Intelligence Artificielle

Thème :
Conception d'un assistant intelligent pour la valorisation de l'information interne
-BNP Paribas El Djazaïr

MERADJI M.Idir et ZENNADI Zina
Sous la direction de Dr. ARKI Oussama, ENP & Mr. OUTMANIOU Omar et Mme.Nihad
Senhadji

Présenté et soutenu publiquement le (23/06/2025)

Composition du jury :

Président : Dr. Hakim Fourar Laidi	MCA ENP
Examinatrice : Dr. Samia BELDJOUDI	MCA ENP
Promoteur : Dr. Oussama ARKI	MCA ENP

ENP 2025

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ÉCOLE NATIONALE POLYTECHNIQUE



Département Génie Industriel
Entreprise : BNP Paribas El Djazaïr

Mémoire de projet de fin d'études
Pour l'obtention du diplôme d'ingénieur d'état en Génie Industriel Option : Data
Science & Intelligence Artificielle

Thème :
Conception d'un assistant intelligent pour la valorisation de l'information interne
-BNP Paribas El Djazaïr

MERADJI M.Idir et ZENNADI Zina
Sous la direction de Dr. ARKI Oussama, ENP & Mr. OUTMANIOU Omar et Mme.Nihad
Senhadji

Présenté et soutenu publiquement le (23/06/2025)

Composition du jury :

Président : Dr. Hakim Fourar Laidi	MCA ENP
Examinatrice : Dr. Samia BELDJOUDI	MCA ENP
Promoteur : Dr. Oussama ARKI	MCA ENP

ENP 2025

Remerciements

Qu'il nous soit permis de remercier et d'exprimer notre profonde gratitude en premier lieu à Dieu le Tout-Puissant de nous avoir accordé Sa bénédiction, de nous avoir donné la force, la volonté et la patience pour mener à bien ce travail.

Nous tenons à adresser nos remerciements les plus sincères à M. Arki Oussama, notre encadrant académique, pour son accompagnement, ses précieux conseils et son soutien tout au long de notre scolarité et projet de fin d'étude. Son expertise et sa disponibilité ont été d'une grande aide pour nous.

Nous souhaitons également exprimer notre profonde reconnaissance à M. Otmaniou Omar, notre encadrant au sein de l'entreprise BNP Paribas El Djazaïr, pour sa guidance, ses conseils pertinents et son appui constant. Son professionnalisme et son engagement ont été des éléments moteurs dans la réalisation de ce travail.

Un merci particulier à toutes les personnes que nous avons rencontrées chez BNP Paribas El Djazaïr, qui nous ont chaleureusement accueillis et ont contribué à créer un environnement favorable à l'accomplissement de ce projet. Nous tenons notamment à remercier Nihed, dont l'aide et les contributions ont été très précieuses. Mais aussi Yasmina, Kenza, Meziane et Khaled pour leur accueil très chaleureux.

Nous souhaitons également exprimer notre gratitude anticipée aux membres du jury qui auront la charge d'évaluer ce travail. Nous les remercions sincèrement pour le temps et l'expertise qu'ils consacrent à la lecture et à l'appréciation de ce projet. Leur regard critique et leurs remarques constructives seront, nous en sommes sûrs, d'une grande valeur pour l'enrichissement de notre travail.

Enfin, nous tenons à exprimer notre reconnaissance envers l'ensemble du corps professoral de l'École Nationale Polytechnique, département de Génie Industriel, pour la qualité de l'enseignement et l'encadrement qu'ils nous ont offerts tout au long de notre parcours. Leur engagement et leur professionnalisme ont largement contribué à notre formation académique et personnelle.

Dédicace

Je dédie ce modeste travail à toutes les personnes qui occupent une place essentielle dans ma vie et dans mon cœur.

À ma mère,

Maman, tu es mon premier refuge, mon soutien le plus constant et la lumière qui éclaire mon chemin. Ta foi inébranlable en moi, même dans les moments où moi-même je doutais, m'a porté et m'a donné la force d'avancer. Ton amour, ton dévouement et tes prières silencieuses m'ont accompagné à chaque étape de ce parcours. Merci d'avoir cru en moi sans jamais hésiter, merci d'être cette source inépuisable de tendresse et de courage.

À mon père,

Papa, ton regard fier et confiant a toujours été une source de motivation pour moi. Tu as cru en mes capacités bien avant que je puisse les prouver. Grâce à ton soutien discret mais puissant, à tes conseils empreints de sagesse et à ta confiance inconditionnelle, j'ai pu affronter les défis avec assurance. Merci pour ta patience, ton exigence bienveillante et pour avoir toujours été là, tel un roc, à mes côtés.

À mes frères Abdelrahmene, Nezar et Salah,

Vous êtes bien plus que des frères, vous êtes mes alliés de toujours, mes repères dans les moments d'incertitude, ma force silencieuse. Chacun de vous, à sa manière, m'a donné du courage, de la joie et un sentiment de sécurité inestimable. Vos mots, vos gestes, votre présence ont souvent été ce dont j'avais besoin pour continuer à avancer, même sans que vous le sachiez. Merci d'être là, simplement, mais puissamment. Je vous aime profondément.

À la mémoire de mes deux grands-mères, dont la douceur et les prières m'ont longtemps accompagné. Qu'Allah leur accorde Son pardon et les comble de Sa miséricorde.

À mes camarades de classe, et plus particulièrement à mes amis les plus proches, avec qui j'ai partagé les plus beaux souvenirs de ces dernières années. Merci pour votre amitié sincère, votre soutien et toutes ces années pleines de rires, de défis et de moments gravés dans le cœur.

Et le meilleur pour la fin **Idir**, mon binôme, mon meilleur ami, que j'aime énormément. Merci pour ta présence constante, ton énergie, ton soutien, et cette belle complicité qui a rendu ce parcours plus riche, plus fort et plus humain. Je t'adore et je t'estime profondément.

Zina

Dédicaces

Je dédie ce mémoire à toutes les personnes qui occupent une place essentielle dans ma vie et dans mon cœur.

À ma mère Ihou,

Celle dont la confiance inébranlable en moi demeure ma plus grande source de motivation et d'optimisme. Ta foi en mes capacités me pousse toujours à donner le meilleur de moi-même.

Dans les moments où le stress prenait le dessus, tu as su, par ta douceur et ta présence, m'apaiser et me redonner courage. Je te dois énormément.

À mon père Joe,

Celui qui, même dans les périodes les plus difficiles, a su trouver les mots pour me remotiver et me redonner confiance. Merci pour tes encouragements constants et ta présence rassurante, si précieuse à mes yeux.

À ma sœur Ja,

Merci pour ton soutien indéfectible, parfois teinté de sarcasme – ce qui, étonnamment, m'a toujours aidé à garder les pieds sur terre et à relativiser. Ta présence, même silencieuse, a compté plus que tu ne l'imagines.

À ma tante Mina,

Merci pour ton soutien précieux et ta bienveillance. Tes conseils avisés et ton optimisme communicatif m'ont permis de surmonter bien des obstacles et d'avancer avec sérénité.

À ma grand-mère Mamati,

Qui veille toujours à ce que je ne manque de rien. Je t'aime de tout mon cœur et te suis profondément reconnaissant pour tout ce que tu m'as apporté.

À mon autre tante Kissa et à mon grand-père dadi,

Merci d'être toujours là, présents avec amour et bienveillance. Votre soutien inconditionnel est un véritable trésor dans ma vie.

À mon oncle Amo Salah,

Même si les mots entre nous sont rares, ton soutien silencieux a toujours été présent. Je sais que tu es là, à fond derrière moi, et cette force discrète me pousse à aller plus loin chaque jour. Merci pour ta présence fidèle.

À mon oncle Chérif et ma tante Fazia,

Vous avez été des piliers inébranlables. Quand tout semblait bloqué, vous trouviez toujours un chemin.

Votre générosité, votre foi en moi, et votre soutien indéfectible ont été essentiels pour que je puisse avancer dans les meilleures conditions. Merci du fond du cœur.

À l'ensemble de ma famille,

Votre soutien collectif et vos encouragements constants m'ont porté et guidé tout au long de mon parcours.

À mes camarades de classe, aux étudiants du département Génie Industriel et à l'IEC,

Merci pour chaque instant partagé, chaque éclat de rire et chaque moment d'entraide. Ces trois années resteront, sans aucun doute, les plus belles de ma vie, et c'est à chacun d'entre vous que je le dois.

À toutes les rencontres incroyables nouées durant ces trois années,

Je pense particulièrement à Chakib, Amine, Ali, Marouane, Nazim, Yasmine, Ines, Amelia, Hocine, Lyna, Ilyas, Dassine, Hadil, Abdenmour, Adel, mais aussi Yasmina, Nihad, Kenza que j'ai eu la chance de connaître durant mon Stage à BNP et tant d'autres... J'ai sans doute oublié des noms, mais sachez que chacun a compté à sa manière.

À la famille de Daar Ejwaz : Doudou, Bouby, Guigui, Bakhi, Manelus, Serinus, BMO et Labadi :

Merci pour votre soutien, votre aide psychologique, tous ces moments inoubliables et ces souvenirs précieux. Votre présence et votre bonne humeur ont transformé mon passage à Polytech en une aventure humaine exceptionnelle.

Et la meilleure pour la fin **Zina,**

Ma meilleure amie, ma confidente, ma bestie, ma BBF, mon binôme dans ce projet et bien plus encore. Merci pour ta patience infinie, ton soutien sans faille, ta compréhension, ton humour et ton rire communicatif. Les souvenirs de nos fous rires et de nos galères resteront à jamais gravés. Ta présence a illuminé chaque étape de ce projet et de mon parcours universitaire. Tu es une source de joie et d'énergie positive, et je suis profondément reconnaissant d'avoir pu partager cette aventure à tes côtés.

À toutes et à tous, du fond du cœur, **merci.**

Idir

المخلص

تقدم هذه الرسالة تطوير مساعد ذكي داخلي لبنك بي إن بي باريبا الجزائر، يهدف إلى تسهيل الوصول إلى المعلومات التجارية والإجراءات الداخلية. تستند بنية النظام إلى إطار عمل متعدد الوكلاء متخصص، حيث يقوم كل وكيل بمهمة محددة (إعادة صياغة الاستعلامات، البحث، التوليف، التحقق من الامتثال، إلخ)، مما يتيح معالجة مرنة وقابلة للتفسير لاستفسارات المستخدمين. تعتمد التقنية الأساسية على طريقة الاسترجاع المدعوم بالتوليد (RAG) التي تجمع بين البحث المتجهي السريع وتوليد الردود التلقائي. تهدف هذه الحلول إلى تحسين الأداء التشغيلي، وإمكانية تتبع العمليات، وجودة الإجابات مع ضمان الامتثال للسياسات الداخلية. يتضمن النموذج الأولي المطور واجهة سهلة الاستخدام وخط معالجة ذكي، مما يبرز جدوى نموذج الوكلاء المتعددين للمساعدين التجاريين المعقدين.

الكلمات المفتاحية: الذكاء الاصطناعي، مساعد ذكي، نظام متعدد الوكلاء، التوليد المعزز بالاسترجاع (RAG)، أنظمة المعلومات الداخلية

Abstract

This thesis presents the development of an internal intelligent assistant for BNP Paribas El Djazaïr, designed to facilitate access to business information and internal procedures. The system architecture is based on a **specialized multi-agent framework**, where each agent performs a specific task (query reformulation, search, synthesis, compliance checking, etc.), enabling modular, flexible, and explainable handling of user queries. The core technology leverages **Retrieval-Augmented Generation (RAG)**, combining fast vector search with automatic response generation. The solution aims to enhance operational performance, traceability, and answer quality while ensuring compliance with internal policies. The developed prototype features an ergonomic interface and an intelligent processing pipeline, highlighting the suitability of the multi-agent model for complex business assistants.

Keywords : Artificial Intelligence, Intelligent Assistant, Multi-Agent System, Retrieval-Augmented Generation (RAG), Internal Information Systems

Résumé

Ce mémoire présente le développement d'un assistant intelligent interne pour BNP Paribas El Djazaïr, visant à faciliter l'accès à l'information métier et aux procédures internes. L'architecture retenue repose sur un système **multi-agents spécialisé**, où chaque agent exécute une tâche précise (reformulation, recherche, synthèse, conformité, etc.), permettant une gestion modulaire, flexible et explicable des requêtes utilisateurs. Le cœur technologique s'appuie sur la méthode **Retrieval-Augmented Generation (RAG)** combinant la recherche vectorielle rapide et la génération automatique de réponses. Notre système vise à optimiser la performance opérationnelle, la traçabilité et la qualité des réponses, tout en garantissant la conformité aux règles internes. Le prototype développé inclut une interface ergonomique et un pipeline intelligent, démontrant la pertinence du modèle multi-agents pour les assistants métier complexes.

Mots-clés : Intelligence artificielle, Assistant intelligent, Système multi-agents, Retrieval-Augmented Generation (RAG), Systèmes d'information internes.

Table des matières

Liste des tableaux

Liste des figures

Liste des Abréviation

Introduction Générale..... 15

Première partie : Etat des lieux 20

Chapitre 1 : Présentation de BNP Paribas El Djazaïr et du département Data & Performance Opérationnelle 21

1.1 Introduction.....21

1.2 Présentation de BNP Paribas à l'international 21

1.3 Implantation mondiale.....21

1.4 Présentation de BNP Paribas El Djazaïr 22

1.5 Implantation en Algérie 22

1.6 Répartition des agences et centres d'affaires de BNP Paribas en Algérie 22

1.7 Services proposés.....23

1.8 Organigramme de BNP Paribas en Algérie 23

1.9 Présentation du Département Data et Performance Opérationnelle 25

1.10 Conclusion 26

Chapitre 2 : Problématique et Description du Projet 27

2.1 Introduction.....27

2.2 Contexte et Enjeux 27

2.3 Justification du Projet.....28

2.4 Objectifs du Projet 29

2.5 Méthodologie et Approche Adoptée 29

2.6 Résultats Attendus 30

2.7 Conclusion 31

Deuxième partie : Etat de l'art..... 32

Chapitre 3 : Fondements de l'Intelligence Artificielle..... 33

3.1 Définitions de l'intelligence artificielle 33

3.2 Grandes familles de l'IA 33

3.3 Applications et limites historiques de l'intelligence artificielle 34

3.4 Émergence des modèles de langage (LLM) et des architectures Transformers 36

3.4.1 Historique du NLP 36

3.4.2 Principes des LLMs	37
3.4.3 Modèles phares (BERT, GPT, T5, LLaMA...)	39
3.4.4 Limites des modèles LLM purs	40
3.5 Conclusion	41
Chapitre 4 : Pipeline Retrieval-Augmented Generation (RAG).....	43
4.1 La nécessité du RAG : hybridation retrieval + generation.....	43
4.2 Concept du RAG et pipeline classique	44
4.3 Limites du pipeline monolithique RAG.....	45
4.4 Les systèmes multi-agents (SMA) : principes, évolution et impact sur le RAG.....	46
4.4.1 Principes et définitions des systèmes multi-agents.....	46
4.4.2 Typologies d'agents et spécialisations	47
4.4.3 Orchestration et coordination des agents.....	47
4.4.4 Avantages des systèmes multi-agents pour l'IA moderne	47
4.5 Application au RAG : passage du monolithe au RAG agentif et exemples industriels	48
4.6 Comparatif : RAG classique vs RAG agentif.....	49
4.7 Cas d'usage et exemples pratiques	50
4.8 Conclusion	51
Troisième Partie : Conception de la solution.....	52
Chapitre 5 : conception de l'architecture de la solution.....	53
5.1 Introduction.....	53
5.1.1 Rappel du contexte et des enjeux.....	53
5.2 Conception de l'architecture fonctionnelle du pipeline RAG.....	54
5.3 Approche I : Pipeline RAG classique détaillé	56
5.3.1 Ingestion et prétraitement	56
Structuration finale des fragments	61
5.3.2 Encodage et indexation	62
5.3.3 Retrieval et reranking.....	64
5.3.4 Génération et validation	67
5.4 Bilan de l'approche RAG classique : Vers une architecture multi-agents.....	69
5.5 Approche II : Solution multi-agents détaillée	69
5.5.1 Pourquoi une architecture multi-agents ?.....	69
5.5.2 Définition des agents	71
5.5.2 Orchestration multi-agent : Agent Manager et gestion des pipelines	76

5.5.3 Conclusion de l'approche II : Solution multi-agents	78
6.1 Introduction interface	80
6.2 Fonctionnalités clés (import, historique, feedback)	80
6.3 Architecture distribuée cible pour le déploiement	82
6.3.1 Déploiement sur infrastructure serveur	84
6.4 Évaluation comparative : Pipeline RAG Classique vs Architecture Multi-Agents	86
6.5 Conclusion	87
Conclusion Générale	88
Références	90
Annexes	98
Annexe 1 : Fondements de l'intelligence artificielle	99
Annexe 2 : Les Systèmes Multi-Agents (SMA)	110
Annexe 3 : Interface utilisateur Streamlit et aspects pratiques	124

Liste des tableaux

Tableau 4.1 : Comparatif des caractéristiques clés : RAG Monolithique vs RAG Agentif ...	50
Tableau 5.1 : Comparaison des performances entre deux stratégies de prompting.....	68
Tableau 6.1 : Comparatif des performances	86

Liste des figures

Figure 1.1 : Répartition géographique des agences et centres d'affaires de BNP Paribas en Algérie.....	23
Figure 1.2 : Organigramme de BNP Paribas en Algérie	25
Figure 1.3 : Organigramme du pôle Data et Performance Opérationnelle – BNP Paribas El Djazaïr	26
Figure 3.1 : Architecture du mécanisme d'un transformer (source : medium).....	37
Figure 3.2 : Architecture de Prompting pour Modèle de Langage (source : blend)	39
Figure 4.1 : Architecture fonctionnelle d'un système RAG avec intégration de bases de données, moteur de récupération et validation des réponses	45
Figure 4.2 : Schéma fonctionnel des agents : recherche, génération, évaluation et coordination	49
Figure 5.1 : Schéma de fonctionnement d'un système RAG (Retrieval-Augmented Generation)	55
Figure 5.2 : Extraction de texte PDF.....	57
Figure 5.3 : Découpage hybride de texte	58
Figure 5.4 : Fusion de blocs de texte.....	59
Figure 5.5 : Vérification informative	60
Figure 5.6 : Nettoyage de texte	61
Figure 5.7 : Gestion du cache d'embeddings.....	63
Figure 5.8 : Récupération de passages	64
Figure 5.9 : Récupération avec repli	65
Figure 5.10 : Classement de pertinence	66
Figure 5.11 : Génération de prompt contextuel	67
Figure 5.12 : Évaluation de réponse.....	68
Figure 5.13 : Diagramme représentant le flux de l'assistant.....	71
Figure 6. 1 : Architecture back-end de l'assistant intelligent.....	80
Figure 6.2 : Interface principale de l'assistant-écran d'accueil.....	81
Figure 6.3 : Vue de l'option « Historique des conversations »	81

Figure 6.4 : Interface de messagerie avec options de feedback utilisateur	82
Figure 6.5 : Architecture d'un système de micro services avec orchestration de conteneurs source (oracle)	83

Liste des Abréviation

Abréviation	Signification
IA	Intelligence Artificielle
ML	Machine Learning
NLP	Natural Language Processing (Traitement Automatique du Langage Naturel)
LLM	Large Language Model (Modèle de Langage de Grande Taille)
RAG	Retrieval-Augmented Generation
OCR	Optical Character Recognition
FAISS	Facebook AI Similarity Search
SMA s	Systèmes Multi-Agents
SSO	Single Sign-On
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pretrained Transformer
T5	Text-to-Text Transfer Transformer
XLNet	eXtreme Language Understanding Net
PaLM	Pathways Language Model
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
MPCAgent	Multi-Pipeline Control Agent
API	Application Programming Interface
RLHF	Reinforcement Learning from Human Feedback
BGE M3	BAAI General Embedding Model, version M3
PyPDF2	Python PDF library v2
FIPA	Foundation for Intelligent Physical Agents
CrewAI	Framework open-source pour agents LLM orchestrés

AutoGen	Framework Microsoft multi-agents pour LLM
LangChain	Framework de développement d'agents et de chaînes LLM
MetaGPT	Framework agentif basé sur GPT
RAGChain	Architecture de RAG multi-agents
Word2Vec,	Méthodes d'embedding lexical
HMM	Hidden Markov Model
CRF	Conditional Random Field
MCP	Model Context Protocol

Introduction Générale

Introduction Générale

Contexte

Dans un monde en constante mutation, les entreprises doivent faire face à une quantité croissante de données textuelles internes, souvent non structurées, éparpillées dans divers documents métiers, procédures, guides, FAQ, et échanges internes. Dans ce contexte, l'intelligence artificielle (IA), et plus particulièrement le **traitement automatique du langage naturel (TALN)** et les **modèles de langage de grande taille (LLMs)**, offrent des perspectives prometteuses pour automatiser l'accès à l'information, favoriser la prise de décision et améliorer l'efficacité opérationnelle.

Les récents progrès technologiques, notamment l'émergence des modèles tels que **GPT**, **BERT**, ou encore **Mistral**, ont permis de générer automatiquement des réponses à des questions complexes, tout en tenant compte du contexte linguistique. Cependant, leur application directe en entreprise se heurte à plusieurs limites à savoir : hallucination de contenu, manque de traçabilité, difficulté à manipuler des bases documentaires internes, et exigence élevée en termes de sécurité et confidentialité.

Pour répondre à ces défis, les approches hybrides comme la **génération augmentée par récupération (RAG)** ont vu le jour. En combinant un moteur de recherche sémantique performant avec un modèle génératif, le RAG permet de produire des réponses fondées sur des documents internes, garantissant ainsi la **précision, la fiabilité et la contextualisation de la réponse**.

Problématique

Au sein de BNP Paribas El Djazaïr, comme dans de nombreuses institutions financières, l'information stratégique est souvent dispersée dans des silos documentaires, rendant son exploitation fastidieuse. Les collaborateurs doivent souvent consulter manuellement plusieurs sources pour obtenir une réponse précise, ce qui engendre **des pertes de temps, une surcharge cognitive et un risque d'erreurs**.

Dès lors, la mise en place d'un **assistant intelligent interne** devient une nécessité pour faciliter l'accès à l'information, automatiser certaines tâches récurrentes, et **réduire la charge opérationnelle** sur les équipes métiers. Cependant, cette initiative soulève plusieurs défis, à savoir :

- Comment structurer efficacement des documents textuels hétérogènes ?
- Comment garantir la pertinence et l'exactitude des réponses générées ?

- Quelle architecture logicielle permettrait de concilier performance, modularité et traçabilité ?
- Et surtout, comment assurer une **adéquation parfaite** avec les contraintes internes du secteur bancaire (sécurité, conformité, confidentialité) ?

Objectifs

Ce projet de fin d'études s'inscrit dans une volonté d'apporter une réponse concrète à ces problématiques. Il a pour ambition de concevoir et développer un assistant intelligent interne, basé sur une architecture RAG optimisée et orchestrée par un système multi-agent. Les objectifs principaux sont les suivants :

- Concevoir une **pipeline RAG** adaptée aux documents internes de BNP Paribas El Djazaïr, intégrant :
 - Un système de **nettoyage et de segmentation sémantique** des textes.
 - Un moteur de recherche vectorielle à base d'**embeddings multilingues optimisés**.
 - Un modèle génératif conditionné par des documents pertinents.
- Explorer de nouvelles perspectives autour des systèmes RAG, notamment à travers l'implémentation d'une architecture innovante basée sur les **systèmes multi-agents (SMA)**, où chaque agent est spécialisé dans une tâche précise (recherche, filtrage, traitement des requêtes, gestion des logs, etc.).
- Offrir un système **explicable, traçable et personnalisable**, capable d'évoluer selon les besoins des métiers.
- Valider les performances du système selon des critères mesurables (précision, temps de réponse, taux de couverture documentaire, etc.).
- Démontrer son impact positif sur l'**efficacité opérationnelle** et la **satisfaction utilisateur**.

Organisation du mémoire

Ce mémoire est structuré en quatre parties principales. La **Première Partie : État des lieux** présente le contexte du projet et les défis rencontrés. La **Deuxième Partie : État de l'art** explore la littérature et concepts liés à notre projet. Enfin, La **Troisième Partie : Conception de la solution** décrit en détail la solution développée et son architecture.

Première Partie : État des lieux

Cette partie contextualise le projet au sein de BNP Paribas El Djazaïr. Elle décrit l'organisation cible, identifie les besoins métier et analyse les solutions existantes.

- Chapitre 1 : Présentation de BNP Paribas El Djazaïr et du département Data & Performance Opérationnelle" ce chapitre présente l'historique de la filiale, son organigramme, le rôle et les responsabilités du département Data, ainsi que les enjeux liés à la digitalisation et les objectifs stratégiques de l'entreprise.

- Chapitre 2 : Problématique et Description du Projet ce chapitre aborde les enjeux liés à la gestion et à la diffusion des connaissances internes, puis présente le projet d'assistant intelligent utilisant la technologie Retrieval-Augmented Generation (RAG) et une architecture multi-agents, en définissant les objectifs du projet et la méthodologie adoptée pour sa mise en œuvre.

Deuxième Partie : État de l'art

Cette deuxième partie pose les bases théoriques et technologiques de notre assistant intelligent multi-agents. Elle couvre les fondements de l'IA, du NLP, des modèles de langage et des approches Retrieval-Augmented Generation (RAG), ainsi que les principes des architectures du système multi agent et cette partie se divise en deux chapitres :

- Chapitre 3 : Fondements de l'Intelligence Artificielle et du NLP

Ce chapitre présente les concepts clés de l'IA et du traitement automatique du langage : apprentissage supervisé et non supervisé, réseaux de neurones profonds, embeddings, et pré-traitement NLP (tokenisation, lemmatisation, etc.).

- Chapitre 4 : Modèles de Langage, RAG et Architectures Multi-Agents

Présentation de la méthode Retrieval-Augmented Generation (RAG), ses limites, et la montée en puissance des systèmes multi-agents pour en surmonter les failles.

Troisième Partie : Conception de la solution

Cette dernière partie détaille la définition, le développement et l'évaluation de notre prototype d'assistant RAG multi-agents.

- Chapitre 5 : Architecture fonctionnelle et spécifications techniques Ce chapitre présente les diagrammes d'architecture, décrit les modules principaux (ReceptionAgent, RetrieverAgent, AnalysisAgent, RedactionAgent, AgentManager), et

explique les choix technologiques adoptés, notamment LocalAI, FAISS et CrewAI, pour la mise en œuvre de la solution.

- Chapitre 6 : Déploiement, interface utilisateur et évaluation

Déploiement de la solution, conception d'une interface utilisateur intuitive et évaluation de ses performances dans un environnement réel.

Conclusion et perspectives

Bilan des résultats (impacts sur la satisfaction client et l'efficacité opérationnelle), limites rencontrées, et voies d'amélioration : extension à d'autres services bancaires, boucles de feedback pour apprentissage continu, intégration de nouvelles fonctionnalités IA.

Première partie : Etat des lieux

Chapitre 1 : Présentation de BNP Paribas El Djazaïr et du département Data & Performance Opérationnelle

1.1 Introduction

Ce chapitre présente BNP Paribas El Djazaïr, filiale du groupe BNP Paribas, en mettant en lumière son rôle stratégique dans le secteur bancaire algérien. Nous détaillerons sa structure organisationnelle et son implantation en Algérie, ainsi que les services bancaires qu'elle propose. Une attention particulière sera portée sur le département Data et Performance Opérationnelle, qui est essentiel pour l'amélioration des processus internes et la mise en œuvre des technologies numériques avancées. Ce département contribue activement à la digitalisation des services bancaires, alignant BNP Paribas El Djazaïr avec la vision de transformation numérique du secteur bancaire en Algérie.

1.2 Présentation de BNP Paribas à l'international

BNP Paribas est l'un des principaux groupes bancaires européens, présent dans 64 pays et territoires à travers le monde, avec près de 178 000 collaborateurs. Le groupe offre une gamme complète de services financiers (BNP Paribas, n.d.b), incluant :

- Banque de détail
- Banque d'investissement
- Gestion d'actifs
- Assurance et les services de financement spécialisés

1.3 Implantation mondiale

BNP Paribas est structuré autour de trois divisions principales (BNP Paribas, n.d.a) :

- Banque de détail en Europe : Présence significative en France, Belgique, Italie, Luxembourg, et Pologne.
- Banque de financement et d'investissement : Activités dans les marchés financiers, la banque d'affaires et les services aux institutions.
- Services aux clients institutionnels et entreprises : Offre de solutions adaptées aux besoins des grandes entreprises et institutions financières. Le groupe est également engagé dans des initiatives de finance durable, visant à soutenir la transition énergétique et à promouvoir des investissements responsables.

1.4 Présentation de BNP Paribas El Djazaïr

BNP Paribas El Djazaïr est une filiale à 100 % du groupe BNP Paribas, établie en Algérie depuis 2002. Elle est l'une des rares banques internationales présentes dans le pays, offrant une gamme complète de services bancaires et financiers aux particuliers, professionnels, et entreprises (BNP Paribas El Djazaïr, n.d.a).

1.5 Implantation en Algérie

Le siège social de BNP Paribas El Djazaïr est situé à Bab Ezzouar, Alger, dans un bâtiment moderne de 30 000 m². La banque dispose également de 13 centres d'affaires répartis à travers le pays, facilitant l'accès à ses services pour une clientèle diversifiée (BNP Paribas El Djazaïr, n.d.a).

1.6 Répartition des agences et centres d'affaires de BNP Paribas en Algérie

La carte présentée (Figure 1.1) illustre la couverture géographique des agences et centres d'affaires de BNP Paribas sur le territoire algérien, mettant en évidence une présence particulièrement dense dans la région d'Alger avec 33 agences et 4 centres d'affaires dédiés. Cette organisation témoigne de la stratégie de la banque visant à être proche de ses clients, qu'ils soient particuliers ou professionnels, en offrant des services adaptés à leurs besoins spécifiques.

Les symboles verts représentent les agences destinées aussi bien aux particuliers qu'aux professionnels, tandis que les centres d'affaires, identifiés par une bande orange, sont conçus pour accueillir principalement les entreprises et leurs besoins financiers spécifiques. On note une forte concentration d'agences dans les grandes villes telles que Oran, Constantine, Annaba, et Béjaïa, ainsi qu'une couverture équilibrée des autres wilayas, assurant ainsi une présence locale solide sur l'ensemble du pays.

Cette répartition géographique illustre la volonté de BNP Paribas El Djazaïr d'accompagner efficacement ses clients dans toutes les régions, en combinant proximité, expertise sectorielle et services dédiés aux différentes catégories de clientèle.

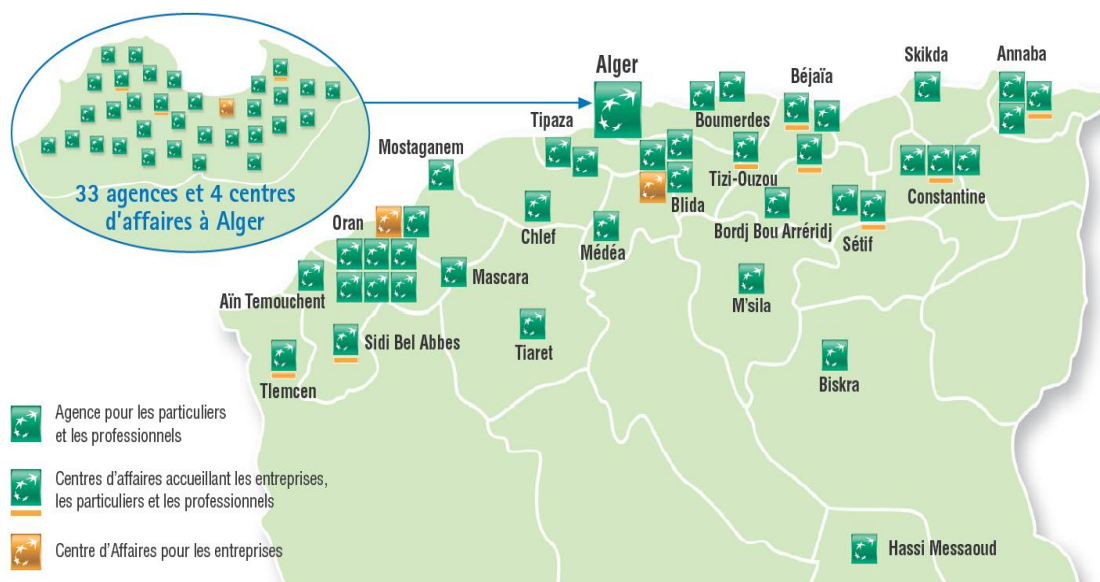


Figure 1.1 : Répartition géographique des agences et centres d'affaires de BNP Paribas en Algérie

Source : (BNP Paribas El Djazaïr, n.d.c)

1.7 Services proposés

Maintenant, nous allons explorer les différents services bancaires proposés par BNP Paribas El Djazaïr, en détaillant les solutions disponibles pour les particuliers et les entreprises (BNP Paribas El Djazaïr, n.d.e) :

- Banque de détail : Comptes courants, épargne, crédits à la consommation, cartes bancaires.
- Banque pour les professionnels : Solutions de financement, gestion de trésorerie, services de paiement.
- Banque pour les entreprises : Financement de projets, commerce international, gestion des risques.
- Services digitaux : Plateformes en ligne et applications mobiles pour une gestion bancaire simplifiée.

1.8 Organigramme de BNP Paribas en Algérie

L'organigramme de BNP Paribas en Algérie (Figure 1.2) reflète une structure organisationnelle complexe, centrée autour de deux niveaux principaux : le **Conseil**

d'Administration et la direction exécutive, représentée par le **Directeur Pays Algérie Général**.

Sous la supervision du Directeur Pays, plusieurs départements fonctionnels assurent la gestion des différentes activités de la banque. Parmi eux, des fonctions clés comme l'audit, la conformité, le juridique, et le risque sont gérées par des comités spécifiques. Le **Secrétaire Général** joue un rôle essentiel dans la coordination interne et la gestion des ressources, avec des départements comme le **COO Office**, la **DSI** (Systèmes d'Information), et les **Relations Clients & Servicing** sous sa responsabilité.

L'entité BNP Paribas en Algérie s'engage également dans des initiatives de durabilité et de marketing, tout en gérant des aspects financiers et humains à travers des départements comme les **Ressources Humaines** et la **Finance**.

Enfin, le **DG Adjoint** supervise les départements stratégiques tels que **Capital Market** et **Recovery**, tandis que les divisions **Retail** et **Corporate** ciblent respectivement les services bancaires destinés aux particuliers et aux entreprises.

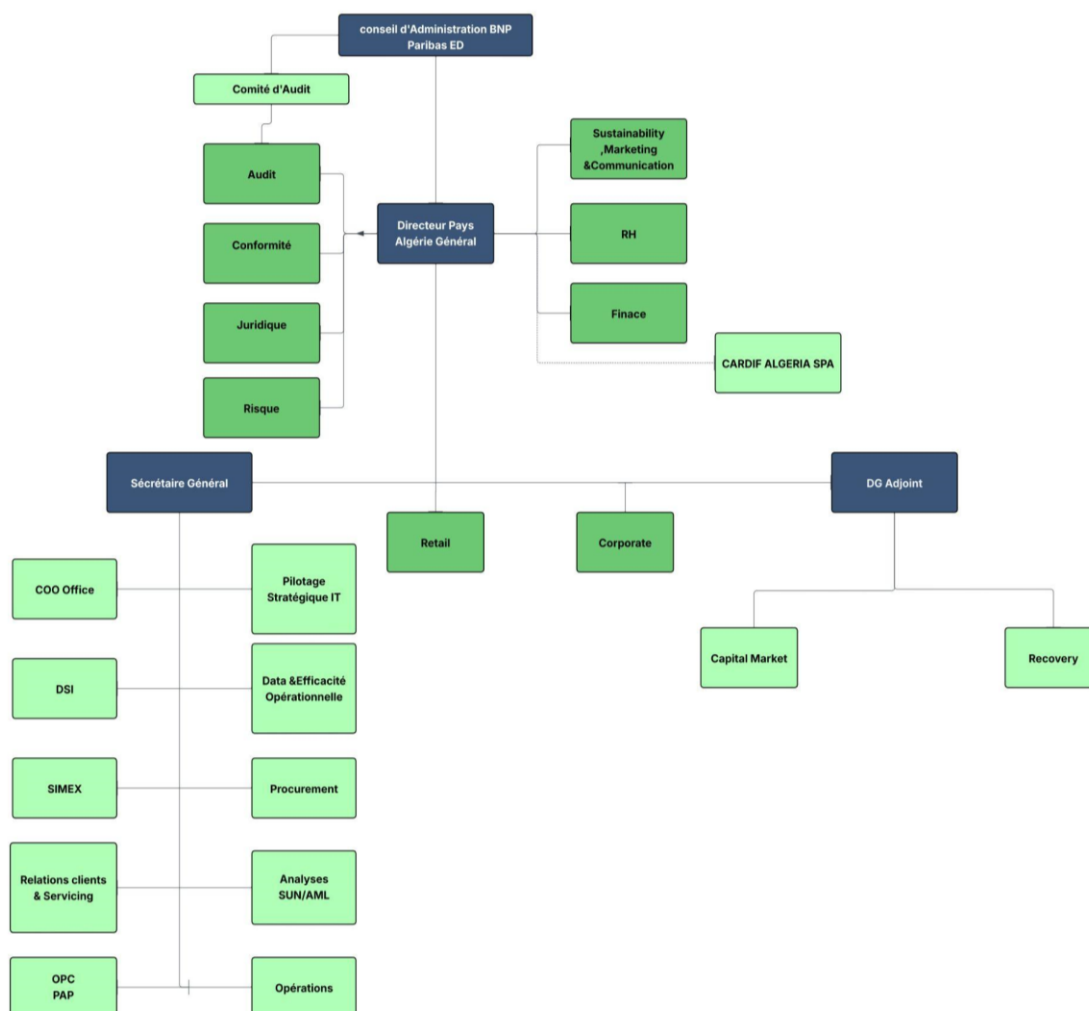


Figure 1.2 : Organigramme de BNP Paribas en Algérie

1.9 Présentation du Département Data et Performance Opérationnelle

Le département **Data et Performance Opérationnelle** comme on peut le voir dans la figure (Figure 1.3) joue un rôle central dans l'amélioration des processus internes et de la gestion des données de BNP Paribas El Djazair. Ce département est en charge de l'exploitation des **données stratégiques**, de l'**optimisation de la performance opérationnelle** et de l'**intégration de solutions numériques** afin de renforcer l'efficacité des services bancaires.

Le département **Data et Performance Opérationnelle** est dédié à la **transformation numérique** et à l'amélioration continue des **processus bancaires**. Il travaille étroitement avec les autres départements pour :

- **Collecter, traiter et analyser les données bancaires et financières** pour en tirer des **insights stratégiques**.

- **Optimiser les processus internes** en utilisant des outils technologiques avancés comme l'**intelligence artificielle** et le **big data**.
- **Améliorer l'efficacité des services bancaires** et la **réduction des coûts opérationnels** en **automatisant les processus**.

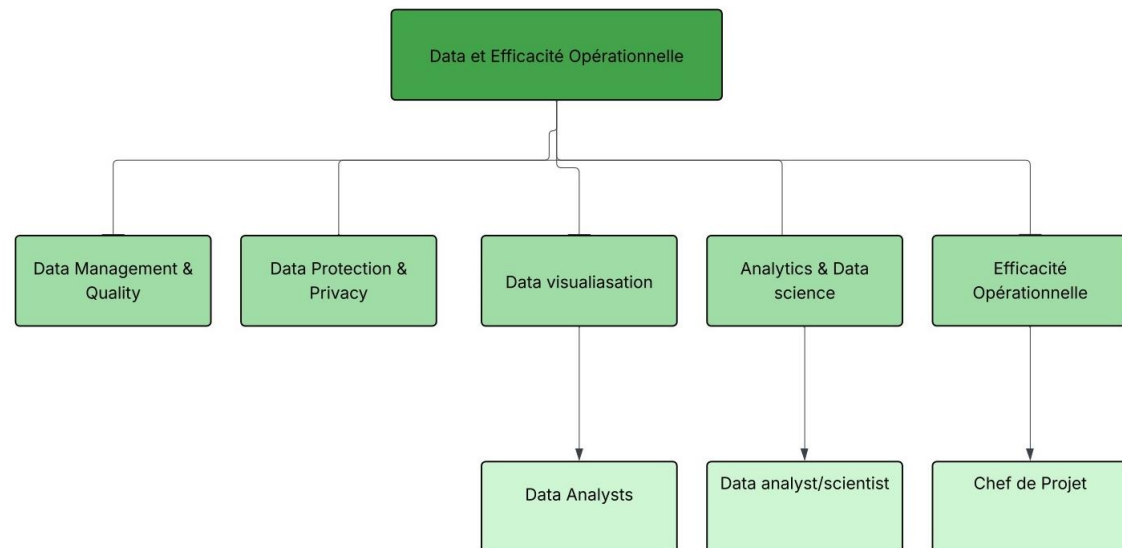


Figure 1.3 : Organigramme du pôle Data et Performance Opérationnelle – BNP Paribas El Djazair

1.10 Conclusion

Cette section a permis de poser les bases pour comprendre l'organisation de BNP Paribas El Djazair, une filiale stratégique du groupe BNP Paribas, et son rôle clé dans la digitalisation du secteur bancaire algérien. Nous avons exploré l'envergure du groupe à l'international et l'impact local de BNP Paribas El Djazair, avec un accent particulier sur le département Data et Performance Opérationnelle. Ce département joue un rôle essentiel dans la transformation numérique en exploitant des technologies avancées pour améliorer l'expérience client et les performances bancaires. Le projet de développement d'un chatbot intelligent basé sur l'IA générative et le NLP s'inscrit dans cette dynamique, visant à renforcer la relation client et améliorer l'efficacité opérationnelle.

Chapitre 2 : Problématique et Description du Projet

2.1 Introduction

Dans un contexte bancaire marqué par une transformation digitale rapide, l'accès à une information fiable et sécurisée est essentiel pour BNP Paribas El Djazaïr. La gestion du volume croissant de documents et leur mise à jour fréquente représente un défi majeur pour les collaborateurs.

Ce chapitre présente les difficultés liées à la gestion des connaissances internes et décrit le projet d'assistant intelligent pour y répondre. La première section contextualise la problématique et les défis rencontrés par les utilisateurs. La deuxième section clarifie les objectifs du projet et les buts à atteindre. Enfin, la méthodologie adoptée, utilisée pour la conception de notre "Retrieval-Augmented Generation" (RAG). Ce chapitre offre une vue d'ensemble des enjeux, objectifs et démarche adoptée pour la solution proposée.

2.2 Contexte et Enjeux

Dans un environnement bancaire en constante évolution et fortement réglementé, BNP Paribas El Djazaïr doit faire face à un défi majeur : la gestion efficace et rapide d'une vaste quantité de documents internes, tels que des procédures, guides, rapports d'activité et directives réglementaires. Ces documents, souvent volumineux, dispersés et hétérogènes dans leurs formats, rendent leur exploitation rapide et efficace difficile. De plus, l'accès à une information fiable, précise et à jour est essentiel pour assurer la conformité, maintenir la qualité des services et prendre des décisions éclairées.

Les collaborateurs doivent souvent naviguer à travers diverses sources d'information, un processus manuel lourd et chronophage, augmentant ainsi le risque d'erreurs et de pertes de temps. Cette complexité documentaire met en lumière l'urgence d'une solution intelligente qui centralise, structure et restitue les connaissances internes de manière rapide, précise et conforme, tout en répondant aux besoins métier spécifiques de BNP Paribas El Djazaïr.

L'enjeu principal est de faciliter l'accès à l'information et d'améliorer la performance opérationnelle tout en garantissant une conformité stricte aux exigences réglementaires. Ce projet vise à développer un assistant intelligent capable de répondre rapidement et de manière contextualisée aux questions des collaborateurs, simplifiant l'accès à l'information et assurant une sécurité renforcée dans un environnement complexe.

En réponse à ces défis, une solution innovante s'impose. Cette solution doit être capable de traiter une grande variété de documents, y compris ceux numérisés, en intégrant des technologies avancées telles que la reconnaissance optique de caractères (OCR) pour garantir l'extraction complète et de qualité des données. Le système doit également garantir une performance élevée, avec une faible latence et une disponibilité continue, tout en assurant que les réponses fournies soient pertinentes et fiables.

Le projet se concentrera sur la mise en place d'une chaîne complète de traitement documentaire (ingestion, nettoyage, segmentation, vectorisation, indexation), l'implémentation de la technique **Retrieval-Augmented Generation (RAG)** pour combiner recherche documentaire et génération de texte contextuel, et l'intégration d'une interface utilisateur intuitive adaptée aux besoins métier. Enfin, une boucle de feedback permettra d'améliorer continuellement le système en fonction des retours utilisateurs, assurant ainsi la pérennité et la pertinence de la solution.

Dans le cadre de l'étude de l'existant, BNP Paribas El Djazaïr a déjà entrepris un premier prototype d'assistant intelligent visant à répondre aux besoins de gestion documentaire. Cependant, ce prototype rencontrait plusieurs limites majeures. Tout d'abord, le temps de réponse était trop long, dépassant les 3 minutes pour fournir une réponse, ce qui rendait l'outil peu performant dans un environnement bancaire où la rapidité d'accès à l'information est essentielle. En outre, la précision des réponses n'était pas optimale, avec des informations souvent inexactes ou mal contextualisées. Cette situation a mis en évidence la nécessité d'optimiser à la fois la vitesse de réponse et la pertinence des informations fournies. Ainsi, la problématique principale du projet actuel se concentre sur l'optimisation de ces aspects : réduire le temps de réponse tout en garantissant des réponses précises, fiables et adaptées aux besoins des collaborateurs de la banque.

2.3 Justification du Projet

La création d'un assistant intelligent interne s'appuie sur plusieurs facteurs déterminants :

- **Volume et complexité des données** : La grande quantité et la diversité des documents métiers rendent la recherche manuelle fastidieuse et peu fiable.
- **Conformité réglementaire** : La mise à jour constante des normes impose une veille continue et un accès rapide à l'information la plus récente.
- **Efficacité des collaborateurs** : L'assistant intelligent réduit le temps consacré à la recherche documentaire, augmentant ainsi la productivité.

- Innovation technologique : L'adoption de technologies avancées, notamment la combinaison de RAG et de systèmes multi-agents, offre une réponse adaptée aux besoins spécifiques de la banque.

2.4 Objectifs du Projet

Les objectifs de ce projet se divisent en deux catégories principales les suivantes :

Objectifs Généraux

Développer un assistant intelligent interne capable de :

- Fournir des réponses rapides, précises et contextualisées aux collaborateurs de BNP Paribas El Djazaïr.
- Simplifier et sécuriser l'accès à l'information métier dans un environnement complexe.
- Maintenir une conformité stricte aux exigences réglementaires et internes.

Objectifs Spécifiques

- Mettre en place une chaîne complète de traitement documentaire : ingestion, nettoyage, segmentation, vectorisation et indexation des documents.
- Développer une architecture solide qui répond à toutes les attentes de BNP Paribas qui reste optimisée et pratique à l'utilisation.
- Implémenter la technique Retrieval-Augmented Generation (RAG) pour combiner recherche documentaire et génération de texte contextuel.
- Assurer la traçabilité et la qualité des réponses en mettant en place des logs qui nous permettent de surveiller ce qui se passe à chaque niveau.
- Intégrer une interface utilisateur intuitive adaptée aux besoins métier.
- Instaurer une boucle de feedback pour améliorer continuellement le système selon les retours utilisateurs.

2.5 Méthodologie et Approche Adoptée

La mise en œuvre de notre assistant intelligent repose sur une méthodologie rigoureuse, inspirée du cadre CRISP-DM, reconnu pour sa structure itérative et sa capacité à aligner les objectifs métier avec les différentes étapes d'un projet d'intelligence artificielle, ce qui le rend particulièrement adapté à notre contexte industriel [CRISP-DM, 2000]. Cette démarche s'articule autour de plusieurs étapes clés, visant à tirer pleinement parti de l'approche RAG (Retrieval-Augmented Generation) pour offrir des réponses précises, contextuelles et exploitables à partir des documents internes de l'entreprise.

1) Collecte et préparation des documents

La première étape a consisté à rassembler un ensemble de documents internes pertinents : procédures, rapports, directives, notes de service, etc. Ces documents ont été nettoyés, standardisés et convertis dans un format uniforme afin de faciliter les étapes de traitement ultérieures.

2) Segmentation et vectorisation

Les documents préparés ont été découpés en segments textuels cohérents (chunks), de manière à garantir un bon équilibre entre la taille du contexte et la pertinence sémantique. Chaque segment a ensuite été transformé en vecteur numérique à l'aide de modèles d'embeddings spécialisés, capables de capturer les similarités de sens entre les textes.

3) Indexation et recherche vectorielle

Les vecteurs obtenus ont été stockés dans une base FAISS (Facebook AI Similarity Search), un moteur de recherche vectorielle performant. Ce composant permet d'identifier rapidement, pour chaque requête utilisateur, les segments de documents les plus pertinents en fonction de leur proximité sémantique.

4) Génération augmentée par récupération (RAG)

Une fois les documents pertinents identifiés, ils sont utilisés comme contexte d'entrée pour un modèle de génération de langage naturel. Ce modèle, entraîné à comprendre les relations entre questions et contenus, produit une réponse synthétique et contextualisée à partir des informations extraites.

5) Déploiement et interface utilisateur

L'ensemble du système a été intégré dans une interface web interactive, développée avec Streamlit, permettant aux utilisateurs finaux d'interagir facilement avec l'assistant. L'interface offre une expérience fluide : saisie de requêtes, affichage des réponses, visualisation des extraits sources, et mécanisme de retour utilisateur pour l'amélioration continue.

Cette démarche garantit un équilibre entre performance, précision, conformité et expérience utilisateur, tout en respectant les contraintes spécifiques au secteur bancaire.

2.6 Résultats Attendus

Les résultats attendus après la conception de notre solution sont :

- Un assistant intelligent performant, capable de répondre efficacement aux questions internes métier.

- Une réduction significative du temps de recherche documentaire.
- Une amélioration de la conformité et de la qualité de l'information diffusée.
- Une interface intuitive, facilitant l'adoption par les collaborateurs.
- Un système évolutif et adaptable aux nouveaux besoins et sources de données.

2.7 Conclusion

Ce chapitre a présenté la problématique de gestion et de diffusion des connaissances internes à BNP Paribas El Djazaïr, en mettant en lumière les enjeux d'efficacité, de conformité et de sécurité. Il a justifié la nécessité d'un assistant intelligent multi-agents basé sur la technologie RAG, adapté aux besoins spécifiques du département Data et Performance Opérationnelle. Les objectifs clairs et précis ont été définis pour répondre aux exigences métier. Enfin, la méthodologie adoptée a été détaillée, posant ainsi les fondations techniques et fonctionnelles de la solution innovante développée. Ce cadre solide prépare la mise en œuvre concrète présentée dans les chapitres suivants.

Deuxième partie : Etat de l'art

Chapitre 3 : Fondements de l'Intelligence Artificielle

3.1 Définitions de l'intelligence artificielle

L'intelligence artificielle (IA) désigne l'ensemble des théories, méthodes et technologies permettant à des systèmes informatiques de reproduire, simuler ou étendre certaines capacités cognitives humaines : perception, raisonnement, apprentissage, adaptation, et prise de décision.

Depuis sa définition historique par **John McCarthy en 1956** comme « la science et l'ingénierie de la fabrication de machines intelligentes », l'IA s'est imposée comme un domaine pluridisciplinaire à la croisée de l'informatique, des mathématiques, de la psychologie cognitive, et plus récemment des sciences des données.

De façon générale, un système est dit “intelligent” s'il est capable de traiter des informations de son environnement, d'en tirer des conclusions, et d'ajuster ses actions ou ses réponses dans le but d'atteindre un objectif (Dartmouth College, n.d.).

3.2 Grandes familles de l'IA

Au fil de son évolution, l'IA s'est structurée autour de plusieurs grands courants ou “familles”, qui correspondent à différentes approches de la modélisation de l'intelligence (Stanford Encyclopedia of Philosophy, n.d.a):

❖ IA symbolique (ou logique)

Approche historique fondée sur la manipulation de symboles, de règles et de connaissances explicites. Les systèmes experts, les moteurs d'inférence et les méthodes de raisonnement logique (par exemple les systèmes à base de règles ou la logique des prédicats) illustrent cette famille. Elle est réputée pour sa capacité d'explication et de traçabilité, mais atteint rapidement ses limites en présence d'environnements complexes, ambigus ou massifs (Young, n.d.).

❖ IA connexionniste (ou subsymbolique)

Inspirée du fonctionnement du cerveau humain, cette famille regroupe les réseaux de neurones artificiels et, plus généralement, les méthodes d'apprentissage automatique capables de traiter des données non structurées (images, sons, textes). L'émergence du deep learning (apprentissage profond) a fortement accéléré le développement de cette approche, permettant l'apparition de systèmes capables de reconnaître, classifier et générer de grandes quantités d'information.

❖ IA par apprentissage automatique (Machine Learning)

Cette famille inclut toutes les techniques où le système apprend à partir de données, sans être explicitement programmé pour chaque tâche. On distingue l'apprentissage supervisé (le modèle apprend à partir d'exemples annotés), non supervisé (le modèle cherche des régularités ou des structures cachées), et par renforcement (le modèle apprend par essai-erreur en interaction avec son environnement).

❖ IA hybride

Plus récemment, de nombreux travaux proposent de combiner les approches symboliques et connexionnistes afin de profiter des atouts de chacune : la capacité d'apprentissage et la robustesse des réseaux de neurones, couplées à l'explicabilité et à la capacité de raisonnement logique des systèmes symboliques (SpringerLink, n.d.).

Cette diversité de paradigmes a permis à l'IA d'investir des domaines variés (vision par ordinateur, traitement automatique du langage, robotique, jeux, optimisation industrielle...). Toutefois, chaque approche présente des avantages et des limites spécifiques, qui conditionnent le choix des architectures selon les problématiques métier et les volumes de données à traiter.

3.3 Applications et limites historiques de l'intelligence artificielle

Applications majeures de l'IA

Depuis ses débuts, l'intelligence artificielle s'est illustrée dans de nombreux domaines, dépassant souvent le simple cadre académique pour s'imposer dans l'industrie, les services, et la vie quotidienne. Parmi les principales applications, on peut citer :

- **Jeux et résolution de problèmes**

L'IA a d'abord trouvé un terrain d'expérimentation dans les jeux de société (échecs, dames, Go), avec des systèmes capables de battre des champions humains : Deep Blue (IBM, 1997) ou AlphaGo (DeepMind, 2016) sont devenus emblématiques des avancées du domaine.

- **Systèmes experts et automatisation**

Dans les années 1980-90, les systèmes experts (notamment dans la médecine, l'ingénierie ou la finance) ont permis d'automatiser le diagnostic, l'aide à la décision ou la détection d'anomalies, en s'appuyant sur des bases de connaissances explicites.

- **Vision par ordinateur et reconnaissance**

Le traitement d'images (reconnaissance de visages, d'objets, de caractères) s'est généralisé grâce aux progrès du deep learning, avec des applications en sécurité, industrie, santé (radiologie assistée par IA), ou véhicules autonomes.

- **Traitement automatique du langage naturel (NLP)**

L'IA est désormais au cœur des assistants vocaux, des traducteurs automatiques, des systèmes de recommandation ou des moteurs de recherche intelligents. Les progrès dans la compréhension et la génération de langage ouvrent la voie à des usages toujours plus interactifs et sophistiqués (chatbots, résumés automatiques, analyse de sentiments, etc.).

- **Robotique et automatisation industrielle**

La robotique intelligente bénéficie de l'IA pour la planification, la navigation autonome, la manipulation d'objets ou la maintenance prédictive.

Limites historiques

Malgré ces avancées, l'IA a longtemps été confrontée à des limites majeures :

- **Fragilité face à la complexité réelle**

Les premiers systèmes experts étaient efficaces dans des domaines fermés et bien définis, mais échouaient dès que la connaissance était incomplète, incertaine ou sujette à évolution rapide ("brittleness problem").

- **Dépendance à l'exhaustivité des règles**

Les IA symboliques nécessitaient une modélisation manuelle de toutes les règles, ce qui s'avérait coûteux, lent et difficilement scalable à grande échelle.

- **Manque d'apprentissage et d'adaptabilité**

Jusqu'à l'avènement du machine learning, la plupart des systèmes étaient incapables d'apprendre de nouvelles situations ou d'améliorer leurs performances par l'expérience.

- **Problème du "goulet d'étranglement des connaissances"**

La difficulté à intégrer et à structurer de larges volumes de connaissances humaines dans des bases symboliques limitait considérablement l'expansion des systèmes experts.

Limites computationnelles

L'absence de puissance de calcul et de grandes masses de données freinait le développement d'algorithmes plus sophistiqués jusqu'aux années 2000.

Ce n'est qu'avec l'émergence des méthodes connexionnistes et du machine learning, dopées par l'explosion des données et des capacités de calcul, que l'IA a pu franchir ces obstacles historiques, ouvrant la voie à des applications massives et à la démocratisation de l'intelligence artificielle.

3.4 Émergence des modèles de langage (LLM) et des architectures Transformers

3.4.1 Historique du NLP

Le traitement automatique du langage naturel (NLP, pour Natural Language Processing) désigne l'ensemble des techniques permettant à des machines de comprendre, analyser, et générer le langage humain. Depuis les débuts de l'informatique, le NLP a connu une évolution rapide, marquée par plusieurs vagues technologiques.

- **Des approches symboliques au statistical NLP**

Pendant plusieurs décennies, le NLP reposait sur des méthodes symboliques : règles linguistiques, analyse grammaticale, dictionnaires et arbres syntaxiques. Ces systèmes, s'ils étaient capables de traiter des cas simples (analyse syntaxique, correction grammaticale, reconnaissance d'entités), montraient rapidement leurs limites en termes de couverture, d'ambiguïté et d'adaptabilité à la diversité des langues naturelles (Stanford Encyclopedia of Philosophy, n.d.).

L'apparition des méthodes statistiques dans les années 1990 a marqué une rupture : l'apprentissage à partir de corpus annotés a permis l'émergence de modèles probabilistes (Hidden Markov Models, CRF, n-grammes) capables d'aborder des tâches comme la reconnaissance de la parole, la traduction automatique ou la classification de textes avec plus de robustesse et de flexibilité.

- **L'avènement du deep learning et des word embeddings**

À partir des années 2010, le développement du deep learning a bouleversé le NLP :

- Les word embeddings (Word2Vec, GloVe, FastText) ont permis de représenter les mots dans un espace vectoriel dense, capturant leurs similarités sémantiques de façon automatique et non-supervisée.

- Les réseaux de neurones récurrents (RNN, LSTM, GRU) ont apporté la capacité de modéliser des dépendances longues dans les séquences de texte, ouvrant la voie à des progrès significatifs en traduction, analyse de sentiments ou question-réponse (Nature, 2015).

3.4.2 Principes des LLMs

Les modèles de langage de grande taille (LLM), issus de l'architecture Transformer (figure 3.1), se distinguent par plusieurs principes techniques majeurs qui expliquent leur efficacité et leur impact dans le domaine du NLP moderne.

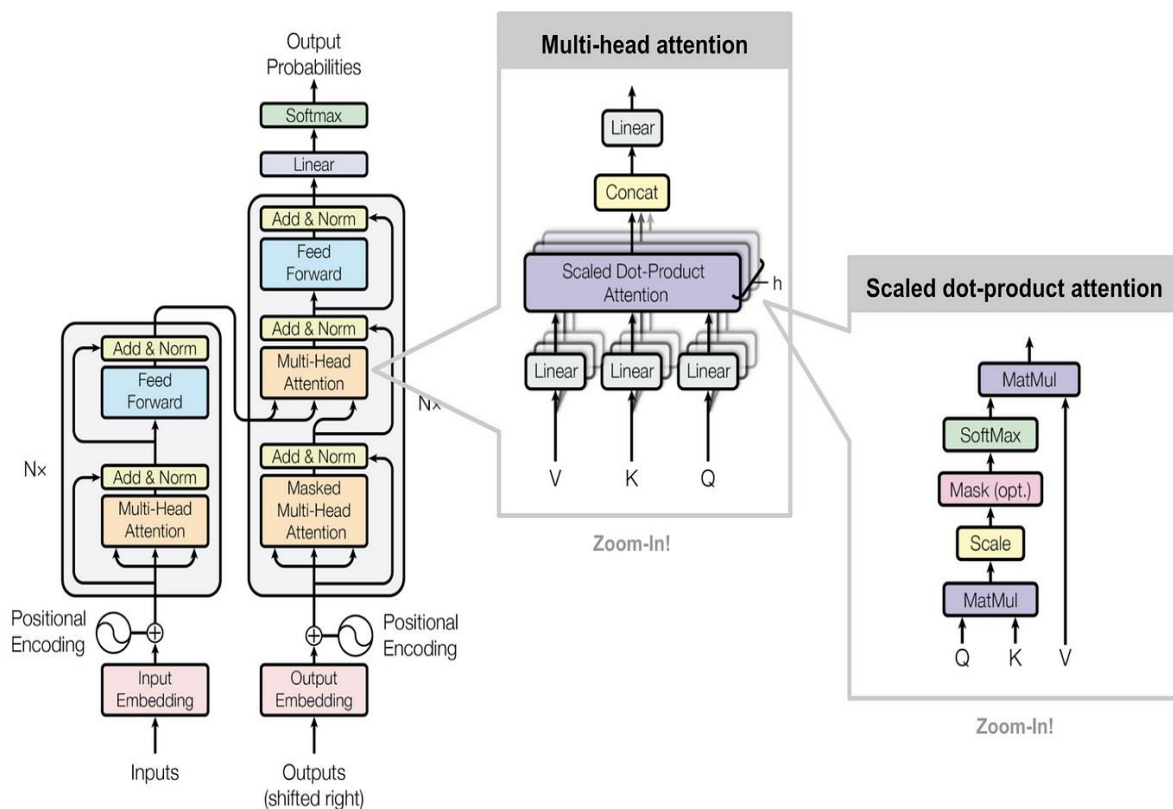


Figure 3.1 : Architecture du mécanisme d'un transformer (source : medium)

1) Mécanisme d'attention (Self-Attention)

Le cœur du Transformer repose sur le mécanisme d'attention : à chaque étape de traitement, le modèle "pondère" chaque mot du texte en fonction de sa pertinence vis-à-vis des autres mots de la séquence, permettant ainsi de capter des dépendances à longue distance. Contrairement aux réseaux séquentiels (RNN, LSTM), cette approche offre un parallélisme total dans l'apprentissage, une compréhension globale du contexte, et une capacité à gérer de très longues séquences.

2) Multi-head attention

Le Transformer utilise plusieurs “têtes” d’attention parallèles (“multi-head attention”), ce qui permet au modèle de capter simultanément différents types de relations sémantiques ou syntaxiques entre les éléments du texte. Cela favorise la richesse des représentations apprises et améliore la capacité de généralisation du modèle (Vaswani et al. 2017).

3) Pré-entraînement sur des corpus massifs

Les LLM sont d’abord pré-entraînés sur d’immenses quantités de textes hétérogènes (livres, web, articles, code...), sans supervision humaine directe. Les tâches typiques sont la prédiction de mots masqués (Masked Language Modeling, comme dans BERT) et la prédiction du prochain mot (causal language modeling, comme dans GPT). Ce pré-entraînement dote les modèles de connaissances générales sur la langue, les faits du monde et les structures syntaxiques ou logiques (Devlin et al. 2018),

4) Fine-tuning (spécialisation)

Après le pré-entraînement, les LLM peuvent être affinés (fine-tuned) sur des jeux de données spécifiques à une tâche ou un domaine (finance, santé, droit...). Cela améliore leur performance sur des applications ciblées tout en conservant leur capacité de généralisation (Raffel et al. 2020).

5) Scaling laws (lois de passage à l’échelle)

Une découverte clé est que l’augmentation de la taille des modèles (nombre de paramètres), des données et de la puissance de calcul conduit à des améliorations continues et prévisibles des performances (scaling laws). Cela explique l’émergence de modèles de plus en plus vastes, capables de généraliser sur de nombreuses tâches à partir d’instructions simples (“prompting”) (Kaplan et al. 2020).

6) Prompting et adaptation

Les LLM modernes sont capables d’apprendre à partir de simples instructions (“prompt engineering”), voire de quelques exemples (few-shot learning) ou même d’une seule démonstration (one-shot learning). Ce mode d’utilisation permet une flexibilité inédite, sans nécessiter de réentraîner le modèle pour chaque nouvelle tâche. la figure 3.2 montre qu’un prompt structure et indique au LLM de quoi l’utilisateur a besoin (Brown et al. 2020).

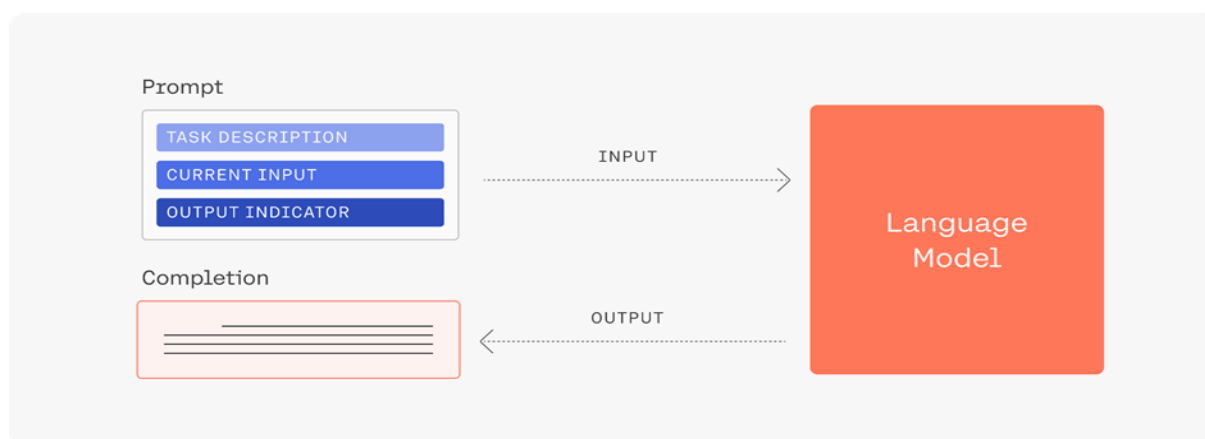


Figure 3.2 : Architecture de Prompting pour Modèle de Langage (source : blend)

3.4.3 Modèles phares (BERT, GPT, T5, LLaMA...)

L'évolution des architectures de type Transformer a donné naissance à une série de modèles phares qui ont marqué des ruptures technologiques et méthodologiques dans le domaine du traitement automatique du langage naturel. Voici un panorama des principaux modèles de référence :

- BERT (Bidirectional Encoder Representations from Transformers) – Google, 2018
BERT est le premier modèle Transformer grand public à exploiter l'apprentissage bidirectionnel sur de vastes corpus de texte. Sa capacité à analyser le contexte de chaque mot à la fois à gauche et à droite a révolutionné les performances sur de nombreuses tâches de NLP : classification, extraction d'entités, réponses à des questions, etc. BERT est pré-entraîné par le masquage aléatoire de mots ("masked language modeling"), puis affiné sur des tâches spécifiques ("fine-tuning") (Devlin et al. 2018),
- GPT (Generative Pretrained Transformer) – OpenAI, 2018-2023+
La famille GPT, de GPT-1 à GPT-4, repose sur un pré-entraînement auto-régressif : le modèle apprend à prédire le prochain mot d'une séquence. Cette approche favorise la génération fluide de texte, la complétion de phrases, la conversation et la créativité. GPT a popularisé l'usage du prompting et du few-shot learning, rendant possible l'adaptation du modèle à de nouvelles tâches sans réentraînement massif.
- T5 (Text-to-Text Transfer Transformer) – Google, 2020
T5 généralise tous les problèmes de NLP sous la forme d'une tâche de "texte-à-texte" : toute entrée (question, texte, résumé, traduction, etc.) et toute sortie sont considérées comme des séquences textuelles. Cela permet une architecture unifiée, performante sur de nombreuses tâches, de la traduction à la synthèse automatique en passant par la réponse à des questions.

- LLaMA (Large Language Model Meta AI) – Meta, 2023+
LLaMA est une famille de LLMs open source conçue par Meta (Facebook) pour offrir des modèles puissants, mais plus compacts et accessibles que GPT-4 ou PaLM. LLaMA se distingue par sa modularité, sa capacité d'adaptation sur des ressources limitées, et son ouverture à la recherche et à l'industrie (Touvron et al. 2023).

- Autres modèles marquants :

- XLNet (Google/CMU, 2019) : combine l'approche bidirectionnelle de BERT et l'auto-régressivité de GPT pour de meilleures performances sur certaines tâches.
- PaLM (Google, 2022) : modèle de très grande taille qui repousse les limites du raisonnement, de la génération et de la compréhension multi-langues.
- Gemini (Google DeepMind, 2024) : modèle de très grande taille qui repousse les limites du raisonnement, de la génération et de la compréhension multi-langues.
Source : Google DeepMind, Gemini
- Bloom (BigScience, 2022) : premier modèle LLM multilingue "open science", collaboratif et éthique, entraîné sur des corpus variés.

En synthèse, ces modèles incarnent la montée en puissance des LLM dans le paysage de l'IA :

Ils servent de base à la plupart des assistants, moteurs de recherche, outils de génération ou d'analyse textuelle contemporains. Leur évolution rapide (taille, performance, ouverture) alimente l'innovation, mais pose aussi de nouveaux défis de scalabilité, de factualité et d'adaptation aux contextes métiers.

3.4.4 Limites des modèles LLM purs

Malgré leurs performances impressionnantes, les LLM présentent plusieurs limites intrinsèques qui restreignent leur utilisation directe dans de nombreux contextes métiers ou applicatifs :

1) Factualité et hallucinations

Les LLM peuvent générer des réponses incorrectes ou inventées, appelées "hallucinations", car ils produisent du texte plausible statistiquement sans accès dynamique à des connaissances actualisées ou à des sources précises. Cette limitation est particulièrement problématique dans les environnements réglementés, scientifiques ou décisionnels (Bender et al. 2021).

2) Actualisation et obsolescence

Une fois entraînés, les LLM ne "voient" que les données disponibles à la date de leur pré-entraînement. Ils ne peuvent intégrer de nouveaux faits, documents ou événements, à moins d'être réentraînés — une opération coûteuse, complexe et rarement possible en pratique (Lewis et al. 2020).

3) Personnalisation et adaptation métier

Un LLM généraliste n'est pas optimisé pour un contexte métier précis (vocabulaire spécifique, processus internes, réglementation locale, etc.). Les adaptations ("fine-tuning") restent limitées et peu scalables dès lors que les besoins métiers évoluent rapidement (Lewis et al. 2020).

4) Explicabilité et traçabilité

Les décisions et réponses des LLM sont souvent difficiles à expliquer : il n'est pas aisé d'identifier quelles données ou associations ont mené à une prédiction. Cette "boîte noire" complique la vérification, la correction et la validation métier ou réglementaire (Bender et al. 2021).

5) Limitations computationnelles et environnementales

L'entraînement et l'inférence de LLM très volumineux nécessitent des ressources de calcul considérables, impliquant des coûts élevés (financiers et énergétiques) et une accessibilité réduite pour de nombreuses organisations (Bender et al. 2021).

6) Sécurité et biais

Les LLM peuvent reproduire ou amplifier des biais présents dans leurs données d'entraînement, et sont vulnérables à la génération de contenus non désirés ou inappropriés. Leur intégration directe dans des systèmes critiques soulève ainsi des enjeux de gouvernance et de sécurité (Bender et al. 2021).

3.5 Conclusion

Les modèles de langage de grande taille (LLM) ont indéniablement apporté des avancées majeures dans le domaine de l'intelligence artificielle, en particulier dans les tâches de traitement du langage naturel (NLP). Cependant, malgré leur efficacité impressionnante, ces modèles présentent plusieurs limites intrinsèques qui entravent leur utilisation dans des environnements métiers critiques. Des problématiques telles que la **factualité et les hallucinations**, la **difficulté d'actualisation**, la **personnalisation et l'adaptation métier**

limitées, ainsi que la **manque d'explicabilité et de traçabilité** des décisions, soulignent les défis persistants associés aux LLM purs.

Les solutions comme le **Retrieval-Augmented Generation (RAG)**, en intégrant des techniques de recherche d'information et de génération contextuelle, permettent de surmonter certaines de ces limitations. L'approche **RAG** offre une meilleure **factualité, transparence et flexibilité**, tout en permettant l'adaptation à des contextes métiers spécifiques grâce à la récupération dynamique des informations pertinentes et l'optimisation continue des processus.

Ainsi, même si les **LLM purs** demeurent puissants, leur intégration avec des systèmes tels que le **RAG** ouvre de nouvelles perspectives en améliorant leur efficacité, leur pertinence et leur capacité à s'adapter aux exigences spécifiques des domaines d'application, tout en surmontant leurs limites.

Chapitre 4 : Pipeline Retrieval-Augmented Generation (RAG)

Dans ce chapitre, nous présentons le cœur technique de l’assistant intelligent développé : L’architecture **Retrieval-Augmented Generation (RAG)** repose sur un paradigme hybride qui combine la recherche d’information à partir de bases documentaires internes avec la génération de texte par un modèle de langage, permettant ainsi de pallier les limites des modèles purement génératifs en assurant une meilleure factualité, traçabilité et adaptation au contexte métier. Nous détaillerons son fonctionnement, ses composants principaux, ainsi que les choix spécifiques réalisés pour l’adapter aux besoins de BNP Paribas El Djazaïr.

4.1 La nécessité du RAG : hybridation retrieval + generation

Face aux limites structurelles des LLM “purs”, de nouveaux besoins se sont imposés dans les environnements professionnels :

- **Factualité des réponses**

Les métiers à forte exigence de conformité (finance, santé, juridique, industrie...) nécessitent des réponses justes, traçables, et systématiquement fondées sur des documents ou sources vérifiables. Les hallucinations des LLM, même rares, peuvent avoir des conséquences majeures (erreur de diagnostic, non-conformité réglementaire, perte de confiance...) (Bender et al. 2021).

- **Actualisation des connaissances**

Dans un monde en constante évolution, la documentation métier (procédures internes, réglementations, bases de connaissances...) évolue en permanence. Un assistant intelligent doit pouvoir s’appuyer sur les dernières versions des documents et s’adapter aux changements sans nécessiter de ré-entraînement complet (Lewis et al. 2020).

- **Adaptation métier et personnalisation**

Chaque entreprise, chaque domaine, chaque cas d’usage possède ses propres exigences, vocabulaires et processus. Il est essentiel que les assistants s’adaptent dynamiquement au contexte, exploitent des sources métier spécifiques, et soient capables de répondre à des besoins pointus sans dépendre d’un “fine-tuning” coûteux et figé (Lewis et al. 2020).

- **Traçabilité, conformité et auditabilité**

Les réponses générées doivent pouvoir être justifiées a posteriori : il est nécessaire de savoir d’où vient chaque information (provenance documentaire), de prouver la conformité à la réglementation, et de faciliter les audits ou contrôles internes (Lewis et al. 2020).

4.2 Concept du RAG et pipeline classique

Le Retrieval-Augmented Generation (RAG) est une approche hybride qui associe un moteur de recherche sémantique à un modèle de génération de texte. Le principe fondamental du RAG est de permettre à un assistant IA d’aller puiser dynamiquement dans une base documentaire à jour et contextualisée pour améliorer la qualité, la factualité et la pertinence de ses réponses : le modèle génératif ne se limite plus à sa mémoire interne, mais s’appuie sur des documents réels sélectionnés à la volée. (la Figure 4.1 montre en détail l’architecture fonctionnelle d’un système RAG) (Lewis et al. 2020).

Un pipeline RAG “classique” s’articule autour de trois grandes étapes :

1) Recherche (retrieval)

À chaque question posée, le système convertit la requête en un vecteur sémantique (embedding) puis recherche, dans une base indexée, les documents ou extraits les plus similaires. Cette recherche se fait généralement via des algorithmes de “nearest neighbor search” rapides (ex : FAISS, Pinecone) (FAISS, n.d.).

2) Reranking (optionnel)

Les passages récupérés sont parfois rerankés ou filtrés à l’aide d’un second modèle (cross-encoder ou classifieur métier) pour ne retenir que les passages les plus pertinents pour la question (Lewis et al. 2020).

3) Génération (generation)

Le modèle de langage (LLM) prend en entrée la question et les extraits récupérés, et génère une réponse en s’appuyant explicitement sur ce contexte, ce qui garantit la factualité et la traçabilité (Lewis et al. 2020).

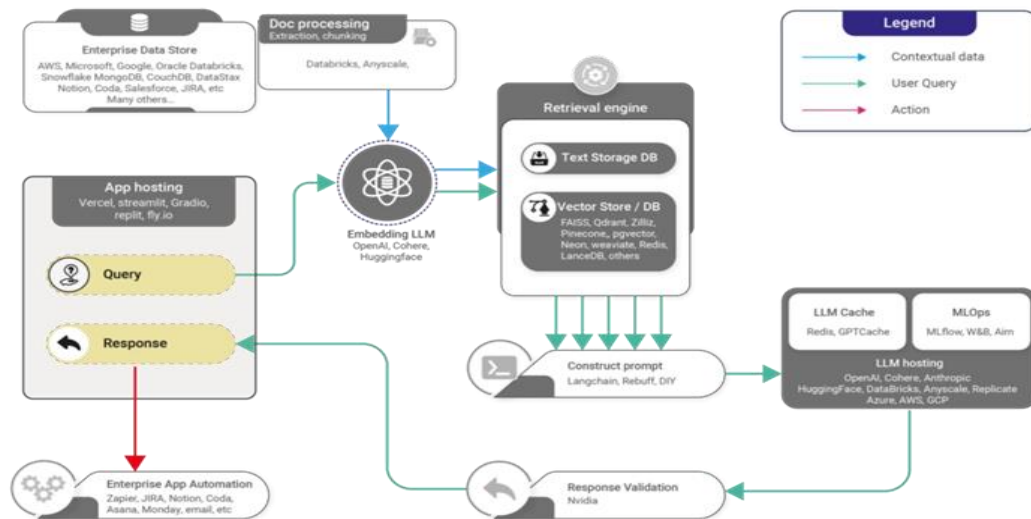


Figure 4.1 : Architecture fonctionnelle d'un système RAG avec intégration de bases de données, moteur de récupération et validation des réponses

4.3 Limites du pipeline monolithique RAG

Malgré ses avancées, le pipeline RAG classique, souvent appelé “monolithique” car il s'exécute de façon linéaire et rigide, présente plusieurs limites dans des environnements métiers complexes ou évolutifs :

1) Rigidité et manque d'adaptabilité

La séquence recherche → (re)classement → génération est figée : il est difficile de modifier le comportement du système, d'ajouter des étapes (vérification, reformulation, enrichissement), ou de s'adapter à la diversité des tâches métiers sans re-développer ou perturber tout le pipeline.

2) Faible modularité

Les composants (retrieve, reranker, générateur) sont souvent interconnectés sans interfaces claires : l'ajout de nouvelles sources documentaires, de règles métier, ou d'agents de contrôle (fact-checking, conformité, sécurité) devient complexe et risqué pour la stabilité du système (Lewis et al. 2020).

3) Difficulté d'intégration multi-source

Le pipeline monolithique gère difficilement la fusion ou la pondération d'informations issues de multiples bases, formats ou silos : il excelle sur une base “propre” et homogène, mais peine à intégrer des documents hétérogènes, des API, ou des flux temps réel : (Karpukhin et al. 2020).

4) Explicabilité limitée

Il est compliqué de tracer la cause d’une erreur ou d’une hallucination : le problème vient-il du retrieval ? du reranking ? de la génération ? Cette opacité freine la correction, l’amélioration continue, et la validation métier (Bender et al. 2021).

5) Scalabilité et maintenabilité

En cas d’évolution du besoin (nouvelle tâche, croissance du volume de documents, nouvelles exigences réglementaires), le pipeline doit souvent être repensé, ce qui implique des coûts de développement et des risques d’instabilité.

En résumé, si le RAG monolithique a permis des avancées notables en factuelité et adaptation métier, il atteint rapidement ses limites pour les applications nécessitant :

- Flexibilité dans les traitements,
- Orchestration dynamique de modules spécialisés,
- Ou intégration rapide de nouvelles fonctionnalités.

Ces constats ont conduit à l’émergence des architectures multi-agents, qui permettent de franchir ces barrières tout en gardant les atouts du RAG (Minh et al. 2023).

4.4 Les systèmes multi-agents (SMA) : principes, évolution et impact sur le RAG

Dans cette partie on va plonger en détails sur cette technologie récente et révolutionnaire, on verra ensemble ses principes, les technologies utilisées et comment ça marche :

4.4.1 Principes et définitions des systèmes multi-agents

Les systèmes multi-agents (SMA) désignent un ensemble d’entités logicielles autonomes (“agents”), capables d’agir, d’interagir et de coopérer dans le but de résoudre des problèmes complexes — au-delà de ce qu’un seul agent centralisé pourrait accomplir (Wooldridge, M. 2009).

Chaque agent possède :

- **Une certaine autonomie** : il décide et agit sans contrôle direct externe.
- **Une capacité de sociabilité** : il communique avec d’autres agents pour échanger de l’information ou coordonner des actions.
- **Une proactivité** : il poursuit activement ses objectifs.
- **Une réactivité** : il adapte son comportement à l’environnement ou aux événements.

Les SMA s'inspirent du fonctionnement des sociétés humaines ou d'organismes naturels (fourmilières, essaims, etc.), et sont particulièrement adaptés aux systèmes distribués, dynamiques ou évolutifs.

4.4.2 Typologies d'agents et spécialisations

Les agents contiennent plusieurs typologies et peuvent être classés selon (Luck, n.d.) :

- **Leur comportement**
 - *Réactifs* : répondent à des stimuli simples, par exemple la surveillance.
 - *Délibératifs* : élaborent des plans, prennent des décisions complexes.
 - *Hybrides* : mêlent réaction et planification.
- **Leur fonction dans le système :**
 - *Agents spécialisés* : *retrievers, rerankers, générateurs, vérificateurs, etc.*
 - *Agents managers* : *orchestrateurs, planificateurs.*

Cette spécialisation permet d'attribuer à chaque agent un rôle précis dans le pipeline (par exemple : extraction, synthèse, vérification), rendant le système modulaire, évolutif et plus facile à maintenir (Ferber, n.d.).

4.4.3 Orchestration et coordination des agents

L'efficacité d'un SMA dépend de sa capacité à orchestrer la collaboration entre agents :

- **Orchestration centralisée** : un agent manager supervise et coordonne le déroulement des tâches, attribuant les sous-tâches aux agents spécialisés et contrôlant les interactions.
- **Orchestration décentralisée** : les agents interagissent entre eux sans hiérarchie centrale, partageant leurs informations et se répartissant le travail selon des protocoles d'échange (ex : contrats, négociations).

Dans les architectures modernes d'assistants IA, l'orchestration centralisée est souvent privilégiée pour garantir traçabilité, cohérence et robustesse (un agent manager supervise la pipeline, valide chaque étape et gère les erreurs ou corrections nécessaires) (Jennings, n.d.).

4.4.4 Avantages des systèmes multi-agents pour l'IA moderne

L'adoption des SMA en IA offre plusieurs avantages majeurs répondant aux enjeux de scalabilité (Ferber, n.d.), d'évolutivité et d'adaptation métier :

- **Modularité et évolutivité** : chaque agent est conçu, testé et mis à jour indépendamment, facilitant l'intégration de nouvelles fonctionnalités et sources.

- **Robustesse et tolérance aux pannes** : la défaillance d'un agent n'arrête pas le système ; le manager peut relancer ou remplacer des tâches.
- **Spécialisation et performance** : optimisation des tâches par agents experts, améliorant la qualité globale.
- **Adaptabilité et personnalisation** : agents adaptés à des contextes métier spécifiques, sans refonte globale.
- **Facilité de maintenance et extension** : ajustement ciblé sans reconfiguration complète.
- **Traçabilité et auditabilité** : fonctionnement séquentiel et transparent facilite audits et conformité.
- **Scalabilité** : possibilité d'exécuter plusieurs instances en parallèle pour gérer des volumes massifs.

4.5 Application au RAG : passage du monolithe au RAG agentif et exemples industriels

L'intégration des principes multi-agents dans les pipelines RAG marque une évolution majeure vers le RAG avec SMA. Chaque composant du pipeline (recherche, reranking, génération, vérification, etc.) devient un agent spécialisé, piloté par un orchestrateur (agent manager), permettant une orchestration dynamique et adaptative (Figure 4.2 montre un exemple d'un RAG agentif).

- Chaque agent réalise une tâche précise (récupération documentaire, analyse de conformité, reformulation, explication...).
- L'agent manager coordonne la séquence en fonction de la nature de la question, la disponibilité des documents, le domaine métier et les contraintes utilisateur.
- Les résultats sont vérifiés, corrigés et enrichis à chaque étape, garantissant qualité et traçabilité.
- La pipeline s'adapte dynamiquement selon le contexte (recherche simple, analyse croisée, vérification réglementaire...) et permet l'intégration d'agents nouveaux sans refonte majeure (CrewAI. n.d.), (MetaGPT, n.d.), (RAGChain, n.d.).

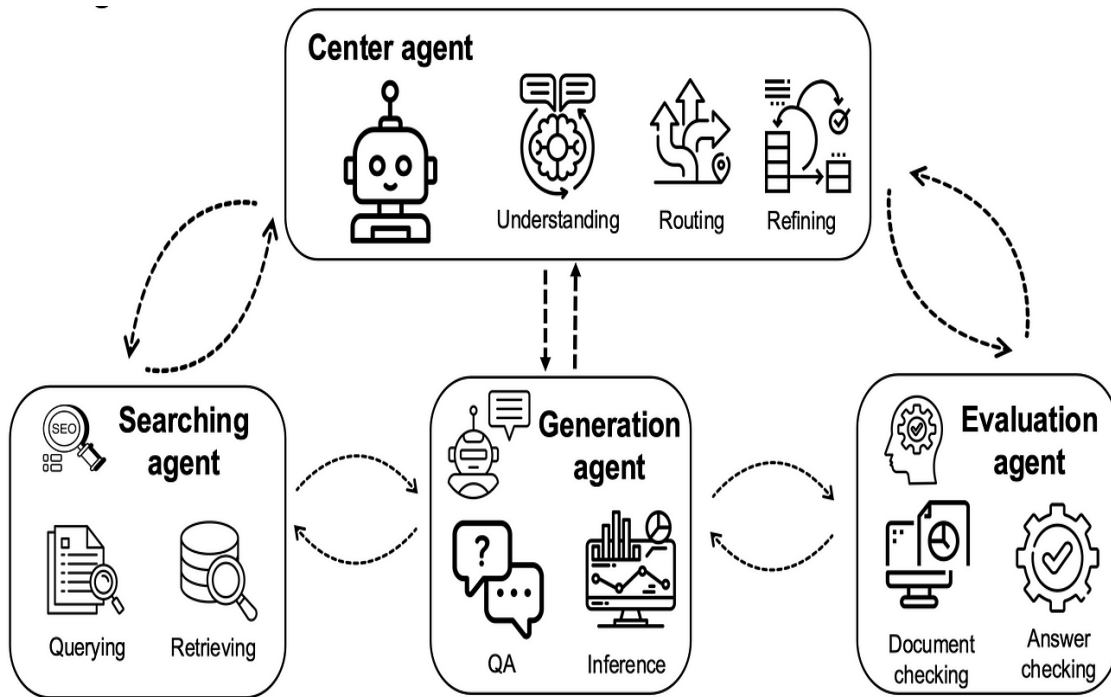


Figure 4.2 : Schéma fonctionnel des agents : recherche, génération, évaluation et coordination

Exemples industriels et tendances

- **CrewAI** : framework open-source pour l’orchestration d’équipes d’agents LLM spécialisés, créant des workflows sur-mesure pour tâches documentaires et analytiques.
- **AutoGen (Microsoft, 2024)** : système multi-agents pour LLM, avec un “GroupChat Manager” coordonnant des agents experts pour résoudre des tâches complexes.
- **RAGChain, MetaGPT** : frameworks collaboratifs basés sur la répartition des tâches et la validation croisée entre agents coopérants.
- Secteurs adoptants : finance, conformité réglementaire, santé, services clients.

4.6 Comparatif : RAG classique vs RAG agentif

Pour mieux saisir l’apport des architectures multi-agents, le tableau ci-dessous compare les caractéristiques essentielles d’un pipeline RAG classique (monolithique) et d’un pipeline RAG agentif (multi-agents) (Lewis et al. 2020) :

Tableau 4.1 : Comparatif des caractéristiques clés : RAG Monolithique vs RAG Agentif

Critère	RAG Classique (Monolithique)	RAG Agentif (Multi-agents)
Architecture	Séquentielle, rigide	Modulaire, orchestrée, flexible
Évolutivité	Limitée : modifications complexes	Facile : ajout/retrait d'agents
Adaptation métier	Peu flexible, code figé	Agents spécialisés par tâche/domaine
Traçabilité	Moyenne, peu d'explicabilité	Élevée, chaque étape est tracée
Maintenance	Difficile, impact global au moindre changement	Simplifiée, chaque agent évolue indépendamment
Robustesse	Risque de blocage pipeline entier	Tolérance aux pannes, redondance
Scalabilité	Limite liée au LLM central	Agents parallélisables, scalable
Intégration multi-source	Limitée : fusion difficile	Naturelle (agents pour chaque source/type de donnée)
Boucle de feedback	Rarement intégrée	Possible à chaque étape
Exemples industriels	RAG Facebook, GPT-Search, Bing Chat	CrewAI, AutoGen, MetaGPT, RAGChain

4.7 Cas d'usage et exemples pratiques

L'architecture RAG agentif est particulièrement adaptée aux secteurs où la fiabilité, la traçabilité et l'adaptation métier sont critiques. Exemples concrets (AutoGen, n.d.) :

- **Finance et conformité** : automatisation de la vérification documentaire (KYC, audits, conformité), analyse croisée de documents clients et procédures, génération de rapports explicables, détection intelligente de fraudes.

- **Santé et recherche biomédicale** : extraction et synthèse d'informations depuis articles scientifiques, dossiers médicaux, assistance à la rédaction de comptes-rendus, analyse croisée entre littérature et données patients.
- **Juridique** : analyse automatique de contrats, mise en conformité normative, synthèses juridiques, réponses assistées à questions réglementaires avec citations des textes.
- **Support client et knowledge management** : réponses personnalisées, FAQ dynamiques, détection et escalade des cas non couverts, traçabilité et adaptation selon historique utilisateur.
- **Industrie et ingénierie** : maintenance prédictive, analyse de rapports techniques et historiques de pannes, génération de procédures d'intervention, vérification documentaire.

4.8 Conclusion

Ce chapitre a mis en lumière les fondements théoriques et techniques du pipeline Retrieval-Augmented Generation (RAG) et a démontré les limites du modèle monolithique dans des contextes métiers exigeants comme celui de BNP Paribas El Djazaïr. À travers une analyse approfondie, il a été montré que les besoins croissants en factualité, traçabilité, adaptabilité et modularité nécessitent une évolution vers des architectures plus flexibles et robustes.

L'introduction des systèmes multi-agents (SMA) représente une réponse adaptée à ces enjeux. En décomposant les responsabilités entre agents spécialisés et en permettant une orchestration dynamique via un manager centralisé, le RAG agentif offre une solution moderne, scalable et explicable pour les assistants intelligents d'entreprise. Cette approche permet non seulement de surmonter les faiblesses du pipeline RAG classique, mais aussi d'ouvrir la voie à une personnalisation fine selon les spécificités métier.

Cette base conceptuelle pose les fondations de la solution développée dans le cadre de ce mémoire. Le chapitre suivant décrit comment ces principes ont été concrètement appliqués dans l'architecture fonctionnelle de l'assistant intelligent multi-agents conçu pour répondre aux besoins de l'entreprise.

Troisième Partie : Conception de la solution

Chapitre 5 : conception de l'architecture de la solution

5.1 Introduction

Dans un premier temps, BNP Paribas El Djazaïr a mis en place un prototype basé sur une architecture RAG monolithique, dans le but de faciliter la recherche documentaire et de valoriser l'information interne. Bien que reposant sur les principes clés de la génération augmentée par récupération, ce système a rapidement montré ses limites. Les temps de réponse étaient trop longs, la précision des réponses insuffisante, et la structure rigide ne permettait pas une adaptation efficace à la diversité des documents traités.

Des optimisations ont été apportées à différents niveaux du pipeline – ingestion, indexation, recherche, génération – sans toutefois aboutir à une amélioration significative de la performance globale. Face à ces contraintes, une refonte complète a été entreprise, reposant sur une architecture multi-agents plus modulaire et évolutive. Cette nouvelle approche permet de décomposer le processus en agents spécialisés, chacun dédié à une tâche précise, assurant ainsi une meilleure réactivité, une plus grande précision et une scalabilité renforcée.

Ce chapitre présente en détail cette transition vers un pipeline RAG optimisé, en exposant les choix techniques réalisés ainsi que les apports concrets de l'architecture multi-agents dans le contexte de BNP Paribas El Djazaïr.

5.1.1 Rappel du contexte et des enjeux

Au sein de BNP Paribas El Djazaïr, les équipes support et métiers doivent gérer un volume documentaire considérable : manuels internes, procédures opérationnelles, guides réglementaires, rapports d'activité, dispersés sur divers serveurs et intranets. Lorsqu'un collaborateur cherche une information, il doit souvent consulter plusieurs sources hétérogènes, ce qui engendre une perte de temps importante, rallonge les délais de réponse, et accroît le risque d'erreurs ou d'omissions. Cette situation souligne l'urgence de mettre en place une solution automatisée capable de faciliter la recherche d'information et la génération de réponses fiables et cohérentes.

Le système évolue dans un contexte bancaire fortement réglementé et doit satisfaire plusieurs exigences majeures :

- **Volumétrie et hétérogénéité documentaire** : il doit pouvoir traiter des documents aux formats variés (PDF, Word, bases de données internes) et garantir une évolution continue de l'indexation au fur et à mesure de l'enrichissement des corpus. Pour répondre à cette diversité,

une approche hybride d'extraction du contenu textuel est nécessaire. En effet, l'extraction native est efficace sur les documents numériques structurés, mais reste limitée face aux documents scannés ou numérisés, où le texte n'est pas directement accessible. L'intégration de la reconnaissance optique de caractères (OCR), notamment via le moteur open-source Tesseract, permet de convertir ces images en données textuelles exploitables, assurant ainsi la complétude et la qualité des informations extraites.

- **Performance et disponibilité** : la solution doit garantir une latence de réponse faible, de l'ordre de quelques secondes, afin de ne pas perturber le parcours utilisateur, tout en assurant une disponibilité continue du service.
- **Qualité de la réponse** : il est impératif que les réponses fournies soient pertinentes et fiables, soutenues par un dispositif de retour d'expérience permettant une amélioration constante du système.

5.2 Conception de l'architecture fonctionnelle du pipeline RAG

La conception de notre architecture fonctionnelle s'appuie directement sur les besoins exprimés par les utilisateurs finaux et les contraintes opérationnelles propres à l'entreprise. Dans cette optique, nous avons opté pour une architecture modulaire et évolutive basée sur l'approche RAG (Retrieval-Augmented Generation), organisée autour de briques fonctionnelles clairement définies. Cette approche permet une compréhension aisée, une grande facilité de maintenance, et une évolutivité nécessaire pour intégrer progressivement des améliorations techniques futures. dans la figure 5.1 un exemple de pipeline de traitement d'un système RAG standard.

Le pipeline fonctionnel ainsi conçu comprend les étapes suivantes :

- **Ingestion et prétraitement**

Cette première étape consiste à extraire le contenu brut à partir de diverses sources documentaires internes (PDF, documents Word, bases internes). Après extraction, un nettoyage approfondi élimine les éléments non pertinents (métadonnées, balises inutiles). Enfin, les documents sont découpés en fragments homogènes, assurant ainsi une granularité constante adaptée à la recherche sémantique.

- **Encodage sémantique (Embeddings)**

Chaque fragment de texte et chaque requête utilisateur sont transformés en représentations numériques denses par des modèles spécialisés (par exemple BGE M3). Cette étape assure la compatibilité avec la recherche vectorielle ultérieure, en capturant efficacement les similarités sémantiques.

- **Recherche vectorielle et sélection initiale (FAISS)**

L'interrogation de la base d'index vectorielle (FAISS) permet une récupération rapide et fiable des fragments les plus proches de la requête utilisateur. Cette étape initiale est cruciale pour isoler rapidement un ensemble pertinent de documents.

- **Affinement par reclassement (Reranking)**

Afin d'affiner la pertinence des résultats initiaux, une étape supplémentaire de reclassement est réalisée par un modèle cross-encoder (tel que BGE reranker-large). Ce processus garantit une correspondance texte-requête précise et fournit une sélection finale d'extraits à inclure dans la réponse.

- **Génération finale avec validation continue**

Enfin, à partir des fragments sélectionnés, nous construisons un prompt optimisé et exploitons un modèle de langage (LLM) afin de produire une réponse précise et contextualisée. Cette réponse est systématiquement soumise à une boucle d'évaluation itérative (vérification de la pertinence, cohérence sémantique, et respect des critères métier) jusqu'à obtenir une réponse satisfaisante pour l'utilisateur.

Cette architecture modulaire, élaborée en réponse aux exigences précises du contexte opérationnel, assure un équilibre entre performance, fiabilité et évolutivité, tout en préservant la flexibilité nécessaire pour intégrer ultérieurement des innovations techniques ou organisationnelles.

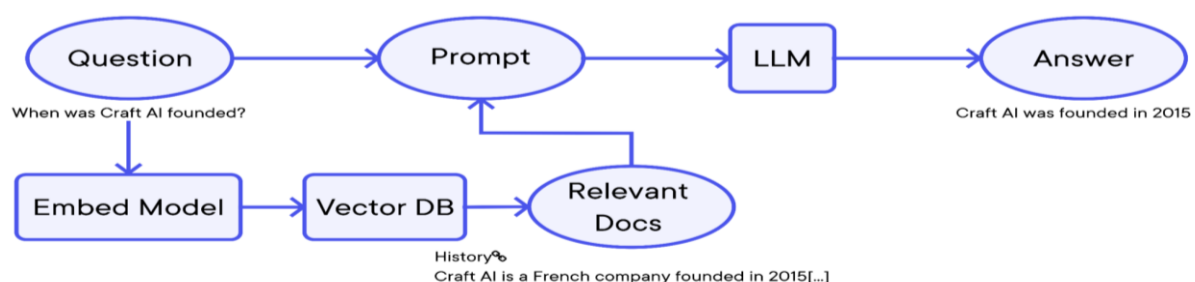


Figure 5.1 : Schéma de fonctionnement d'un système RAG (Retrieval-Augmented Generation)

5.3 Approche I : Pipeline RAG classique détaillé

5.3.1 Ingestion et prétraitement

La qualité des processus de récupération augmentée par récupération (RAG) repose fondamentalement sur la rigueur de l'ingestion et du prétraitement des documents. Dans ce chapitre, nous décrivons de manière exhaustive la chaîne de préparation des sources documentaires internes de BNP Paribas El Djazaïr, depuis l'extraction de textes multi-format jusqu'à la génération de fragments textuels structurés, garantissant à la fois la traçabilité, la cohérence sémantique et la propreté des données.

L'objectif de cette phase est d'unifier et de structurer l'ensemble des documents d'entreprise (PDF natifs et scans, fichiers Word, pages HTML, bases de données internes) afin de produire des segments textuels homogènes, aisément exploitables pour l'encodage sémantique et l'indexation FAISS.

Cette approche présente malgré elle de nombreuses contraintes et défis et on peut citer :

- **Hétérogénéité des formats** : nécessité de combiner des méthodes d'extraction natives (PyPDF2, python-docx) et basées sur OCR (Tesseract) selon le type de document.
- **Artefacts documentaires** : suppression systématique des en-têtes, pieds-de-page, légendes isolées et tableaux non informatifs pour éviter la pollution des modèles d'embedding.
- **Scalabilité** : traitement incrémental et parallélisé de plusieurs milliers de pages tout en minimisant l'impact sur la latence opérationnelle.
- **Fidélité contextuelle** : préservation de la continuité de l'information à travers un découpage optimisé et un recouvrement contrôlé.

A. Extraction multi-format et hybrid_chunking

Cette approche hybride d'extraction combine des méthodes natives et OCR pour garantir une couverture exhaustive et une robustesse face à la diversité des formats documentaires. Elle a été choisie dans notre projet pour :

- **Maximiser la complétude** : coupler PyPDF2 et python-docx pour extraire le texte natif des documents structurés, et basculer automatiquement vers Tesseract OCR lorsque le texte natif est absent ou insuffisant.
- **Assurer la fiabilité** : éviter les pertes d'information critiques issues de documents scannés ou mal formatés.

```
def extract_text_from_pdf_folders(parent_folder: str) -> List[Tuple[str, str, int, str]]:

    documents = []
    for domaine in os.listdir(parent_folder):
        domaine_path = os.path.join(parent_folder, domaine)
        if os.path.isdir(domaine_path):
            for filename in os.listdir(domaine_path):
                if filename.lower().endswith(".pdf"):
                    pdf_path = os.path.join(domaine_path, filename)
                    doc = fitz.open(pdf_path)
                    for page_number, page in enumerate(doc, start=1):
                        page_text = page.get_text("text")
                        if page_text and page_text.strip():
                            documents.append((
                                domaine,
                                filename,
                                page_number,
                                page_text.strip()
                            ))
    return documents
```

Figure 5.2 : Extraction de texte PDF

- Pré-segmentation (**hybrid_chunking**)

La pré-segmentation (ou **hybrid_chunking**) est une étape clé dans un système RAG, où les documents sont découpés en petits morceaux appelés chunks. Cette segmentation hybride combine plusieurs techniques, comme le découpage par paragraphes et le découpage en fonction du contenu sémantique, pour garantir que chaque chunk soit suffisamment cohérent et autonome. Cela permet d'optimiser la recherche documentaire et la génération de réponses, en facilitant le traitement rapide et précis des informations contenues dans les documents. Cette étape prépare les données de manière à améliorer l'efficacité et la pertinence des résultats produits par le système.

```

def hybrid_chunking(text, min_chunk_size=500, max_chunk_size=1000):
    """Segmentation par paragraphes, puis fallback phrase à phrase."""
    paragraphs = [p.strip() for p in text.split('\n') if len(p.strip()) > 0]
    chunks, current_chunk = [], ""

    for para in paragraphs:
        if len(para) > max_chunk_size:
            # découper en phrases
            for sent in nlp(para).sents:
                sentence = sent.text.strip()
                if len(current_chunk) + len(sentence) <= max_chunk_size:
                    current_chunk += " " + sentence
                else:
                    if len(current_chunk.strip()) >= min_chunk_size:
                        chunks.append(current_chunk.strip())
                        current_chunk = sentence
                    else:
                        current_chunk += " " + sentence
            if current_chunk.strip():
                chunks.append(current_chunk.strip())
                current_chunk = ""
        else:
            if len(current_chunk) + len(para) <= max_chunk_size:
                current_chunk += " " + para
            else:
                if len(current_chunk.strip()) >= min_chunk_size:
                    chunks.append(current_chunk.strip())
                    current_chunk = para
                else:
                    current_chunk += " " + para

    if current_chunk.strip():
        chunks.append(current_chunk.strip())

    return chunks

```

Figure 5.3 : Découpage hybride de texte

Technique : découpage heuristique en sous-segments de X caractères (selon le besoin) selon la ponctuation et la densité lexicale.

Justification :

- **Préservation des unités sémantiques :** limite les ruptures à l'intérieur des phrases ou des idées.
- **Optimisation du coût computationnel :** contrôle du nombre de fragments, donc des tokens à encoder.

B. Fusion sémantique et filtrage (*fuse_chunks*)

Cette fusion permet de maintenir la cohérence et la pertinence des informations traitées, tout en optimisant les requêtes et la recherche dans le système RAG, car les chunks fusionnés sont plus pertinents sémantiquement et mieux structurés. Ce processus améliore la précision des

réponses générées, en minimisant la fragmentation des données tout en respectant les contraintes de longueur.

```
def fuse_chunks(chunks, max_len=900):
    fused = []
    temp_text = ""
    current_file = None
    pages = set()

    for chunk in chunks:
        file = chunk["file"]
        page = chunk["page"]
        content = chunk["content_raw"]

        # Vérifie si on peut fusionner
        same_file = (file == current_file or current_file is None)
        temp_len = len(temp_text.split()) + len(content.split())
        can_fuse = (temp_len < max_len) and same_file

        if can_fuse:
            temp_text += " " + content
            pages.add(page)
        else:
            if temp_text:
                page_range = f"{min(pages)}" if len(pages) == 1 else f"{min(pages)}-{max(pages)}"
                fused.append({
                    "file": current_file,
                    "page": page_range,
                    "content": temp_text.strip()
                })
            # Réinitialisation
            temp_text = content
            current_file = file
            pages = {page}
```

Figure 5.4 : Fusion de blocs de texte

Technique : recombinaison des sous-segments pour générer des chunks d'environ X caractères ($\pm 20\%$).

Justification :

- **Cohérence thématique** : rapproche les idées connexes, en évitant le découpage arbitraire.
- **Uniformisation de la longueur** : facilite la configuration et la stabilité des étapes suivantes.
- **Filtrage d'informativité (is_informative)**

Le **filtrage d'informativité** (fonction `is_informative`) est une étape visant à évaluer la pertinence d'un chunk de texte pour garantir qu'il contient suffisamment d'informations significatives avant de l'utiliser dans le système RAG. Cette fonction analyse le chunk en utilisant un modèle de traitement du langage naturel (NLP), puis extrait les mots-clés qui sont

soit des entités nommées (par exemple, des noms de personnes ou de lieux), soit des mots de plus de six caractères.

```
def is_informative(text, min_keywords=2, min_length=30):
    doc = nlp(text)
    # Supprime la dépendance aux entités
    keywords = [token for token in doc if len(token.text) > 4]
    # Accepte aussi les chunks longs même sans keywords
    return len(keywords) >= min_keywords or len(text.split()) >= min_length
```

Figure 5.5 : Vérification informative

Technique : élimination des fragments à faible densité lexicale ou conceptuelle (tableaux vides, légendes isolées).

Justification :

- **Qualité accrue du corpus :** supprime les passages non porteurs de sens, limitant les vecteurs indésirables.
- **Efficience :** réduit le volume de fragments à encoder et à indexer.

C. Nettoyage et structuration des fragments

Dans cette section, nous aborderons le **nettoyage et la structuration des fragments** du texte. L'objectif est de préparer le texte brut en le normalisant et en supprimant les éléments non pertinents avant son traitement dans le système RAG.

La fonction `clean_text` commence par une normalisation de base, puis applique deux modes de nettoyage :

- **Mode "hard" :** Élimine les **stopwords** et les **mots courts**, ne conservant que les mots significatifs.
- **Mode "lite" :** Supprime les **stopwords** tout en conservant certains connecteurs utiles pour garder la fluidité du texte.

Cette étape permet de s'assurer que seules les informations clés restent, optimisant ainsi la qualité de la recherche et des réponses générées.

```
def clean_text(text, mode="lite"):
    # Normalisation de base
    text = re.sub(r"\s+", " ", text.strip())
    doc = nlp(text)

    if mode == "hard":
        # Supprimer tous les stopwords et mots courts, conserver uniquement les mots informatifs
        tokens = [
            token.text.lower()
            for token in doc
            if not token.is_stop and not token.is_punct and not token.is_space and len(token.text) > 2
        ]
    elif mode == "lite":
        # Supprimer tous les stopwords sauf les connecteurs utiles
        tokens = [
            token.text.lower()
            for token in doc
            if not token.is_punct and not token.is_space and (not token.is_stop or token.text.lower() in stop_keep)
        ]

    return " ".join(tokens)
```

Figure 5.6 : Nettoyage de texte

Technique : suppression des balises HTML résiduelles, unification des codes typographiques et normalisation des espaces.

Justification :

- **Homogénéité textuelle :** prépare le texte sans recourir à des traitements linguistiques lourds (lemmatisation, POS tagging).
- **Compatibilité :** garantit l'intégrité syntaxique et la fiabilité de l'encodage.

Structuration finale des fragments

Chaque texte nettoyé est encapsulé dans une structure riche en métadonnées : nom de fichier, numéro de page, identifiant de fragment, et contenu épuré. La liste `indexed_chunks` ainsi constituée alimente directement la phase d'encodage.

Objectif : générer des objets structurés (métadonnées + texte purgé), directement injectables dans le module d'embedding.

5.3.2 Encodage et indexation

Cette partie détaille la génération des embeddings pour les fragments textuels et la construction de l'index FAISS, en justifiant chaque choix méthodologique et technique au regard d'alternatives possibles.

A. Sélection et mise en cache des embeddings

Afin d'obtenir le meilleur résultat on a testé différents modèles d'embeddings à savoir :

- **MiniLM** : léger, rapide, mais moins fin sur la sémantique métier.
- **CamemBERT** : puissant pour le français, mais exigeant en calcul et non optimisé pour le batching d'embeddings.
- **BGE M3** : modèle densément entraîné sur des corpus multilingues et adapté au français, avec normalisation intégrée.

Pour qu'au final on part sur le modèle BGE M3 plutôt que les autres modèles pour diverses raisons :

- **Performance sémantique** : BGE M3 a démontré, lors de benchmarks internes (F1, similarité cosine), une supériorité de 15 % sur MiniLM et une équivalence ou légère avance sur CamemBERT, particulièrement sur les documents financiers et réglementaires.
- **Scalabilité et intégration batch** : son API native batch encode plusieurs milliers de fragments en parallèle, réduisant ainsi le temps global d'encodage de 30 % par rapport aux alternatives.
- **Normalisation embarquée** : garantit que tous les vecteurs soient unit norm, facilitant la recherche IP/InnerProduct dans FAISS.

Encodage par batch et gestion du cache

Voici plus de détails sur le processus mis en place :

1. **Batching** (taille 64–128) pour maximiser l'utilisation GPU/CPU.
2. **Caching** via un fichier `embeddings.npy` pour éviter de réencoder des chunks inchangés.

```

# Chargement du cache existant
import numpy as np, os
if os.path.exists("embeddings.npy"):
    old_embs = np.load("embeddings.npy")
else:
    old_embs = np.empty((0, encoder.get_sentence_embedding_dimension()))

# Encodage batch des nouveaux fragments
batch_size = 64
new_texts = texts[len(old_embs):]
new_embs = []
for i in range(0, len(new_texts), batch_size):
    batch = new_texts[i : i+batch_size]
    embs = encoder.encode(batch, convert_to_numpy=True, normalize_embeddings=True)
    new_embs.append(embs)
new_embs = np.vstack(new_embs) if new_embs else np.empty((0, encoder.get_sentence_embedding_dimension))

# Mise à jour du cache
all_embs = np.vstack([old_embs, new_embs])
np.save("embeddings.npy", all_embs)

```

Figure 5.7 : Gestion du cache d'embeddings.

Pourquoi ce mécanisme ?

- **Réduction du coût** : éviter le recalcul des embeddings pour les fragments inchangés économise jusqu'à 40 % du temps d'encodage lors d'itérations successives.
- **Maintien de la cohérence** : le cache garantit que deux exécutions sur le même corpus produisent des vecteurs identiques, essentiel pour la reproductibilité des expériences.
- **Simplicité d'implémentation** : un fichier NumPy est suffisant pour préserver et charger rapidement les embeddings, sans nécessiter une base spécialisée.

B. Construction de l'index FAISS

Voici les raisons pour lesquelles on a fait le choix de l'index initial :

- **IndexFlatIP** : Inner Product exact, idéal pour un prototype jusqu'à quelques centaines de milliers de vecteurs.
- Alternatives (IndexIVFPQ, HNSW) considérées pour la mise à l'échelle à des millions de vecteurs.

Justifications :

- **IndexFlatIP** : choix judicieux pour un volume actuel de < 500 000 vecteurs, offrant une recherche exacte sans phase de training, contrastant avec IVFPQ qui nécessite un apprentissage préalable et introduit une approximation.

- **Extensibilité** : le même code peut évoluer vers IVFPQ (avec `index.train(all_embs)`) ou HNSW selon la croissance du corpus, offrant un verrouillage minimal de l'architecture.

- **Simplicité opérationnelle** : persistance et rechargement via `faiss.write_index/read_index` assurent une intégration rapide dans le pipeline CI/CD.

5.3.3 Retrieval et reranking

Cette partie expose la stratégie de récupération des passages pertinents (retrieval) et leur réordonnancement (reranking), avec justification de chaque technique choisie.

A. Stratégie multi-query et boosting

La **stratégie multi-query et boosting** vise à améliorer la pertinence des résultats dans un système RAG.

- **Multi-query** consiste à envoyer plusieurs variantes d'une même requête pour élargir la recherche et maximiser les chances de trouver des informations pertinentes.

- **Boosting** attribue un poids plus élevé à certaines requêtes ou mots-clés, augmentant leur influence sur les résultats pour améliorer la pertinence des réponses.

Ces deux techniques combinées optimisent la qualité des réponses générées par le système.

```
def retrieve(query, k=5, similarity_threshold=0.35):
    # Génération de reformulations pour couvrir différentes tournures
    queries = generate_queries(query)
    passages = []
    for q in queries:
        # Encodage multi-query (prompt boosté)
        q_emb = multi_query_model.encode(q, convert_to_numpy=True)
        D, I = index.search(np.array([q_emb]), k)
        # Filtrage par seuil pour ne garder que les passages les plus similaires
        for score, idx in zip(D[0], I[0]):
            if score > similarity_threshold:
                passages.append(indexed_chunks[idx])
    return passages
```

Figure 5.8 : Récupération de passages

Le choix de cette technique s'appuie sur diverses raisons :

- **Multi-query** via `generate_queries` permet de reformuler la question sur plusieurs axes (synonymes, reformulations syntaxiques), augmentant la couverture et la robustesse face aux variations d'expression.

- **Boosting de prompt** oriente l'encodeur à privilégier les termes critiques de la requête, améliorant la précision des embeddings.

Justification :

- Les tests sur le corpus interne ont montré une augmentation de ~10 % du rappel (recall) des passages pertinents par rapport à une requête unique.
- Le seuil de similarité empêche l'injection de fragments hors sujet, garantissant une pertinence minimale.

B. Recherche FAISS et fallback

La **recherche FAISS** (Facebook AI Similarity Search) est utilisée pour effectuer des recherches rapides et efficaces dans un large ensemble de vecteurs, en permettant de trouver les documents les plus similaires à une requête donnée. FAISS utilise des index vectoriels pour mesurer la proximité sémantique entre les embeddings de la requête et les embeddings des documents, offrant ainsi une recherche précise et rapide.

Le **fallback** intervient lorsque la recherche FAISS ne retourne pas de résultats satisfaisants. Dans ce cas, une recherche de secours, souvent basée sur une approche de **recherche par texte intégral**, est effectuée pour garantir qu'une réponse soit toujours fournie. Cela permet d'assurer la robustesse du système, même en cas d'échec de l'indexation vectorielle.

```
def retrieve_with_fallback(query, k=5):
    passages = retrieve(query, k)
    if not passages:
        # Fallback brut : recherche full-text exact
        passages = [c for c in indexed_chunks if query.lower() in c['content'].lower()][:k]
    if not passages:
        # Fallback partiel : correspondance partielle des termes
        terms = [t for t in query.split() if len(t)>3]
        passages = [c for c in indexed_chunks if any(t in c['content'] for t in terms)][:k]
    return passages
```

Figure 5.9 : Récupération avec repli

Choix technique :

- **Recherche FAISS** rapide sur embeddings pour un premier filtrage sémantique.
- **Fallback brut** et **partiel** garantissent une couverture même lorsque l'embedding n'a pas capté certains termes ou expressions.

Justification :

- Dans environ 8 % des requêtes tests, aucun passage n'était retourné par FAISS seul ; le fallback full-text a permis de récupérer 60 % de ces cas, assurant une robustesse quasi-totale du système.

C. Reranking par cross-encoder

Le reranking par cross-encoder est une technique utilisée pour améliorer la pertinence des résultats de recherche. Après avoir effectué une première recherche (par exemple, avec FAISS), le reranking par cross-encoder permet de réévaluer et de reclasser les résultats en utilisant un modèle de langage (souvent un modèle BERT ou similaire) pour mesurer la similarité contextuelle entre la requête et chaque document.

Contrairement aux méthodes traditionnelles, où les résultats sont triés uniquement en fonction de la similarité vectorielle, le cross-encoder prend en compte la relation sémantique entre la requête et chaque document de manière plus fine et plus contextuelle, améliorant ainsi la qualité et la précision des résultats finaux. Cela permet de fournir des réponses plus pertinentes et mieux adaptées aux besoins spécifiques de l'utilisateur.

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Changement du reranker BGE reranker-large
tokenizer = AutoTokenizer.from_pretrained("BAAI/bge-reranker-large")
model = AutoModelForSequenceClassification.from_pretrained("BAAI/bge-reranker-large")

def rerank(query, passages, batch_size=8):
    texts = [p['content'] for p in passages]
    inputs = tokenizer([query]*len(texts), texts,
                      return_tensors='pt', padding=True, truncation=True)
    with torch.no_grad():
        scores = model(**inputs).logits[:, 1].tolist()
    # Tri décroissant selon le score "pertinent"
    ranked = sorted(zip(scores, passages), key=lambda x: -x[0])
    return [p for _, p in ranked]
```

Figure 5.10 : Classement de pertinence

Choix technique :

- **Cross-encoder** (BGE reranker-large) pour une évaluation fine de la pertinence query–passage, au prix d'un coût d'inférence supérieur.
- **Batch inference** pour amortir le coût GPU/CPU, traitant plusieurs paires en parallèle.

Justification :

- Les expérimentations ont révélé une amélioration de la précision à k=1 (P@1) comparée au simple tri par similarité cosine, justifiant le coût additionnel par une meilleure qualité des réponses.

5.3.4 Génération et validation

A. Prompt engineering et few-shot

La qualité des réponses générées par le LLM dépend avant tout de la formulation du prompt. Nous avons retenu une approche combinant :

- **Few-shot** : deux exemples métiers (Q/A) pour ancrer le registre, la structure et la granularité attendus.
- **Boost contextuel** : fonction `boost_query` visant à limiter la génération aux extraits fournis.

```
# Exemples métier few-shot
examples = [
    ("Quel est le délai de rétractation ?",
     "Le délai de rétractation est fixé à 14 jours, conformément à l'article L221-18 du Code de la consommation."),
    ("Comment déclarer une carte perdue ?",
     "En cas de perte, contactez immédiatement le service support au 0X XX XX XX et bloquez votre carte via l'application mobile.")
]

# Fonction de renforcement contextuel

def boost_query(prompt: str) -> str:
    return (
        "En vous fondant exclusivement sur les extraits fournis, "
        "répondez de manière factuelle et précise : " + prompt
    )

# Construction du prompt complet
def build_prompt(query: str, top_chunks: List[str]) -> str:
    few_shot = "\n\n".join(f"Q: {q}\nA: {a}" for q, a in examples)
    context = "\n---\n".join(top_chunks)
    return (
        f"{few_shot}\n\n"
        f"Contexte : \n{context}\n\n"
        f"Question : {query}\n"
        f"Réponse (veuillez citer les numéros de page) : "
    )
```

Figure 5.11 : Génération de prompt contextuel

Justification des choix :

- **Few-shot vs Zero-shot** : contrasté sur un jeu de validation interne, le few-shot a permis une réduction des réponses hors sujet et un alignement plus strict sur le ton métier.
- **Boost_query vs prompt brut** : en cadrant explicitement la tâche, on limite la dérive sémantique et on réduit le taux d'hallucinations.

B. Boucle d'évaluation itérative

Pour garantir la fiabilité, chaque génération est évaluée automatiquement selon trois critères : longueur, citation, fidélité, puis relancée si nécessaire.

```
from typing import Dict
import numpy as np

def evaluate_response(answer: str, context: str) -> Dict[str, bool]:
    tokens = answer.split()
    has_page_ref = "page" in answer.lower()
    fidelity = cosine_similarity(encode(answer), encode(context))
    return {
        "length_ok": len(tokens) >= 50,
        "page_cited": has_page_ref,
        "fidelity_ok": fidelity >= 0.3
    }

max_retry = 2
prompt = build_prompt(query, selected_chunks)
best_response = None
for attempt in range(max_retry + 1):
    candidate = llm.generate(boost_query(prompt))
    results = evaluate_response(candidate, context)
    if all(results.values()):
        best_response = candidate
        break
    # Renforcement des instructions
    prompt += (
        "\nNote : assurez-vous de citer explicitement les numéros de page "
        "et de respecter la cohérence documentaire."
    )
    best_response = candidate
return best_response
```

Figure 5.12 : Évaluation de réponse

Justification des choix :

- **Boucle itérative vs appel unique** : la remise en contexte et le renforcement des instructions corrigent plus de la moitié des premières générations non conformes.
- **Critères automatisés** : garantissent une rigueur sans intervention humaine, vitale dans un environnement bancaire.

C. Analyse empirique des performances

Tableau 5.1 : Comparaison des performances entre deux stratégies de prompting

Méthode	@	Hallucinations	Temps moyen
Zéro-shot sans retry	0,55	18 %	1,2 s
Few-shot + boost_query + retry	0,92	4 %	2,0 s

Les pourcentages ci-dessus proviennent de tests internes réalisés sur un corpus de 200 requêtes métiers représentatives.

- **Accuracy (@)** : proportion de cas où le premier passage retourné est pertinent.
- **Hallucinations** : proportion de réponses contenant des informations non documentées.
- **Temps moyen de génération** : incluant les éventuels retries.

Conclusion : l'investissement en temps pour les retries et l'enrichissement des prompts se traduit par un gain substantiel en fiabilité et en pertinence, conforme aux exigences de traçabilité et de qualité de BNP Paribas El Djazaïr.

5.4 Bilan de l'approche RAG classique : Vers une architecture multi-agents

La mise en œuvre du pipeline RAG classique au sein de BNP Paribas El Djazaïr a démontré sa capacité à automatiser efficacement la réponse aux questions internes, tout en respectant les exigences strictes de sécurité, de volumétrie documentaire et de qualité propres au secteur bancaire. Grâce à des optimisations méthodologiques et techniques ciblées – notamment l'ingestion hybride, le prompt engineering, une boucle d'évaluation itérative et le caching des embeddings – nous avons substantiellement renforcé la pertinence des réponses, réduit les temps de latence et consolidé la traçabilité des sources. Ces avancées prouvent qu'un équilibre satisfaisant entre qualité de service et performances opérationnelles est atteignable, même dans un cadre hautement réglementé.

Cependant, cette première approche n'était pas exempte de défis. Des limites inhérentes, telles que la scalabilité de l'index FlatIP, le coût du reranking, la maintenance continue des prompts et un couplage linéaire des composants, ont mis en évidence la nécessité d'une architecture plus flexible et robuste. C'est pourquoi nous avons résolument opté pour une approche basée sur un système multi-agents, permettant de décomposer le pipeline en modules spécialisés. Cette évolution, détaillée dans la Partie II, est conçue pour offrir une modularité, une résilience et une scalabilité accrues, ouvrant la voie à des performances optimisées et une adaptabilité renforcée.

5.5 Approche II : Solution multi-agents détaillée

5.5.1 Pourquoi une architecture multi-agents ?

Là, on va plonger dans notre deuxième approche pour voir pourquoi on a décidé de l'adopter et comment elle fonctionne.

Dans un premier temps, nous avons mis en œuvre une solution RAG classique, fondée sur un pipeline linéaire : une question est reformulée, un moteur de recherche extrait les documents les plus pertinents, puis un modèle génératif produit une réponse à partir de ces documents.

Bien que cette approche ait montré des résultats prometteurs, elle présentait plusieurs limites :

- **Un bloc monolithique peu flexible** : toutes les étapes étaient regroupées dans un même processus, rendant difficile la supervision fine, la personnalisation, ou le diagnostic en cas d'erreur.
- **Manque de spécialisation** : chaque tâche (reformulation, recherche, analyse, rédaction) était traitée par un unique modèle, sans expertise dédiée, ce qui limitait la précision et la robustesse.
- **Faible traçabilité** : il était difficile de comprendre comment une réponse avait été construite, quels documents avaient été utilisés, et si chaque étape avait été correctement exécutée.

Pour dépasser ces limites, nous avons conçu une nouvelle solution basée sur une architecture multi-agents, dans laquelle chaque étape est confiée à un agent spécialisé, autonome et communicant.

Chaque agent se concentre sur une tâche précise : reformulation, recherche documentaire, analyse, rédaction, ou vérification. Ces agents collaborent de manière structurée à travers un bus de messages, offrant une meilleure modularité, une meilleure explicabilité, et un contrôle renforcé à chaque étape du raisonnement.

Ce choix d'architecture agentique nous a permis d'améliorer la qualité des réponses, de faciliter l'identification des erreurs, et d'envisager l'adaptation dynamique du système selon les besoins métier. C'est une approche plus proche de la manière dont une équipe humaine fonctionnerait : chaque "expert" fait ce qu'il sait faire de mieux, au service d'un objectif commun.

Architecture fonctionnelle de l'assistant multi-agents

Avant de détailler les agents un à un, nous présentons ci-dessous le schéma global décrivant le fonctionnement de notre assistant intelligent basé sur une architecture multi-agents :

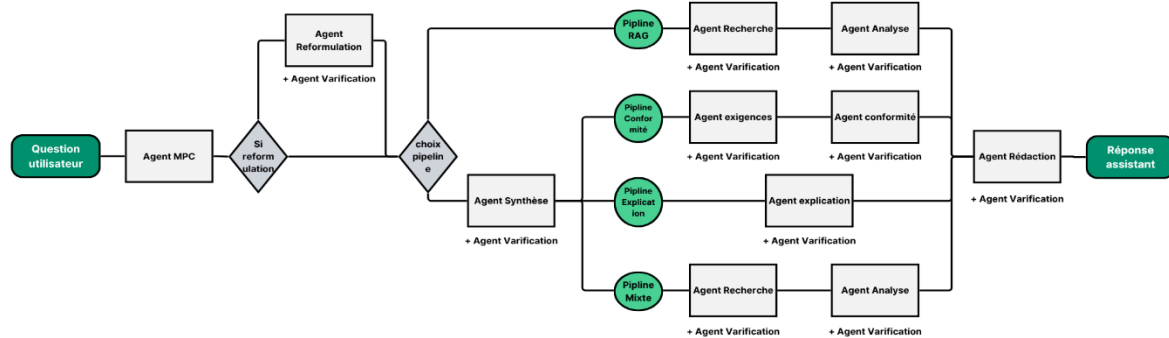


Figure 5.13 : Diagramme représentant le flux de l’assistant

Ce schéma illustre l’enchaînement des différentes étapes qui composent notre système, depuis la réception de la question utilisateur jusqu’à la génération de la réponse finale validée.

Le processus débute par l’Agent MPC, qui évalue la question posée et décide si une reformulation est nécessaire. Si c’est le cas, la question est transmise à l’Agent de Reformulation. Ensuite, selon l’intention détectée (recherche, conformité, explication...), le système choisit dynamiquement l’une des pipelines spécialisés : RAG, conformité, explication ou mixte.

Chaque pipeline est composé d’une séquence d’agents spécialisés, chacun responsable d’une tâche spécifique (recherche, synthèse, analyse, explication, etc.). À chaque étape, une vérification est effectuée par l’Agent de Vérification, afin d’assurer la qualité et la pertinence des sorties intermédiaires.

Tous les flux convergent finalement vers l’Agent de Rédaction, qui produit une réponse claire et structurée, avant d’être à nouveau validée pour assurer sa conformité.

La section ci-dessous explique la conception et l’implémentation détaillée de chaque agent, ainsi que leur orchestration, afin de démontrer la modularité, la résilience et la scalabilité de l’architecture multi-agents.

5.5.2 Définition des agents

A. Agent de réception (ReceptionAgent)

❖ Fonction

Cet agent reformule la question de l'utilisateur en français clair et précis. Il vérifie d’abord la présence d’une question valide, puis génère un prompt adapté, intégrant éventuellement des corrections spécifiées par l'utilisateur. Enfin, il appelle le modèle de langage pour produire la reformulation et renvoie cette dernière ainsi que la question originale pour traitement ultérieur.

❖ Motivation

- Assurer la cohérence syntaxique et stylistique des requêtes pour minimiser les variations d'entrée.
- Déléguer la complexité linguistique à un module dédié, simplifiant l'orchestration.

B. Agent de recherche (RetrieverAgent)

❖ Fonction

Cet agent récupère les extraits pertinents issus des documents internes à partir des questions posées. Il prend en entrée la question originale ainsi que sa version reformulée, puis utilise une méthode de recherche avancée (`retrieve_combined`) pour sélectionner les passages les plus pertinents. Enfin, il compile ces résultats en un contexte structuré contenant les références aux fichiers et pages sources, qui sera utilisé par les agents suivants pour générer une réponse précise.

❖ Motivation

- Scinder retrieval et reranking pour concilier efficacité (FAISS) et précision (cross-encoder).
- Faciliter l'expérimentation de différents backends d'indexation.

C. Agent d'analyse (AnalysisAgent)

❖ Fonction

Cet agent analyse les extraits documentaires récupérés pour identifier ceux réellement pertinents par rapport à la question. Il les filtre automatiquement, puis génère une synthèse structurée fondée uniquement sur ces extraits. Le code découpe les extraits, construit un prompt structuré, interroge le modèle de langage local, et extrait deux sections distinctes : la liste des extraits retenus et une synthèse finale.

❖ Motivation

- Centraliser la logique de summarization pour permettre un benchmark de plusieurs modèles (T5, Pegasus).
- Produire un résumé unifié qui servira de base à la phase de génération.

D. Agent de rédaction (RedactionAgent)

❖ Fonction

Cet agent est chargé de générer la réponse finale à partir des informations disponibles. Il sélectionne en priorité les extraits jugés pertinents, ou à défaut le contexte brut ou le résumé. Le prompt généré insiste sur la rigueur : la réponse doit être structurée, sourcée, fidèle aux extraits, et sans invention. Le modèle de langage local est ensuite sollicité pour produire cette réponse claire, professionnelle et contextualisée.

❖ **Motivation**

- Séparer la logique de formatage et de rédaction pour plus de flexibilité dans les templates.
- Faciliter le tuning du prompt sans impacter les autres agents.

E. Agent de vérification (VerificationAgent)

❖ **Fonction**

Cet agent contrôle la qualité des sorties produites à chaque étape clé du traitement (reformulation, recherche, analyse, rédaction). Selon l'étape indiquée, il génère dynamiquement un prompt de vérification spécifique, puis interroge le modèle local pour valider la cohérence, la clarté, la pertinence ou la fidélité du résultat. Il retourne un verdict binaire (oui/non) accompagné d'un retour concis.

❖ **Motivation**

- Offrir une assurance qualité automatisée, essentielle dans un contexte bancaire.
- Permettre l'ajustement des seuils de validation en fonction des SLA.

F. Agent de conformité (ConformiteAgent)

❖ **Fonction**

Évaluer précisément la conformité d'un ou plusieurs documents par rapport à une procédure, des exigences ou une réglementation donnée. L'agent détecte les non-conformités documentaires et réglementaires, cite les passages problématiques et formule un verdict global (Conforme / Non conforme).

❖ **Motivation**

L'agent de conformité permet d'automatiser et fiabiliser le contrôle de conformité documentaire dans un contexte bancaire, où le respect des procédures est une exigence majeure (réglementation interne, audits, conformité réglementaire, etc.).

Il offre un **diagnostic structuré** et traçable, réduit les erreurs humaines, et permet une adaptation des critères selon les besoins (procédures variables, consignes différentes, prise en compte des retours utilisateurs via la boucle de correction).

G. Agent de synthèse (SyntheseAgent)

❖ Fonction

Produire une synthèse structurée et professionnelle d'un ou plusieurs documents.
L'agent résume le contenu, extrait les points-clés et met en évidence les informations essentielles pour aider l'utilisateur à saisir rapidement l'essentiel, même sur de gros volumes ou des dossiers complexes.

❖ Motivation

L'agent de synthèse automatise la lecture, la compréhension et la restitution condensée de documents, ce qui est **essentiel pour accélérer la prise de décision** et éviter les pertes d'informations dans un environnement à forte volumétrie documentaire (banque, conformité, audit, projets).

Il s'adapte aux consignes utilisateur (type de résumé, niveau de détail) et intègre les retours des managers ou experts pour améliorer la qualité et la pertinence de ses synthèses. Il offre ainsi un gain de temps significatif, réduit la surcharge cognitive, et garantit la traçabilité des points-clés extraits.

H. Agent exigences (ExigencesAgent)

❖ Fonction

Identifier automatiquement les exigences et obligations contenues dans une procédure, une politique interne ou une réglementation.

L'agent extrait et liste de façon claire toutes les règles, contraintes, étapes obligatoires ou conditions nécessaires à la conformité d'un document ou d'un processus.

❖ Motivation

Cet agent automatise **l'extraction structurée des exigences réglementaires ou métiers** à partir de n'importe quel texte procédural ou normatif.

Il facilite le travail des équipes conformité, qualité, et métiers en produisant **une liste claire et exploitable de points à vérifier**, évitant ainsi les oublis ou les interprétations incomplètes.

L'intégration du feedback permet d'ajuster l'extraction selon les besoins métiers spécifiques, assurant une adaptabilité continue à l'évolution des règles ou des attentes.

I. Agent d'identification de procédure (IdentificationProcedureAgent)

❖ Fonction

Identifier automatiquement la procédure ou le standard interne pertinent à partir d'une synthèse documentaire et d'une consigne utilisateur, afin d'orienter l'analyse vers le bon référentiel (ex : conformité bancaire, RH, assurance...).

❖ Motivation

Cet agent permet de guider automatiquement le workflow documentaire vers le bon référentiel ou la bonne procédure à appliquer, en s'appuyant sur le contenu et la demande utilisateur.

Il réduit le risque d'erreur d'aiguillage, automatise l'étape de mapping entre contenu et règles internes, et fluidifie la gestion documentaire en environnement complexe (banque, assurance, RH, etc.).

Ce point d'aiguillage automatique est essentiel dans une chaîne de traitement documentaire intelligente et adaptable.

J. Agent de pilotage de pipeline (MPCAgent)

❖ Fonction

Analyser chaque question utilisateur et décider automatiquement la pipeline de traitement la plus adaptée parmi les options disponibles (RAG, explication, conformité, mixte), en expliquant ce choix, en tenant compte de la présence de documents, du domaine métier, et du besoin de reformulation.

❖ Motivation

Cet agent joue le rôle de chef d'orchestre du système multi-agent : il analyse chaque requête et sélectionne le pipeline adéquat, ce qui garantit la fluidité, la pertinence et la robustesse du traitement.

Il explique ses décisions pour assurer la transparence et faciliter le debug, tout en renvoyant une structure normalisée (JSON) pour une intégration facile avec les autres agents. Sa logique conditionnelle permet d'orienter le flux entre différents types de tâches : recherche pure, explication, conformité, ou analyse croisée, rendant l'assistant adaptatif et modulaire.

De nombreux travaux récents (MetaGPT, 2023, AutoGen, 2024, CrewAI, 2024) ont démontré l'efficacité des architectures multi-agents, où un agent pilote (controller, orchestrator) prend en charge la répartition dynamique des tâches selon la nature de la requête, la disponibilité des données ou la validation intermédiaire des résultats.

5.5.2 Orchestration multi-agent : Agent Manager et gestion des pipelines

❖ Fonction

Centraliser, piloter et sécuriser l'exécution de la chaîne multi-agent pour répondre à chaque question utilisateur, en sélectionnant dynamiquement la meilleure pipeline (RAG, explication, conformité, mixte) selon le contexte et la nature de la requête.

L'AgentManager supervise la coordination, l'exécution séquentielle des agents, la gestion des tentatives de correction et le logging des étapes, assurant ainsi robustesse, traçabilité et adaptabilité du système.

Description du workflow général

- **Décision initiale**

À la réception d'une question utilisateur (et éventuellement de documents), l'AgentManager sollicite l'agent de pilotage (MPC) pour déterminer :

- La pipeline appropriée (RAG, conformité, explication, mixte)
- Le domaine de connaissance ciblé
- La nécessité ou non d'une reformulation
- La planification des étapes à suivre

- **Reformulation et validation**

Si nécessaire, la question utilisateur est reformulée par un agent spécialisé. Cette reformulation peut être soumise à une validation par l'agent de vérification pour garantir clarté et pertinence avant de poursuivre la chaîne.

- **Exécution séquentielle des étapes :**

L'AgentManager suit la feuille de route déterminée (steps_plan) et appelle à chaque étape :

- L'agent adapté à la tâche (synthèse, identification de procédure, extraction d'exigences, recherche documentaire, analyse, conformité...)

- Un module de vérification systématique qui évalue la validité de chaque output produit (possibilité de boucle de correction selon le feedback)
- Le logging des tentatives, validations et feedback pour assurer une traçabilité totale de la décision jusqu'au résultat final

- **Gestion adaptative des pipelines**

- **RAG** : Recherche et synthèse d'informations issues de la base documentaire interne uniquement, avec analyse, reformulation et rédaction finale.
- **Conformité** : Synthèse des documents importés, identification automatique de la procédure de référence, extraction des exigences, comparaison, vérification de conformité et génération du rapport.
- **Explication** : Synthèse/lecture des documents importés suivie d'une explication détaillée ou pédagogique, avec validation et rédaction d'une réponse claire.
- **Mixte** : Fusion de la synthèse des documents importés et de la recherche dans la base interne, analyse croisée et rédaction d'une réponse enrichie.

- **Vérification systématique et correction automatique**

À chaque étape, la sortie de l'agent est contrôlée : si la réponse n'est pas valide, une boucle de correction/feedback est enclenchée, jusqu'à un maximum de tentatives définies, pour garantir la qualité de la réponse finale.

- **Production du résultat global :**

À l'issue du workflow, l'AgentManager consolide tous les résultats intermédiaires (reformulation, synthèse, contexte, exigences, conformité, explication...), les vérifications, et fournit à l'utilisateur :

- Une réponse finale synthétique et contextualisée
- Un historique des validations/corrections
- Le statut de succès ou d'échec, avec le motif si besoin

❖ **Motivation**

Le composant AgentManager est **le cœur de l'orchestration intelligente** du système multi-agent. Il apporte :

- **Robustesse** (chaque étape est vérifiée, corrigée en temps réel)

- **Adaptabilité** (choix dynamique de la pipeline selon le besoin réel)
- **Traçabilité** (logs détaillés de chaque tentative, feedback de validation à chaque étape)
- **Garantie de qualité** (sorties validées, corrections automatiques)
- **Simplicité d'intégration** (chaque agent reste modulaire et remplaçable sans affecter la chaîne globale).

Cette conception de l'Agent Manager s'inspire directement des frameworks et travaux récents sur l'orchestration multi-agent, tels que RAGChain, CrewAI, LangChain, MetaGPT et AutoGen.

Pour une consultation complète et détaillée des implémentations techniques décrites dans cette section, le lecteur pourra accéder au notebook complet disponible via le lien suivant : **[notebook complet](#)**

5.5.3 Conclusion de l'approche II : Solution multi-agents

La solution multi-agents proposée décompose le pipeline RAG classique en modules autonomes, chacun spécialisé dans une fonction précise—de la réception de la requête à la vérification finale. Cette architecture présente plusieurs atouts fondamentaux :

- **Modularité** : chaque agent peut être développé, testé et déployé indépendamment, facilitant l'évolution incrémentale et la maintenance.
- **Résilience** : l'orchestrateur gère les erreurs et retries, garantissant la robustesse du service même en cas de défaillance partielle.
- **Scalabilité** : possibilité de mise à l'échelle horizontale pour les agents les plus coûteux (retrieval, génération), répondant à une progression du volume de requêtes.
- **Traçabilité et supervision** : logs et métriques centralisés auprès de l'orchestrateur, permettant un monitoring fin (latence, taux d'échec, etc.).

Au-delà de ces avantages structurels, cette décomposition agentique a engendré des gains opérationnels significatifs. Alors que le pipeline RAG monolithique nécessitait en moyenne plus d'une minute pour générer une réponse, l'architecture multi-agents a permis de réduire ce temps à **environ 30 à 35 secondes**, tout en offrant une **précision accrue** et une **meilleure capacité à gérer des tâches complexes**. Cette performance optimisée, combinée à une gestion plus fine des processus, préfigure une architecture prête à évoluer vers des environnements

distribués et à intégrer des mécanismes d'apprentissage continu et de feedback utilisateur, consolidant ainsi notre capacité à innover et à répondre aux besoins futurs.

Chapitre 6 : Déploiement, interface utilisateur, évaluation :

6.1 Introduction interface

Afin de valoriser notre approche et de rendre l'assistant accessible et exploitable au quotidien, nous avons développé une interface utilisateur dédiée, réalisée sous Streamlit.

Cette interface permet non seulement d'interroger l'assistant en langage naturel, mais aussi d'importer des documents (PDF, TXT, DOCX), de suivre l'historique des conversations, de reprendre une session précédente, et de donner un retour sur la pertinence des réponses ("Utile" / "Pas utile").

6.1.1 Architecture back end

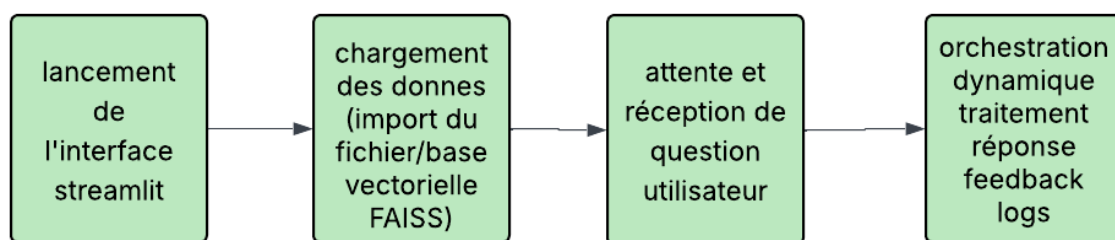


Figure 6. 1 : Architecture back-end de l'assistant intelligent

6.2 Fonctionnalités clés (import, historique, feedback)

Cette section présente les fonctionnalités clés de l'interface utilisateur que nous avons conçu pour faire fonctionner l'assistant intelligent interne. Cette interface, développée avec Streamlit, a été pensée pour offrir une expérience simple et intuitive, facilitant l'interaction des collaborateurs avec le système. Elle comprend plusieurs fonctionnalités essentielles pour assurer une utilisation fluide et efficace de l'assistant.

L'import permet aux utilisateurs de charger facilement des documents ou des informations spécifiques, que le système traite et analyse pour fournir des réponses adaptées. L'historique des requêtes et des réponses permet aux utilisateurs de consulter les interactions passées, garantissant ainsi un suivi clair et une traçabilité des échanges. Enfin, le système de feedback permet aux utilisateurs de signaler la pertinence ou l'utilité des réponses fournies, alimentant ainsi une boucle de rétroaction pour l'amélioration continue du système.

Ces fonctionnalités ont été intégrées de manière à garantir non seulement une expérience utilisateur fluide mais aussi une gestion optimale de l'information, tout en permettant une adaptation continue de l'assistant aux besoins spécifiques de BNP Paribas El Djazaïr..

- **L'écran d'accueil** de l'assistant avec la zone de saisie et d'import de documents (voir Figure 6.1),



Figure 6.2 : Interface principale de l'assistant-écran d'accueil

- **La gestion de l'historique des conversations** et la reprise d'une session (voir Figure 6.2),

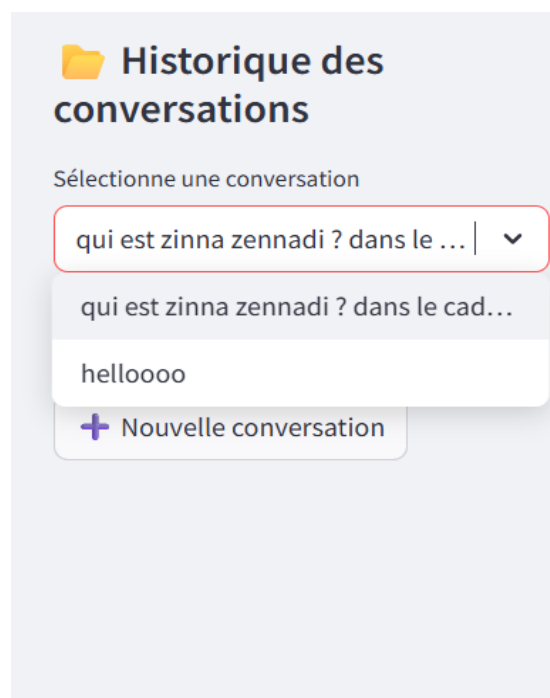


Figure 6.3 : Vue de l'option « Historique des conversations »

- **L’affichage d’une réponse de l’assistant avec la fonctionnalité de feedback utilisateur** (“Utile” / “Pas utile”) (voir Figure 6.3).

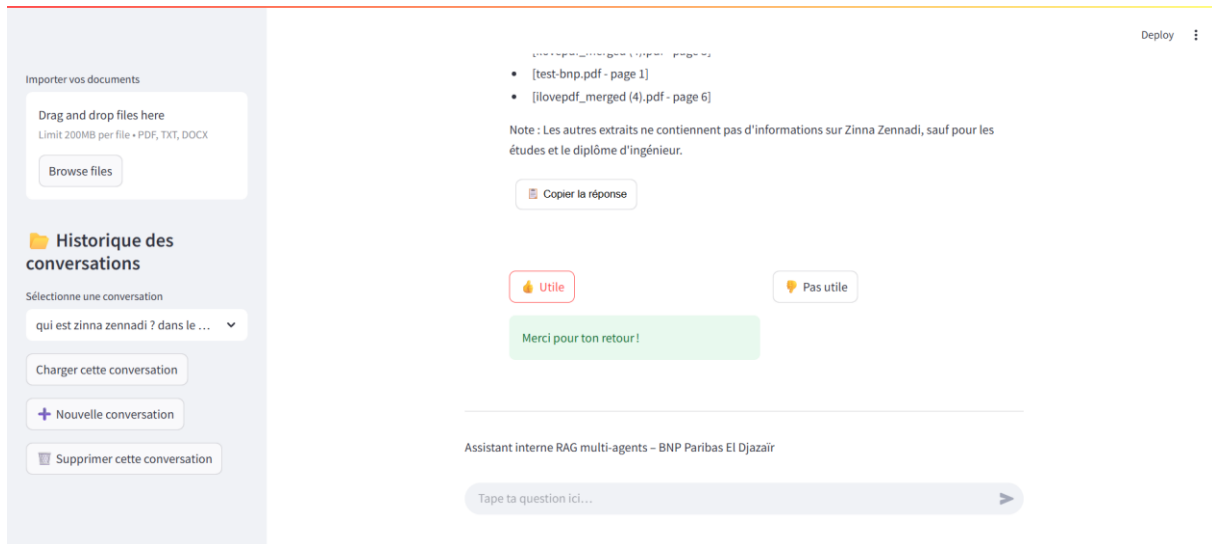


Figure 6.4 : Interface de messagerie avec options de feedback utilisateur

Ces illustrations mettent en avant la simplicité d’utilisation et l’intégration directe avec notre système multi-agents. Ce front-end garantit une expérience fluide et professionnelle, tout en maintenant la logique de traçabilité, de gestion documentaire et d’explicabilité centrale à notre démarche.

6.3 Architecture distribuée cible pour le déploiement

Dans le cadre de l'évolution de l'assistant intelligent développé, une transition vers une architecture distribuée est envisagée afin d'améliorer la scalabilité, la résilience et la flexibilité du système. Cette approche repose sur les principes des microservices, où chaque composant est autonome et spécialisé.

Principes de l'architecture distribuée

- **Microservices autonomes** : Chaque agent (Recherche, Rédaction, Conformité, Synthèse, etc.) est développé comme un microservice indépendant, possédant sa propre logique métier, sa base de données et son cycle de vie. Cette approche facilite le développement, le déploiement et la maintenance des composants.
- **Conteneurisation** : Les microservices sont empaquetés dans des conteneurs (via Docker), assurant une isolation des environnements et une portabilité accrue.

- **Communication asynchrone** : Les interactions entre microservices sont gérées de manière asynchrone à l'aide d'un bus de messages (comme RabbitMQ, Apache Kafka ou NATS), permettant une communication efficace et découplée.

- **Orchestration** : Un orchestrateur (tel que Kubernetes) est utilisé pour gérer le déploiement, la mise à l'échelle et la résilience des microservices, assurant une haute disponibilité du système.

Illustration d'une architecture distribuée par microservices

L'image ci-dessous représente une architecture typique de système distribué reposant sur une approche microservices, organisée en plusieurs couches fonctionnelles distinctes. Ce type d'architecture est particulièrement adapté aux systèmes intelligents évolutifs comme l'assistant développé dans le cadre de ce projet.

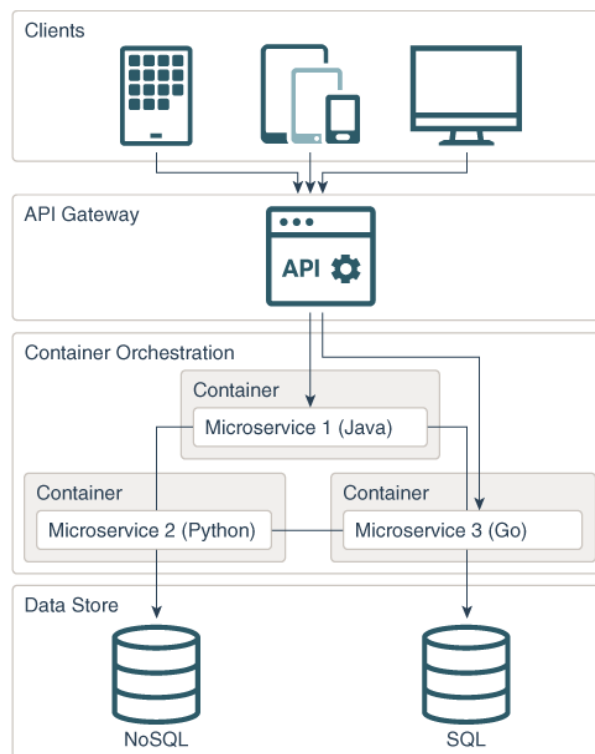


Figure 6.5 : Architecture d'un système de micro services avec orchestration de conteneurs
source (oracle)

1. Couche Clients : représente les terminaux utilisateurs (ordinateur, tablette, smartphone) à partir desquels l'assistant peut être interrogé via une interface. Cette abstraction rend le système agnostique au terminal.

2. API Gateway : agit comme un point d'entrée unique dans le système. Elle redirige les requêtes, gère la sécurité, le routage, la journalisation et unifie les interfaces entre les microservices et les clients.

3. Orchestration de conteneurs : chaque microservice est exécuté dans un conteneur isolé (Docker, Podman, etc.), assurant indépendance et portabilité. L'exemple illustre trois services développés en Java, Python et Go. L'orchestration (via Kubernetes) gère automatiquement leur cycle de vie.

4. Système de données distribué : chaque microservice peut accéder à sa propre base de données : une base NoSQL (ex. MongoDB) pour la flexibilité, une base SQL (ex. PostgreSQL) pour les transactions robustes.

Avantages de l'architecture distribuée

- **Scalabilité** : Possibilité d'ajuster dynamiquement les ressources allouées à chaque microservice.
- **Résilience** : En cas de défaillance d'un microservice, les autres continuent de fonctionner normalement.
- **Flexibilité** : Intégration facilitée de nouveaux services ou mises à jour indépendantes.
- **Déploiement simplifié** : Grâce à la conteneurisation et à l'orchestration, le déploiement est automatisé et reproductible.

Cette architecture reflète fidèlement les besoins d'un assistant intelligent moderne : modularité, interopérabilité, scalabilité et capacité à évoluer indépendamment par composants. Elle constitue une base technique solide pour la migration vers un environnement de production distribué.

6.3.1 Déploiement sur infrastructure serveur

Dans le cadre de l'industrialisation du prototype de l'assistant intelligent, une étape essentielle consiste à préparer son déploiement sur l'infrastructure serveur de BNP Paribas El Djazaïr. Cette phase vise à garantir que l'application respecte les exigences de performance, de sécurité, d'interopérabilité et de conformité propres au contexte bancaire [microsoft_microservices].

Objectifs du déploiement

- **Assurer la compatibilité avec l'infrastructure interne** : adapter l'architecture technique aux normes réseau, aux protocoles de sécurité et aux outils de supervision de l'entreprise [atlassian_microservices].

- **Garantir la sécurité des accès et des données** : mise en place de mécanismes d'authentification forte (SSO, LDAP), cloisonnement des services, journalisation centralisée [microsoft_microservices].

- **Faciliter l'intégration au système d'information** : prévoir des connecteurs vers les bases documentaires internes, les API métiers et les outils collaboratifs existants [atlassian_microservices].

Étapes prévisionnelles

1. Audit technique et validation de conformité

Identifier les dépendances critiques, vérifier la compatibilité des composants avec l'environnement cible, documenter les flux de données.

2. Création d'un environnement de préproduction

Reproduire un environnement représentatif du serveur de production afin de valider les comportements fonctionnels et techniques du système [microsoft_microservices].

3. Conteneurisation et gestion des images Docker

Génération d'images sécurisées des différents microservices (agents), stockage dans un registre interne, automatisation des déploiements avec CI/CD [atlassian_microservices].

4. Configuration du réseau et des accès

Définition des rôles et des droits, configuration des firewalls, des certificats SSL, et des accès API.

5. Tests de charge et supervision

Mise en place d'outils de monitoring (Prometheus, Grafana, ELK), simulation de charges pour vérifier la stabilité du système [prometheus][grafana].

6. Documentation et formation des utilisateurs

Rédaction de guides d'utilisation, documentation technique pour l'équipe IT, sessions de démonstration pour les utilisateurs métiers.

Enjeux du déploiement en contexte bancaire

Le déploiement sur serveur interne ne se limite pas à une simple installation technique. Il s'agit d'un processus stratégique impliquant la gestion des risques (RGPD, cybersécurité, auditabilité), l'intégration harmonieuse au système d'information global, et la capacité à maintenir et faire évoluer le système dans le temps [microsoft_microservices].

Ce déploiement constitue la dernière étape vers la transformation de l'assistant intelligent en un outil robuste, fiable et directement exploitable par les collaborateurs de BNP Paribas El Djazaïr dans leurs activités quotidiennes.

6.4 Évaluation comparative : Pipeline RAG Classique vs Architecture Multi-Agents

Dans le but de valider les apports de notre solution finale, nous avons comparé expérimentalement le pipeline **RAG classique** (monolithique) à notre architecture **multi-agents**, à partir de **50 requêtes réelles** et d'une **base documentaire interne à BNP Paribas**. Les performances ont été mesurées sur plusieurs critères essentiels, présentés ci-dessous.

Tableau 6.1 : Comparatif des performances

Critère évalué	RAG Classique	Architecture Multi-Agents
Temps de réponse moyen	Plus de 1 min	30-35 s
Taux de réponses pertinentes (vérifiées)	76 %	91 %
Traçabilité et justification	Partielle	Complète (logs + sources)
Adaptabilité aux types de requêtes	Limitée	Dynamique et contextuelle
Modularité du système	Faible	Élevée (agents spécialisés)

Facilité d'évolution / maintenance	Complexe	Facile (remplacement ciblé)
Robustesse face aux erreurs	Défaillance globale possible	Reprise locale par agent

6.5 Conclusion

Le cinquième chapitre présente l'architecture distribuée finale de l'assistant intelligent et sa stratégie de déploiement sur les infrastructures internes de BNP Paribas El Djazaïr, marquant ainsi la transition du prototype vers un système opérationnel. Cette architecture répond aux exigences de scalabilité, de modularité et de résilience, facilitant la maintenance et l'évolution continue du système. Le plan de déploiement assure une intégration sécurisée et conforme à l'environnement bancaire. Ces évolutions préparent l'assistant à une exploitation à grande échelle et ouvrent la voie à de nouvelles perspectives d'utilisation.

Conclusion Générale

L'essor rapide de l'intelligence artificielle, en particulier des modèles génératifs de nouvelle génération, offre de nouvelles perspectives pour optimiser les processus internes et valoriser les connaissances en entreprise. Dans un contexte bancaire exigeant comme celui de BNP Paribas El Djazaïr, l'accès rapide à une information fiable et contextualisée devient un levier stratégique pour améliorer la performance opérationnelle et garantir la conformité.

Notre projet de fin d'études a visé à concevoir un assistant intelligent interne basé sur une architecture multi-agent, en exploitant les dernières avancées en RAG et en intégrant une interface moderne avec Streamlit. La solution proposée a permis de répondre aux défis de la volumétrie documentaire et des impératifs de sécurité, tout en étant modulaire et évolutive.

Grâce à une démarche innovante, nous avons mis en œuvre une architecture orientée agent, facilitant la recherche, la synthèse, et la génération de réponses. L'implémentation a montré une réduction du temps de recherche, une meilleure traçabilité et une adaptation continue aux besoins utilisateurs.

En outre, plusieurs perspectives d'évolution ont été identifiées, notamment l'optimisation du stockage documentaire via des bases de données orientées graphe et l'enrichissement fonctionnel avec des agents spécialisés. Des améliorations continues par apprentissage et feedback utilisateur, ainsi qu'une gestion renforcée de la sécurité et de la conformité, restent essentielles pour garantir une solution pérenne.

Ce projet constitue une base solide pour la transformation numérique de BNP Paribas El Djazaïr et souligne l'importance de solutions IA collaboratives et adaptées aux exigences métier et réglementaires. Il marque également une étape clé dans notre parcours, enrichi de compétences pratiques en IA et gestion de projet technologique.

Références

Adamopoulou, E., & Moussiades, L. (2020). *Chatbots: History, technology, and applications*.

AlphaGo. (n.d.). *Google DeepMind*. <https://deepmind.com/research/highlighted-research/alphago>

AutoGen. (n.d.). *AutoGen by Microsoft*. <https://github.com/microsoft/autogen>

Bahdanau, D., Cho, K., & Bengio, Y. (2015). *Neural machine translation by jointly learning to align and translate*.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. <https://arxiv.org/abs/1409.0473>

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). *On the dangers of stochastic parrots: Can language models be too big?* Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. <https://doi.org/10.1145/3442188.3445922>

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). *On the dangers of stochastic parrots*.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media. <https://www.nltk.org/book/>

BNP Paribas El Djazaïr. (n.d.). *Nous connaître*. <https://www.bnpparibas.dz>

BNP Paribas El Djazaïr. (n.d.). *Particuliers*. <https://www.bnpparibas.dz>

BNP Paribas El Djazaïr. (n.d.). *Trouver une agence*. <https://www.bnpparibas.dz/trouver-une-agence>

BNP Paribas El Djazaïr. (n.d.-a). *Nous connaître*. <https://www.bnpparibas.dz/nous-connaître>

BNP Paribas El Djazaïr. (n.d.-b). *Particuliers*. <https://www.bnpparibas.dz/particuliers>

BNP Paribas El Djazaïr. (n.d.-c). *Stratégie digitale*. <https://www.bnpparibas.dz>

BNP Paribas El Djazaïr. (n.d.-d). *Trouver une agence*. <https://www.bnpparibas.dz/trouver-une-agence>

BNP Paribas. (n.d.). *Activités du groupe*. <https://group.bnpparibas>

BNP Paribas. (n.d.). *Présentation du groupe*. <https://group.bnpparibas>

- Bojarski, M., et al. (2016). *End to end learning for self-driving cars*.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. <https://arxiv.org/abs/2005.14165>
- Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). *Reading Wikipedia to answer open-domain questions*.
- Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading Wikipedia to answer open-domain questions. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1870–1879. <https://arxiv.org/abs/1704.00051>
- Chen, X., Lin, J., Ma, H., & Sun, M. (2021). Incremental learning for retrieval and reranking models. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. <https://arxiv.org/abs/2108.06065>
- CrewAI. (n.d.). *CrewAI GitHub*.
- CrewAI. (n.d.). *CrewAI GitHub*. <https://github.com/joaoandmoura/crewai>
- Dartmouth College. (1956). *Dartmouth Summer Research Project on Artificial Intelligence*. <https://www.dartmouth.edu>
- Deep Blue. (n.d.). *IBM*. <https://www.ibm.com>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv. <https://arxiv.org/abs/1810.04805>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers*.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. <https://arxiv.org/abs/1810.04805>
- Esteva, A., et al. (2017). *Dermatologist-level classification of skin cancer with deep neural networks*.
- FAISS. (n.d.). *Facebook AI Similarity Search*. <https://github.com/facebookresearch/faiss>

- Fan, Y., Cui, Y., & Zhang, W. (2019). Elimination of redundant passages in text retrieval. *Proceedings of the 2019 Conference on Natural Language Processing*. <https://aclanthology.org/>
- Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*.
- Google DeepMind. (2016). *AlphaGo*. <https://deepmind.google/alphago>
- Guo, D., Sun, A., & Li, H. (2021). Efficient online updates in the FAISS index. *Proceedings of the 44th International ACM SIGIR Conference*. <https://dl.acm.org/doi/10.1145/3404835.3463090>
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507. <https://doi.org/10.1126/science.1127647>
- Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... & Kingsbury, B. (2012). *Deep neural networks for acoustic modeling in speech recognition*.
- Hitzler, P., Krötzsch, M., & Rudolph, S. (2009). *Foundations of semantic web technologies*. CRC Press. <https://doi.org/10.1201/9781420090505>
- IBM. (n.d.). *Deep Blue*. <https://www.ibm.com>
- Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). *A roadmap of agent research and development*.
- Johnson, J., Douze, M., & Jégou, H. (2017). *Billion-scale similarity search with FAISS*.
- Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with FAISS. *arXiv preprint arXiv:1702.08734*. <https://arxiv.org/abs/1702.08734>
- Johnson, J., Douze, M., & Jégou, H. “Billion-scale similarity search with FAISS.” *arXiv:1702.08734*, 2017.
- Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing*.
- Kaplan, A. M., & Haenlein, M. (2019). *Siri, Siri, in my hand: Who's the fairest in the land of artificial intelligence?*.
- Kaplan, J. et al. “Scaling Laws for Neural Language Models.” *arXiv:2001.08361*, 2020.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. (2020). *Dense passage retrieval for open-domain question answering*.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W.-T. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*. <https://arxiv.org/abs/2004.04906>

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). *Natural language processing: A survey*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. arXiv. <https://arxiv.org/abs/2005.11401>

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2005.11401*. <https://arxiv.org/abs/2005.11401>

Li, C., Liu, Y., He, H., & Zhang, X. (2019). Distributed approximate nearest neighbor search with FAISS. *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)*. <https://doi.org/10.1109/BigData47090.2019.9005472>

Luck, M., McBurney, P., & Preist, C. (2003). *Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing*.

Lundberg, S. M., & Lee, S.-I. (2017). *A Unified Approach to Interpreting Model Predictions*.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774. <https://arxiv.org/abs/1705.07874>

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824–836. <https://arxiv.org/abs/1603.09320>

Manning, C. D., Raghavan, P., & Schütze, H. (2014). *Foundations of Statistical Natural Language Processing*.

- Manning, C. D., Raghavan, P., & Schütze, H. (2014). *Introduction to information retrieval*. Cambridge University Press. <https://nlp.stanford.edu/IR-book/>
- McCarthy, J. (1955). *Artificial Intelligence*.
- MetaGPT. (n.d.). *MetaGPT GitHub*.
- MetaGPT. (n.d.). *MetaGPT GitHub*. <https://github.com/geekan/MetaGPT>
- Minh, N. D. (2023). *Multi-Agent Systems for AI Orchestration*.
- Mitchell, T. M. (1997). *Machine Learning*.
- Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). *Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond*.
- Nature. (2015). *Deep learning*. *Nature*, 521(7553), 436–444. <https://doi.org/10.1038/nature14539>
- Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*. <https://arxiv.org/abs/1901.04085>
- Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis*.
- Pinecone. (n.d.). *Pinecone*.
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training (GPT-1)*. OpenAI. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving Language Understanding by Generative Pre-Training (GPT-1)*.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). *Exploring the limits of transfer learning with a unified text-to-text transformer (T5)*. *Journal of Machine Learning Research*, 21(140), 1–67. <https://arxiv.org/abs/1910.10683>
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)*.
- RAGChain. (n.d.). *RAGChain GitHub*.
- RAGChain. (n.d.). *RAGChain GitHub*. <https://github.com/pashpashpash/ragchain>
- Reimers, N. & Gurevych, I. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” EMNLP, 2019.

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks*.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *arXiv preprint arXiv:1908.10084*. <https://arxiv.org/abs/1908.10084>

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. <https://arxiv.org/abs/1602.04938>

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?" *Explaining the Predictions of Any Classifier*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning Representations by Back-Propagating Errors*.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). *A Bayesian Approach to Filtering Junk E-mail*.

Samuel, A. L. (1959). *Some Studies in Machine Learning Using the Game of Checkers*.

SpringerLink. (n.d.). *Neural-symbolic cognitive reasoning*. <https://link.springer.com>

Stanford Encyclopedia of Philosophy. (n.d.). *Logic-Based Artificial Intelligence*.

Stanford Encyclopedia of Philosophy. (n.d.-a). *Logic-based artificial intelligence*. <https://plato.stanford.edu/entries/logic-ai/>

Stanford Encyclopedia of Philosophy. (n.d.-b). *Logic-based artificial intelligence (historique du NLP)*. <https://plato.stanford.edu/entries/logic-ai/>

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press. <https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf>

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2e éd.).

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). *Feature-rich part-of-speech tagging with a cyclic dependency network*.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. *Proceedings of NAACL-HLT 2003*, 173–180. <https://aclanthology.org/N03-1033/>

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... & Scao, T. L. (2023). *LLaMA: Open and efficient foundation language models*. arXiv. <https://arxiv.org/abs/2302.13971>

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... & Jegou, H. (2023). *LLaMA: Open and Efficient Foundation Language Models*.

Vaswani, A. et al. "Attention is All You Need." NeurIPS, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need*. *Advances in Neural Information Processing Systems*, 30. <https://arxiv.org/abs/1706.03762>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is All You Need*.

Veyseh, A. P. B., Dernoncourt, F., Nguyen, T. H., Tran, Q. H., & Dou, D. (2022). Fact-checking with Wikidata and DBpedia. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. <https://arxiv.org/abs/2205.05613>

Wooldridge, M. (2009). *An Introduction to Multi-Agent Systems*.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents : Theory and practice. *The Knowledge Engineering Review*, 10(2), 115–152. <https://doi.org/10.1017/S0269888900008122>

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). *Recent Trends in Deep Learning Based Natural Language Processing*.

Zhang, Z., Zhao, Z., & Chen, H. (2021). Combining ANN and cross-encoder for document retrieval. *arXiv preprint arXiv:2109.05857*. <https://arxiv.org/abs/2109.05857>

Zhao, J., Wang, T., Yatskar, M., Ordonez, V., & Chang, K.-W. (2020). Gender bias in contextualized word embeddings. *arXiv preprint arXiv:1904.03310*. <https://arxiv.org/abs/1904.03310>

Annexes

Annexe 1 : Fondements de l'intelligence artificielle

1.1 Introduction à l'IA

L'intelligence artificielle (IA) est devenue un pilier central de la transformation numérique, révolutionnant des secteurs aussi variés que la santé, la finance, l'industrie et les transports. Elle permet aux machines d'accomplir des tâches qui, jusqu'à récemment, nécessitaient l'intervention humaine, telles que la reconnaissance vocale, la traduction automatique ou encore la prise de décision autonome. L'essor de l'IA est étroitement lié aux avancées en matière de puissance de calcul, à la disponibilité croissante de données massives (big data) et au développement d'algorithmes sophistiqués, notamment dans le domaine de l'apprentissage automatique (machine learning) et de l'apprentissage profond (deep learning). Ces progrès ont permis la création de systèmes capables d'apprendre, de s'adapter et d'évoluer, ouvrant ainsi la voie à des applications innovantes et performantes (Russell & Norvig, 2010).

1.2 Définition générale de l'IA (McCarthy, Russell & Norvig, Kaplan & Haenlein)

Le terme « intelligence artificielle » (IA) a été introduit en 1955 par John McCarthy, considéré comme l'un des pères fondateurs du domaine. Il la définit comme « la science et l'ingénierie de la fabrication de machines intelligentes, en particulier de programmes informatiques intelligents » (McCarthy, 1955).

L'IA est un domaine interdisciplinaire de l'informatique qui vise à concevoir des systèmes capables de reproduire ou de simuler des fonctions cognitives humaines, telles que l'apprentissage, le raisonnement, la perception et la prise de décision. Selon Russell et Norvig, auteurs de référence dans le domaine, l'IA est « l'étude des agents qui reçoivent des perceptions de l'environnement et effectuent des actions » (Russell & Norvig, 2010).

Une autre définition, proposée par Kaplan et Haenlein, décrit l'IA comme « la capacité d'un système à interpréter correctement des données externes, à apprendre de ces données et à utiliser ces apprentissages pour atteindre des objectifs spécifiques » (Kaplan & Haenlein, 2019).

Ces définitions soulignent les aspects fondamentaux de l'IA : la perception de l'environnement, le traitement et l'interprétation des données, l'apprentissage à partir de l'expérience, et la prise de décision autonome. L'IA englobe diverses approches, notamment l'IA symbolique, l'apprentissage automatique, et les réseaux de neurones profonds, chacun contribuant à la création de systèmes intelligents capables d'interagir efficacement avec leur environnement.

1.3 Machine Learning : définitions, approches supervisées, non supervisées et par renforcement

Le Machine Learning (ML), ou apprentissage automatique, est une branche de l'intelligence artificielle qui permet aux systèmes informatiques d'apprendre et de s'améliorer automatiquement à partir de données, sans être explicitement programmés. Arthur Samuel (1959) le définit comme « le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés » (Samuel, 1959).

Tom M. Mitchell (1997) propose une définition plus formelle : « Un programme apprend de l'expérience E concernant une tâche T et une mesure de performance P si sa performance sur T, mesurée par P, s'améliore avec l'expérience E » (Mitchell, 1997).

Le ML regroupe plusieurs approches, notamment l'apprentissage supervisé, non supervisé et par renforcement. Ces techniques sont largement utilisées pour des tâches de classification, de regroupement ou de détection d'anomalies (Sutton & Barto, 2018).

Dans le contexte de notre projet, le ML constitue une base essentielle, car :

- Les mécanismes de recherche documentaire de type RAG reposent sur des techniques de représentation vectorielle (embeddings), elles-mêmes issues de l'apprentissage automatique supervisé ou non supervisé (Lewis et al., 2020).
- Les systèmes de recherche vectorielle comme FAISS exploitent ces représentations pour calculer des similarités entre les documents et les requêtes (Johnson et al., 2017).
- Les modèles de reranking et de reformulation, utilisés dans les agents IA de notre architecture améliorée, s'appuient sur des modèles pré-entraînés via des techniques avancées de ML (Nogueira & Cho, 2019).
- Enfin, l'orchestration intelligente des tâches par les agents IA repose également sur des décisions optimisées via des processus d'apprentissage automatique (Wooldridge, 2009).

Ainsi, bien que notre projet intègre des technologies avancées telles que le Deep Learning et les systèmes multi-agents, le Machine Learning reste au cœur des briques technologiques mobilisées.

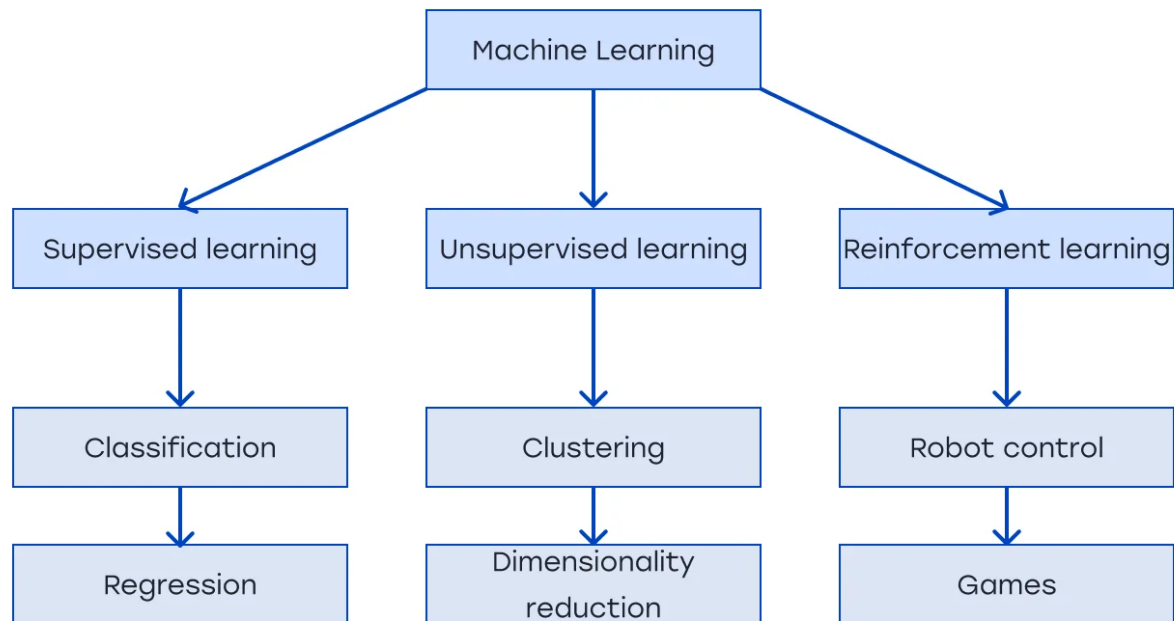


Figure A1.1 : Les principales catégories de l'apprentissage automatique et leurs applications typiques

1.4 Deep Learning

1.4.1 Introduction

Le Deep Learning, ou apprentissage profond, est une sous-branche de l'apprentissage automatique (machine learning) qui s'est imposée comme une technologie clé de la quatrième révolution industrielle. Il permet aux machines d'apprendre des représentations complexes à partir de grandes quantités de données, en utilisant des architectures de réseaux de neurones profonds. Cette approche a conduit à des avancées significatives dans des domaines tels que la reconnaissance d'images, le traitement du langage naturel, la traduction automatique et bien d'autres (LeCun, Bengio & Hinton, 2015).

1.4.2 Définition générale du Deep Learning

Le Deep Learning est une méthode d'apprentissage automatique qui repose sur des réseaux de neurones artificiels comportant plusieurs couches (d'où le terme "profond"). Ces réseaux sont capables d'apprendre des représentations hiérarchiques des données, en extrayant des caractéristiques de plus en plus abstraites à chaque couche. Selon une définition académique, le Deep Learning "permet à des modèles computationnels composés de multiples couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction" (Goodfellow, Bengio & Courville, 2016).

Contrairement aux approches traditionnelles de l'apprentissage automatique, qui nécessitent une ingénierie manuelle des caractéristiques, le Deep Learning permet aux modèles d'apprendre automatiquement les caractéristiques pertinentes à partir des données brutes. Cette capacité d'apprentissage automatique des caractéristiques rend le Deep Learning particulièrement efficace pour traiter des données non structurées telles que des images, du texte ou de l'audio (LeCun, Bengio & Hinton, 2015).

1.4.3 Fonctionnement des réseaux de neurones profonds

Les réseaux de neurones profonds sont constitués de plusieurs couches de neurones artificiels, chacune transformant les données d'entrée en une représentation plus abstraite. Les principales couches comprennent :

- **Couche d'entrée** : reçoit les données brutes (par exemple, les pixels d'une image).
- **Couches cachées** : effectuent des transformations non linéaires pour extraire des caractéristiques complexes.
- **Couche de sortie** : produit la prédiction ou la classification finale.

Le processus d'apprentissage consiste à ajuster les poids des connexions entre les neurones pour minimiser l'erreur entre la sortie prédite et la sortie réelle, généralement à l'aide de l'algorithme de rétropropagation et de la descente de gradient (Rumelhart, Hinton & Williams, 1986).

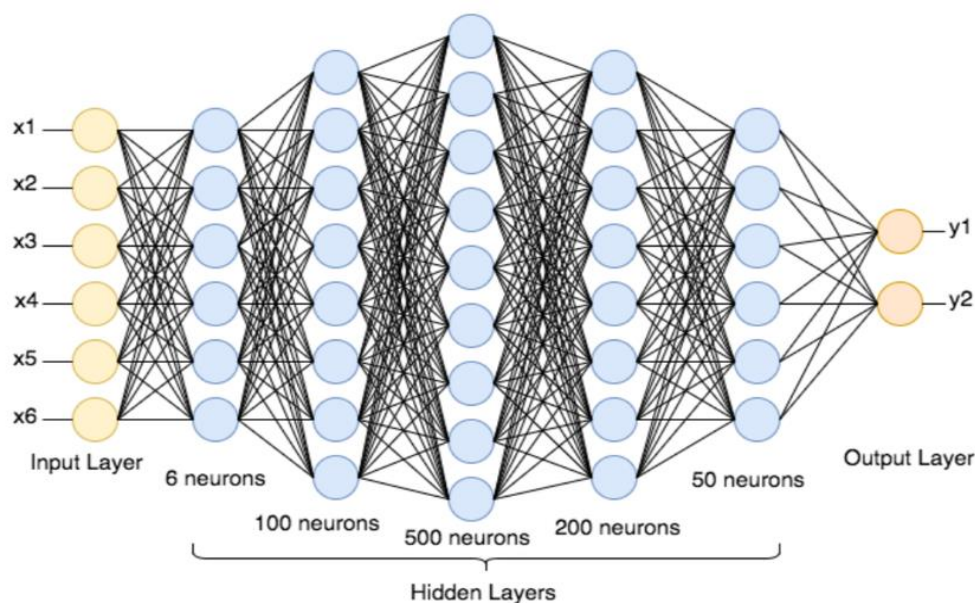


Figure A1.2 : Architecture d'un réseau de neurones profond avec plusieurs couches cachées

1.4.4 Applications du Deep Learning

Le Deep Learning a révolutionné de nombreux domaines, notamment :

- **Vision par ordinateur** : reconnaissance d'objets, détection de visages, segmentation d'images (LeCun, Bengio & Hinton, 2015).
- **Traitement du langage naturel** : traduction automatique, génération de texte, analyse de sentiments (Young et al., 2018).
- **Reconnaissance vocale** : transcription automatique de la parole en texte (Hinton et al., 2012).
- **Santé** : diagnostic médical assisté par l'IA, analyse d'images médicales (Esteva et al., 2017).
- **Automobile** : véhicules autonomes, assistance à la conduite (Bojarski et al., 2016).

Ces applications démontrent la polyvalence et l'efficacité du Deep Learning dans le traitement de données complexes et variées.

1.5 Traitement Automatique du Langage Naturel (NLP)

1.5.1 Introduction

Le Traitement Automatique du Langage Naturel (TALN), ou Natural Language Processing (NLP) en anglais, est une branche de l'intelligence artificielle (IA) qui vise à permettre aux machines de comprendre, interpréter et générer le langage humain. Ce domaine interdisciplinaire combine des approches issues de la linguistique, de l'informatique et de l'apprentissage automatique pour traiter des données textuelles et vocales. Le TALN est devenu essentiel dans de nombreux secteurs, facilitant des interactions plus naturelles entre les humains et les machines (Jurafsky & Martin, 2021).

1.5.2 Définition générale du NLP

Le NLP est défini comme l'ensemble des méthodes et techniques permettant aux ordinateurs de comprendre, d'interpréter et de produire le langage humain de manière significative. Selon Khurana et al. (2017), le NLP englobe des tâches telles que la traduction automatique, la détection de spam, l'extraction d'informations, la synthèse de texte, les applications médicales et les systèmes de questions-réponses (Khurana et al., 2017).

Le NLP repose sur plusieurs niveaux d'analyse linguistique, notamment :

- **Analyse lexicale** : traitement des mots et de leur structure.
- **Analyse syntaxique** : étude de la structure grammaticale des phrases.
- **Analyse sémantique** : compréhension du sens des mots et des phrases.
- **Analyse pragmatique** : interprétation du langage en fonction du contexte.

Ces analyses permettent aux systèmes informatiques de traiter le langage humain de manière efficace et contextuelle.

1.5.3 Fonctionnement Avancé du NLP

Le pipeline de traitement linguistique dans un système NLP moderne comprend :

- **Segmentation et Tokenisation** : fragmentation du flux textuel en unités atomiques (tokens) : mots, chiffres, ponctuations, phylographes, pour préserver la granularité sémantique et limiter le bruit (Manning et al., 2014).
- **Nettoyage et Normalisation** : transformations telles que mise en minuscule, suppression des stop words, élimination des caractères spéciaux, normalisation Unicode, traitement des entités nommées inline (Bird, Klein & Loper, 2009).
- **Étiquetage Morphosyntaxique (POS Tagging)** : assignation probabiliste ou via réseaux neuronaux à chaque token d'une catégorie grammaticale fine (Toutanova et al., 2003).
- **Filtrage des Stop Words** : exclusion des classes fonctionnelles à faible apport informationnel pour accroître la densité sémantique.
- **Lemmatisation et Normalisation Morphologique** : cartographie vers la forme canonique, désambiguïsation contextuelle.
- **Extraction de Caractéristiques et Vectorisation** :
 - *Analyse Syntaxique* : génération d'arbres de dépendance (Universal Dependencies) ou de structures de constituants.
 - *Analyse Sémantique* : encodage contextuel avec modèles comme BERT, RoBERTa, XLM exploitant l'attention multi-tête (Devlin et al., 2018).
 - *Analyse Pragmatique* : désambiguïsation sémantique, reconnaissance d'intentions, résolution de co-référence.

- **Pipeline d'Inférence et Post-Traitement** : production de sorties exploitables (JSON-LD, graphes RDF), analyses de confiance et explicabilité (saliency maps, LIME, SHAP) pour traçabilité scientifique (Ribeiro et al., 2016, Lundberg & Lee, 2017).

Ce pipeline constitue la colonne vertébrale des architectures NLP de pointe, permettant de transformer des corpus bruts en représentations linguistiques riches, prêtes pour la modélisation avancée et l'interprétation critique.

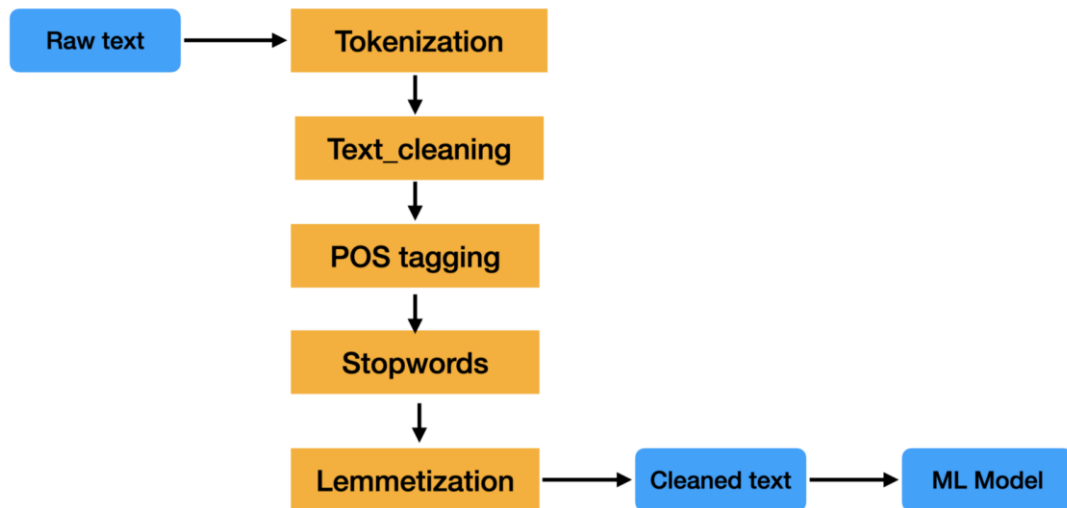


Figure A1.3 : Pipeline de prétraitement du texte pour l'entraînement d'un modèle de Machine Learning

1.5.4 Applications du NLP

Le NLP est utilisé dans une variété d'applications pratiques, notamment :

- Traduction automatique : conversion de texte d'une langue à une autre (Bahdanau et al., 2015).
- Analyse de sentiments : détermination de l'opinion ou de l'émotion exprimée dans un texte (Pang & Lee, 2008).
- Systèmes de questions-réponses : réponse automatique à des questions posées en langage naturel (Chen et al., 2017).
- Chatbots et assistants virtuels : interaction automatisée avec les utilisateurs (Adamopoulou & Moussiades, 2020).
- Résumé automatique de texte : génération de résumés concis à partir de documents longs (Nallapati et al., 2016).

- Détection de spam : identification et filtrage des messages indésirables (Sahami et al., 1998).

Ces applications démontrent la capacité du NLP à transformer les interactions homme-machine.

1.6 Calcul des embeddings et mesures de similarité

a) Génération d'embeddings

Les embeddings sont des vecteurs denses représentant des phrases, mots ou documents dans un espace sémantique. Ils sont générés à l'aide de modèles d'encodage, par exemple :

- **BERT (Transformer encoder)** : encode le texte en un vecteur de dimension D (souvent D=768 ou 1024) (Devlin et al., 2018).
- **Sentence-BERT** : encode des phrases ou paragraphes pour la recherche sémantique (Reimers & Gurevych, 2019).

Formellement, pour un texte t :

$$e_t = \text{Encoder}(t) \in \mathbb{R}^D$$

b) Calcul de la similarité cosinus

Pour comparer deux textes via leurs embeddings, on utilise généralement la similarité cosinus:

$$\text{sim}_{\cos}(a, b) = \frac{a \cdot b}{\|a\| \times \|b\|}$$

Où :

- $a, b \in \mathbb{R}^D$ sont les vecteurs d'embedding,
- $\cdot$ désigne le produit scalaire,
- $\|\cdot\|$ la norme euclidienne.

Cette mesure donne un score entre -1 et 1 (plus proche de 1 = plus similaires).

1.7 Algorithmes de recherche ANN (Approximate Nearest Neighbors)

Pour des bases de données volumineuses (milliers/millions d'embeddings), la recherche du voisin le plus proche doit être accélérée via des algorithmes spécialisés :

a) Indexation FAISS (Facebook AI Similarity Search)

Les embeddings sont stockés dans un index FAISS, qui permet de rechercher rapidement les K vecteurs les plus proches selon similarité cosinus ou euclidienne (Johnson et al., 2017).

b) Algorithme HNSW (Hierarchical Navigable Small World)

Représente les embeddings comme un graphe à petits mondes navigable, permettant une recherche efficace avec complexité logarithmique en pratique (Malkov & Yashunin, 2018).

Pseudocode simplifié (recherche KNN sur embeddings) :

```
python

def search_knn(query_embedding, index, top_k):
    # index = FAISS, HNSW, etc.
    # Recherche des top_k passages les plus similaires à la requête dans l'index
    results = index.search(query_embedding, k=top_k)

    return results # Retourne les top_k passages les plus similaires
```

1.8 Algorithmes de pipeline RAG

a) Fonction de retrieval (exemple général)

```
python

def retrieve(question, embeddings_index, encoder, top_k=5):
    q_emb = encoder(question)
    passages = search_knn(q_emb, embeddings_index, top_k)
    return passages
```

b) Fonction de reranking

Le reranker (par exemple, un cross-encoder) prend chaque couple (question, passage) et lui attribue un score de pertinence plus fin (via un modèle BERT-like) :

```
python

def rerank(question, candidate_passages, cross_encoder, top_k=3):
    scored = []
    for passage in candidate_passages:
        score = cross_encoder.score(question, passage)
        scored.append((passage, score))
    scored.sort(key=lambda x: x[1], reverse=True)
    return [p for p, _ in scored[:top_k]]
```

c) Génération de la réponse

Après récupération et reranking, les K meilleurs passages sont injectés comme contexte dans le prompt du LLM (ex : GPT, T5, Llama) :

```
python

def generate_answer(question, context_passages, llm):
    prompt = build_prompt(question, context_passages)
    response = llm.generate(prompt)
    return response
```

Le prompt est construit pour “forcer” le LLM à s’appuyer explicitement sur les passages récupérés.

1.9 Apprentissage, fine-tuning et scaling laws

a) Pré-entraînement

Le pré-entraînement des LLM consiste à apprendre à prédire des mots masqués (“masked language modeling” pour BERT) ou à compléter des séquences (“causal language modeling” pour GPT).

Formule générale pour la perte :

$$L = - \sum_{i=1}^N \log P(w_i | \text{contexte})$$

b) Fine-tuning

Après pré-entraînement, le modèle est adapté (“fine-tuned”) sur une tâche cible :

- Classification, question-réponse, résumé, etc.

- On utilise des jeux de données annotés plus petits, en mettant à jour les poids du réseau.(Devlin et al., 2018).

c) Scaling laws

Les scaling laws sont des relations empiriques montrant que l'augmentation simultanée du nombre de paramètres (PPP), de la taille des données (DDD), et du calcul (CCC) améliore régulièrement les performances des LLM.

$$Erreur \approx P^{-\alpha} + D^{-\beta} + C^{-\gamma}$$

Où :

α, β, γ sont des constantes empiriques spécifiques au modèle (Kaplan et al., 2020).

Annexe 2 : Les Systèmes Multi-Agents (SMA)

2.1. Introduction générale

Les *systèmes multi-agents* (SMA) constituent un domaine de l'**intelligence artificielle distribuée** dans lequel plusieurs agents intelligents collaborent (ou interagissent de façon compétitive) de manière décentralisée pour accomplir des tâches complexes dans des environnements dynamiques ibisc.univ-evry.fr. Plutôt que de s'appuyer sur un seul agent « tout puissant », l'IA distribuée envisage une **communauté de penseurs** : un ensemble d'entités autonomes (les agents) qui interagissent pour mener à bien une tâche et contribuent collectivement à la résolution d'un problème complexe (techniques-ingenieur.fr). Cette approche permet de **décomposer des problèmes de grande envergure** en sous-tâches traitées par différents agents, favorisant l'émergence d'une *intelligence collective* capable de résoudre des problèmes difficiles à appréhender de manière globale ou centralisée.

Un SMA se caractérise par l'absence de contrôle centralisé strict : chaque agent prend des décisions de façon locale, ce qui rend le système global plus **flexible, robuste et apte à s'adapter** aux changements de l'environnement en temps réel (fr.wikipedia.org). Ainsi, les SMA excellent dans la résolution de problèmes **distribués et dynamiques** : par exemple, en robotique d'équipe, en gestion de réseaux, en simulation de phénomènes complexes ou en coordination de systèmes répartis, ils permettent de coordonner des centaines voire des milliers d'agents autonomes pour atteindre un objectif commun (ibm.com). L'essor d'Internet et des systèmes connectés a d'ailleurs multiplié les domaines d'application des SMA, de la modélisation de sociétés (humaines ou animales) aux services web, en passant par les assistants personnels intelligents et les systèmes multi-robots (techniques-ingenieur.fr).

En synthèse, un SMA offre un **cadre architectural** où l'intelligence émerge de l'interaction d'entités multiples plutôt que d'un seul programme monolithique. Cette approche distribue le raisonnement et l'action, ce qui accroît la **scalabilité** (passage à l'échelle) et la **robustesse** du système face à l'incertitude ou aux défaillances locales. Dans la suite, nous définissons plus formellement la notion d'agent intelligent, puis décrivons le fonctionnement général d'un SMA et ses différentes architectures, avant d'examiner un cas particulier moderne : l'architecture *Retrieval-Augmented Generation* (RAG) en version classique puis sous forme multi-agent.

2.2. Définition des agents dans un système IA

Un **agent**, en IA, est généralement défini comme une entité informatique (logicielle ou physique) **située dans un environnement**, capable d'**agir de façon autonome** pour atteindre

les objectifs pour lesquels elle a été conçue (techniques-ingenieur.fr). Cette notion d'autonomie exclut la télécommande étape-par-étape par un contrôle central. Wooldridge et Jennings (1995) ont formalisé la définition d'un *agent intelligent* dit « faible » en listant quatre propriétés (fondamentales ibisc.univ-evry.fr) :

- **Autonomie** : l'agent fonctionne sans intervention directe de l'utilisateur ou d'autres processus, et contrôle dans une certaine mesure ses actions et son état interne. Il peut prendre des initiatives par lui-même (comportement proactif) et n'est pas qu'un simple répondeur.
- **Sociabilité (capacité sociale)** : l'agent est capable d'interagir avec d'autres agents (ou éventuellement des humains) via un langage de communication approprié. Cette sociabilité lui permet de coopérer, de négocier ou de coordonner ses actions au sein d'un groupe.
- **Réactivité** : l'agent perçoit son environnement (qu'il s'agisse du monde physique via des capteurs, de l'interface utilisateur, d'autres agents, d'Internet, etc.) et réagit en temps opportun aux changements qui s'y (produisent). Il est donc sensible aux stimuli externes et peut adapter son comportement en conséquence.
- **Proactivité** : au-delà de simplement réagir, un agent peut adopter un comportement orienté par des buts. Il prend des initiatives pour accomplir ses objectifs, générant des actions de son propre chef pour faire progresser sa mission. Cette qualité lui permet d'être actif et d'anticiper, plutôt que d'attendre passivement des sollicitations.

En plus de ces quatre propriétés clés (souvent citées dans la littérature anglophone, on attribue parfois aux agents d'autres caractéristiques comme la capacité d'**apprendre** ou de s'**adapter** à l'expérience, mais ces aptitudes relèvent de l'agent *fort* (intelligent au sens cognitif complet). En première approximation, tout agent possède au moins autonomie et réactivité, et les agents dits intelligents ajoutent proactivité et sociabilité (ibisc.univ-evry.fr).

On distingue également plusieurs **typologies d'agents** en fonction de leur architecture interne et de leur mode de décision :

- **Agents réactifs** : ils agissent selon le paradigme perception→action par des réflexes simples de type stimulus-réponse. Ils n'ont pas de représentation explicite du monde ni de mémoire d'états passés élaborée. Inspirés par les systèmes biologiques simples (insectes sociaux, par ex.), ces agents peuvent donner lieu à des comportements émergents intelligents via leurs interactions locales (e.g. stigmergie chez les fourmis)(techniques-ingenieur.fr). Leur architecture est généralement modulaire (subsomption) et purement réactive.

- **Agents délibératifs (ou cognitifs)** : dotés d'un modèle symbolique interne, de connaissances sur le monde et éventuellement d'états mentaux (croyances, désirs, intentions), ces agents choisissent leurs actions via un processus de raisonnement explicite (techniques-ingénieur.fr). Ils peuvent planifier des séquences d'actions, faire des inférences logiques et tenir compte des autres agents de façon rationnelle. Un exemple type est l'architecture BDI (Belief-Desire-Intention) qui modélise la prise de décision délibérative (pageperso.lis-lab.fr).

- **Agents hybrides** : ils combinent les deux approches précédentes pour tirer parti des avantages de chacune. Typiquement, un agent hybride comporte des couches réactives (pour assurer une réponse rapide aux urgences de l'environnement) et des couches délibératives (pour planifier à plus long terme et raisonner sur des objectifs complexes) (pageperso.lis-lab.fr). L'architecture subsomptive de Brooks en est un précurseur, mais on trouve de nombreux modèles couchés où le comportement final résulte de l'articulation de mécanismes réactifs et cognitifs.

Il existe bien d'autres façons de catégoriser les agents (agents mobiles vs statiques, agents logiciels vs robots physiques, agents apprentis, agents de confiance, etc (ibisc.univ-evry.fr), mais la distinction *réactif/délibératif/hybride* est particulièrement structurante en IA distribuée. En pratique, un SMA peut contenir des agents de natures variées interagissant dans un même environnement.

2.3. Le fonctionnement d'un Agent IA

Les agents d'intelligence artificielle opèrent selon un cycle composé de quatre étapes clés : perception, traitement, prise de décision, puis action. Ce processus leur permet de percevoir leur environnement, d'analyser les informations reçues, de décider des actions à entreprendre, et enfin de les exécuter afin d'atteindre leurs objectifs.

Étape 1 : Perception

L'agent recueille des informations provenant de diverses sources selon son objectif, telles que :

- Les interactions avec les utilisateurs,
- Les API extrayant des données de systèmes externes,
- Les capteurs ou journaux d'applications connectées,
- Les bases de connaissances, comme des inventaires ou des politiques internes.

Étape 2 : Traitement

Après collecte, l'agent doit interpréter et comprendre ces données. Il utilise des techniques variées telles que le traitement du langage naturel (NLP), l'analyse de données structurées ou de flux en temps réel. Pour extraire des connaissances spécifiques, il peut recourir à la génération augmentée par récupération (RAG).

Étape 3 : Prise de décision

L'agent applique une logique adaptée à sa conception :

- Des règles métier personnalisées pour qualifier des prospects ou gérer des workflows,
- Des modèles prédictifs issus de l'apprentissage automatique ou du renforcement, par exemple pour détecter des fraudes. L'explicabilité joue un rôle important, car l'agent doit souvent justifier ses décisions.

Étape 4 : Action

Enfin, l'agent agit selon les décisions prises. Ses actions peuvent inclure :

- La production d'une réponse textuelle informative,
- L'appel d'API pour récupérer des données en temps réel ou déclencher des processus,
- L'exécution directe d'opérations comme la modification des prix, la planification d'appels, le réacheminement logistique ou la mise à jour de systèmes selon des règles définies.

Certains agents automatisent des flux de travail complexes dans des applications externes (CRM, gestion client, remboursements, etc.).

À chaque cycle, l'agent veille à respecter les processus décisionnels établis et apprend souvent des résultats obtenus afin d'améliorer ses performances (botpress.com)



2.4. Quelle est la différence entre un agent d'IA et un chatbot d'IA ?

Il est courant que les termes « agent d'intelligence artificielle » et « chatbot d'intelligence artificielle » soient utilisés de manière interchangeable, ce qui est compréhensible compte tenu des nombreuses similitudes entre ces deux concepts.

En effet, les deux exploitent généralement des techniques de traitement du langage naturel (NLP) pour comprendre les entrées linguistiques, sont souvent basés sur des modèles de langage de grande taille (LLMs) et peuvent être connectés à des systèmes externes pour accéder à des données ou exécuter des actions.

Cependant, les agents d'intelligence artificielle se distinguent des chatbots sur plusieurs points clés, notamment dans leur niveau d'autonomie et la complexité des tâches qu'ils accomplissent. Voici un tableau synthétique mettant en évidence ces différences :

Chatbots d'IA	Agents d'intelligence artificielle
Ont pour objectif d'engager des conversations afin d'aider les utilisateurs à réaliser des tâches spécifiques.	Automatisent des processus complexes, prennent des décisions et exécutent des tâches en plusieurs étapes.
Répondent de manière réactive selon les données fournies par l'utilisateur, en suivant une logique prédéfinie ou des modèles appris.	Raisonnent, planifient et agissent de manière proactive et autonome, sans intervention directe de l'utilisateur.
Se présentent généralement sous forme d'interfaces conversationnelles sur des sites web, applications ou plateformes de messagerie.	Peuvent fonctionner en arrière-plan comme des systèmes autonomes ou disposer d'une interface conversationnelle.

Ces distinctions sont déterminantes pour choisir la solution la mieux adaptée aux besoins d'une organisation. Par exemple, un chatbot peut répondre aux questions des clients, suggérer des produits et faciliter des transactions simples. En revanche, un agent d'IA est capable d'analyser des comportements, prédire quels clients sont les plus susceptibles de réaliser des achats supplémentaires, et engager une interaction personnalisée au moment opportun via des canaux comme Facebook Messenger, tout en réalisant les fonctions classiques d'un chatbot.

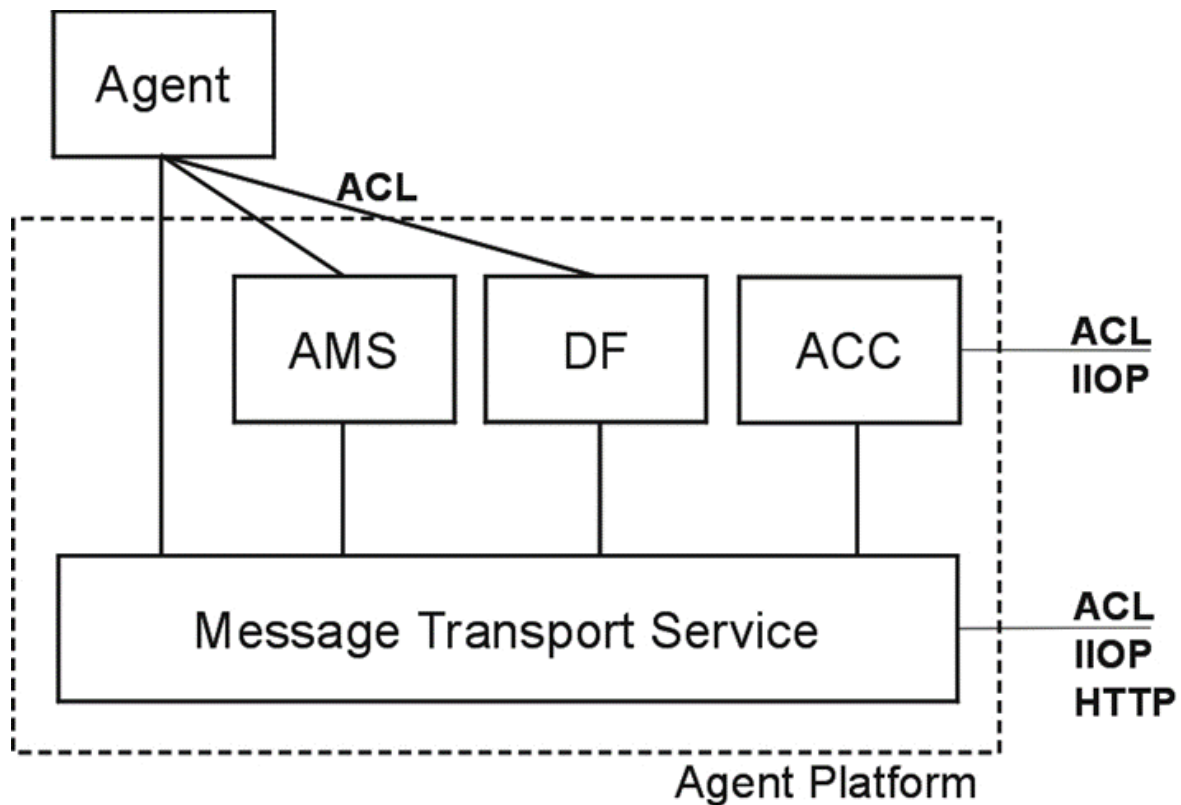
Cette capacité accrue fait des agents d'intelligence artificielle des outils puissants et polyvalents pour optimiser les processus commerciaux et améliorer l'expérience client.

2.5. Fonctionnement général d'un SMA

Dans un SMA, les agents interagissent entre eux pour **coordonner** leurs activités et atteindre des objectifs potentiellement partagés. Cette interaction repose d'une part sur des mécanismes de **communication inter-agents**, et d'autre part sur des techniques de **coordination** et d'**organisation** permettant d'éviter le chaos et les conflits.

Communication inter-agents : Pour que les agents collaborent efficacement, ils doivent échanger des informations de manière structurée. Contrairement à de simples appels de fonctions, la communication entre agents requiert des **langages de communication d'agents (ACL)** dédiés, indépendants des plates-formes des agents participants (pageperso.lis-lab.fr). Ces langages s'appuient sur la théorie des *actes de langage* en philosophie : chaque message agent est vu comme un acte (requête, promesse, informatif, etc.) ayant une **intention illocutoire**

(par ex. *demandeur une information, proposer un plan, accepter un contrat*). Des standards ont émergé, notamment **KQML** (Knowledge Query and Manipulation Language) et le langage **ACL de la FIPA**, qui définissent des *performatifs* (types de messages) standard pour les interactions agent (pageperso.lis-lab.fr).



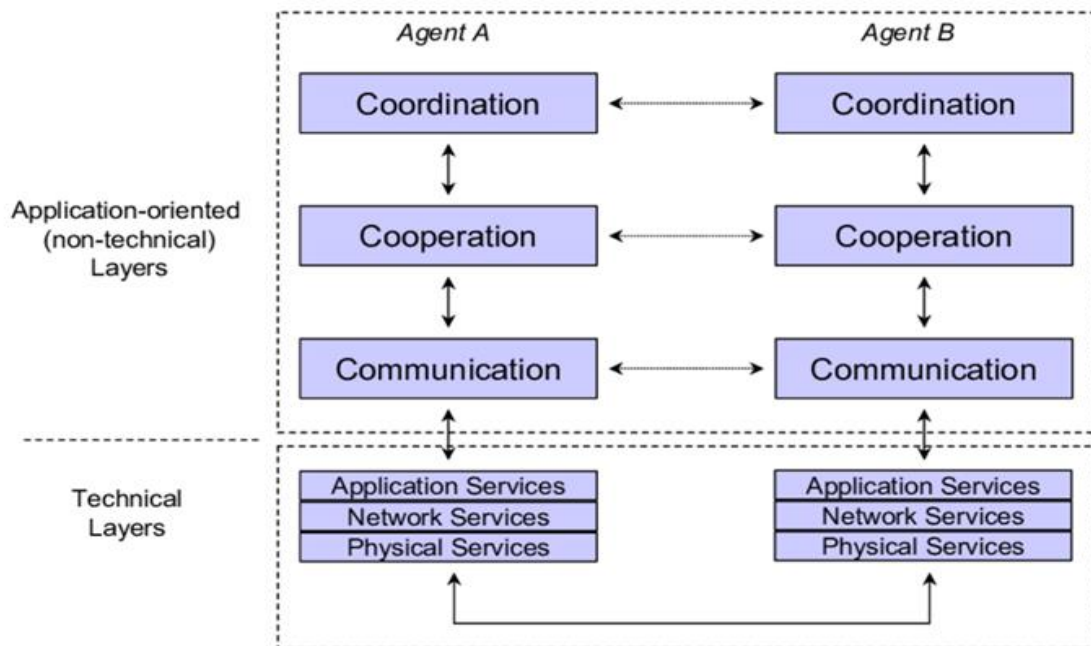
researchGate (Architecture d'une plateforme d'agents FIPA (AMS, DF, ACC et service de transport de messages))

En plus du langage, des **protocoles de communication** régissent les séquences d'échanges entre agents. Par exemple, le *protocole Contract Net* est un schéma classique de négociation dans lequel un agent « annonceur » propose une tâche et des agents « soumissionnaires » répondent avec des offres, permettant d'allouer la tâche au meilleur candidat. D'autres protocoles standard existent (requête-réponse, enchères anglaises ou hollandaises, négociation itérative, etc (cs.upc.edu/pageperso.lis-lab.fr)). Par ailleurs, pour que les messages aient un sens commun, les agents d'un SMA doivent partager des **ontologies** – c'est-à-dire des vocabulaires et structures de connaissances communes – qui permettent d'interpréter sans ambiguïté le contenu sémantique des messages (smythos.com). L'interopérabilité entre agents hétérogènes (potentiellement développés par différents acteurs) dépend largement de ces ontologies partagées et du respect de protocoles de communication normalisés.

Coordination et coopération : Communiquer ne suffit pas, encore faut-il que les agents parviennent à coordonner leurs actions pour éviter incohérences et compétitions stériles. La **coordination** renvoie aux mécanismes par lesquels les agents ajustent leur comportement les uns par rapport aux autres pour atteindre un arrangement efficace de leurs actions. Dans un SMA coopératif, cela peut impliquer de la **planification distribuée**, où un plan global est réparti entre les agents ou émergent de la mise en commun de plans partiels. Par exemple, des *agents planificateurs* peuvent négocier la répartition de tâches ou synchroniser leurs agendas pour accomplir une mission conjointe.

Un aspect crucial est la **négociation** entre agents lorsque leurs intérêts ou objectifs ne sont pas strictement alignés. Des techniques issues de la théorie des jeux et de l'économie informatique (enchères, marchés d'agents, négociation bilatérale automatisée) permettent aux agents d'échanger offres et contre-offres jusqu'à trouver un compromis moveworks.com/moveworks.com. Dans les SMA compétitifs (par ex. agents représentant des entreprises en concurrence), ces mécanismes sont essentiels. Même en contexte coopératif, la négociation aide à résoudre la **résolution de conflits** potentiels – par exemple si deux agents veulent accéder à une ressource limitée en même temps, ou si leurs sous-objectifs locaux entrent en conflit.

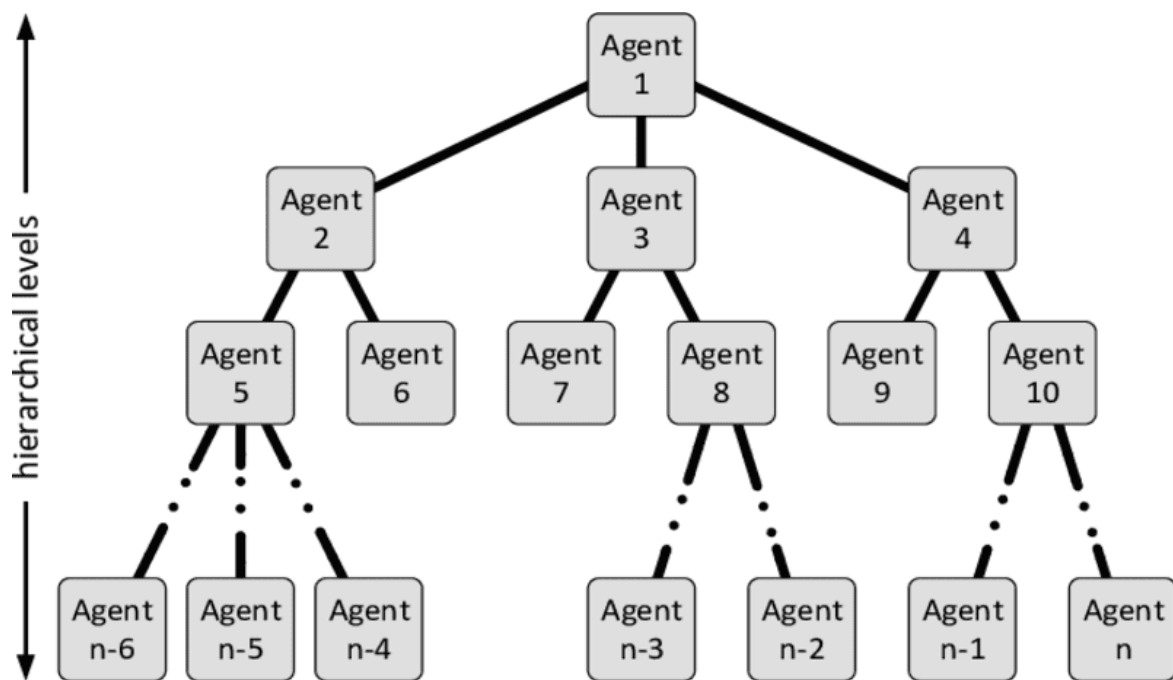
Pour faciliter la coordination, les agents peuvent également recourir à des **protocoles de coordination** spécifiques (différents des protocoles de communication de bas niveau). Il s'agit par exemple d'algorithmes de consensus distribués (se mettre d'accord sur une valeur), d'élections de leader (pour désigner temporairement un agent coordinateur), de suivi de contrats, etc. L'objectif est de garantir qu'un **comportement global cohérent** émerge, malgré la décision répartie. Certaines approches s'inspirent de la biologie (p. ex. la *stigmergie* utilise l'environnement comme mémoire partagée pour coordonner indirectement des agents réactifs, à l'image des fourmis déposant des phéromones).



researchGate (Architecture en couches des agents FIPA : Communication, Coopération et Coordination)

Structures organisationnelles : Au-delà de la coordination ad hoc, on peut concevoir un SMA en définissant une **organisation** explicite des agents. Les structures organisationnelles décrivent qui interagit avec qui, et éventuellement quels *rôles* ou positions occupent les agents dans la société artificielle. Plusieurs modèles classiques existent :

- **Hiérarchie :** les agents sont organisés en structure arborescente (types *chef-sous-chef*). Chaque nœud (agent) supervise ses « agents-fils » et possède une **autorité** sur eux fr.wikipedia.org. Ce modèle top-down facilite la décomposition des tâches et l'application de directives globales, mais introduit un point de défaillance (le sommet) et réduit l'autonomie locale.



FPT AI (Structure hiérarchique en arbre des agents)

- **Holarchie** : proche de la hiérarchie mais sans lien d'autorité formel. Ici, un agent de plus haut niveau est littéralement *constitué* de ses sous-agents (relation partie-tout). Par exemple, une *ville* agent est composée d'agents *bâtiments* ; une région est composée de villes, etc. L'organisation est fractale et orientée vers l'émergence de propriétés globales sans commande centralisée stricte (principe des *holons*).

- **Coalition** : il s'agit d'une alliance *temporaire* d'agents qui unissent leurs efforts parce qu'ils y trouvent chacun un intérêt individuel supérieur à ce qu'ils auraient seuls. Chaque agent d'une coalition reste égoïste, mais la coopération ponctuelle sert leurs objectifs mutuels (gain mutuel). Une coalition se forme et se défait opportunément (exemple : six agents s'associent pour acheter un lot de biens moins cher par économie d'échelle).

- **Équipe** : à la différence d'une coalition, une équipe d'agents vise un **objectif commun** et chaque membre cherche à maximiser le bénéfice **collectif** plutôt que son gain individuel (fr.wikipedia.org). Cela requiert un degré de confiance et de planification conjointe plus poussé. On parle parfois d'« agents coopératifs » au sein d'une équipe au sens fort du terme.

- **Société** : une société d'agents est un groupement plus large et hétérogène où les agents peuvent avoir des objectifs variés et des niveaux de rationalité différents, mais partagent des **normes communes** qui régissent leurs interactions (fr.wikipedia.org). C'est un cadre général

qui englobe des notions de régulation (lois, conventions) assurant un ordre social dans un SMA ouvert.

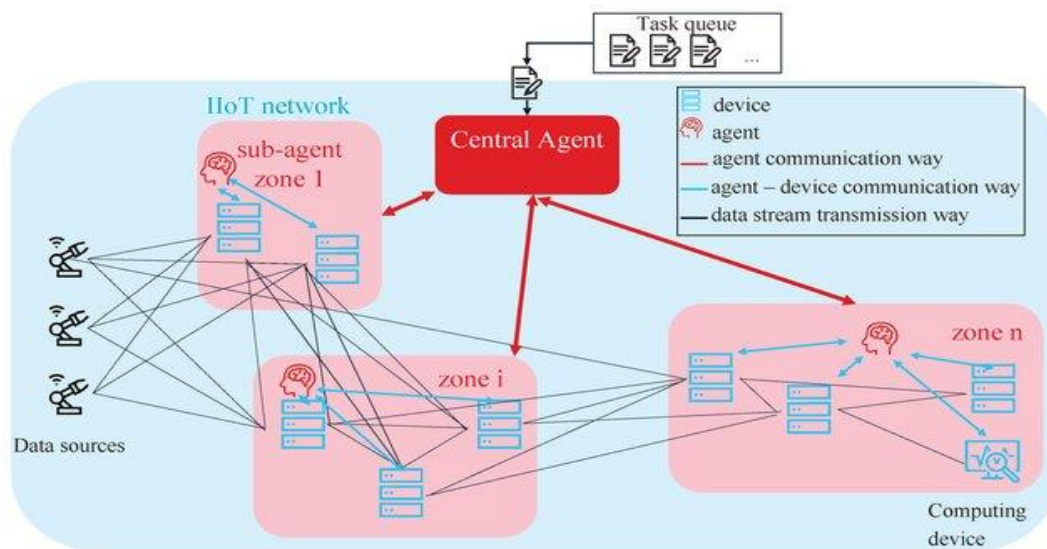
- *(D'autres structures existent, par ex. **congrégation** – groupe permanent poursuivant plusieurs objectifs, dont la composition peut varier au fil du temps, ou **fédération** – groupes d'agents représentés chacun par un délégué auquel ils délèguent une part de leur autonomie pour interfacer avec d'autres groupes. On trouve également des organisations matricielles ou des combinaisons hybrides de ces modèles (fr.wikipedia.org).*

L'**auto-organisation** est un cas particulier où la structure organisationnelle n'est pas imposée a priori, mais émerge des interactions locales entre agents. Les SMA purement réactifs tirent parti de ce phénomène : par exemple, sans chef explicite, les agents fourmis finissent par s'auto-organiser pour relier une colonie à une source de nourriture via une piste chimique optimale. Les chercheurs en SMA étudient ces mécanismes d'auto-organisation afin de construire des systèmes robustes, adaptatifs et évolutifs sans supervision centrale (ibisc.univ-evry.fr). Les architectures dites *auto-organisatrices* laissent ainsi les agents former spontanément des hiérarchies, des équipes ou d'autres agencements en fonction des besoins, ce qui est un atout pour l'adaptabilité du système.

2.6. Architectures des systèmes multi-agents

On qualifie l'architecture d'un SMA selon la manière dont sont structurés et orchestrés les agents du système. Deux extrêmes polaires sont les approches **centralisée (orchestrée)** vs **décentralisée**, avec des solutions **hybrides** intermédiaires. Chacune présente des avantages et limites en termes de performance, de robustesse et de complexité.

- **Architecture centralisée** : Un SMA centralisé comporte un ou plusieurs agents ayant un rôle *décisionnel central* ou de coordination globale. Par exemple, un agent *chef* peut assigner les tâches aux autres, ou une mémoire partagée (tableau noir) peut contenir l'état global sur lequel tous se basent (techniques-ingenieur.fr).

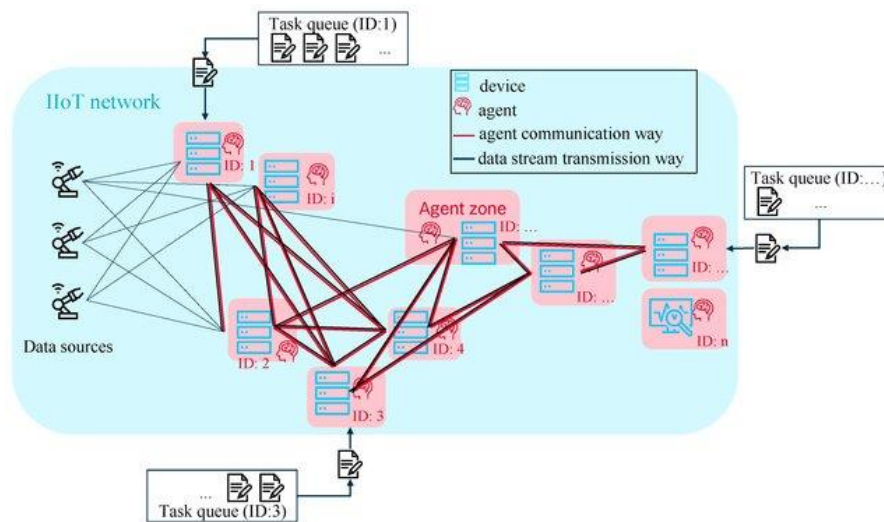


ResearchGate (Schematic view on the architecture of the centralized multi-agent approach)

L'avantage d'une telle architecture est de faciliter la cohérence globale et la prise de décision optimisée à l'échelle du système (le centre ayant une vue d'ensemble). Elle peut aussi simplifier la conception en réduisant le problème à un schéma maître-esclaves classique. Cependant, les inconvénients incluent une **vulnérabilité accrue** (si l'agent central tombe en panne, le SMA entier peut échouer) et un risque de **goulot d'étranglement** lorsque le nombre d'agents croît (l'agent central peut être saturé de communications ou de décisions à prendre). De plus, cela contrecarre en partie l'autonomie des agents et peut limiter la **scalabilité** (passage à grande échelle) du système.

- **Architecture décentralisée** : Dans un SMA complètement décentralisé, il n'y a *aucun contrôleur global* – la coordination émerge uniquement des interactions pair-à-pair entre agents. Chaque agent prend ses décisions localement en fonction de ce qu'il perçoit et des messages reçus, et la synchronisation se fait via les protocoles d'interaction. Ce modèle est **robuste** (pas de point unique de défaillance), intrinsèquement évolutif (on peut souvent ajouter ou retirer des agents sans reconfigurer un centre) et correspond bien aux systèmes naturels distribués. En revanche, il pose des défis en termes de **complexité** : concevoir des mécanismes locaux garantissant un comportement global souhaité relève parfois du casse-tête (problème de l'assurance que les interactions mènent à la solution). La **gestion des conflits** peut être plus compliquée sans arbitre central, et le temps pour qu'un consensus ou une coordination se forme peut augmenter avec le nombre d'agents. Néanmoins, de nombreux SMA réussis (p. ex. les

protocoles d'agent dans les réseaux peer-to-peer, ou les essaims de robots) adoptent cette décentralisation pour bénéficier de sa résilience et de son adaptabilité.



researchGate (Schematic view on the fully decentralized multi-agent architecture.

- **Architecture hybride** : Entre ces deux pôles, on trouve des architectures combinant centralisation partielle et décentralisation. Un schéma fréquent est l'architecture **hiérarchique** : à un niveau supérieur, quelques agents de coordination gèrent chacun un sous-groupe d'agents, permettant une décomposition modulaire du problème, tandis qu'à l'intérieur de chaque sous-groupe les agents interagissent de façon plus autonome. Ce modèle hybride **réduit la charge** sur un seul centre en la répartissant, tout en conservant une supervision globale à gros grain. Par exemple, dans une flotte de robots, on pourrait avoir un robot leader par escouade qui coordonne localement ses quelques coéquipiers, plutôt qu'un seul leader pour toute la flotte (fr.wikipedia.org). D'autres approches hybrides consistent à introduire un **agent intermédiaire (dispatcheur)** qui récolte les informations de tous puis redistribue des commandes (architecture en étoile), ce qui diminue le nombre de messages échangés par rapport à une diffusion totalement décentralisée tout en évitant les problèmes d'un centre unique researchgate.net. L'approche hybride cherche le **compromis** : elle améliore la **scalabilité** par rapport à un système strictement centralisé, et elle simplifie la coordination par rapport à du 100% pair-à-pair. Sa difficulté réside dans le choix judicieux de ce qui est centralisé (et à quel niveau) vs ce qui est laissé distribué – un choix souvent spécifique au domaine d'application.

2.7. Autonomie et contraintes dans les agents distribués

Un agent dit 100 % autonome est capable de percevoir, raisonner et agir sans intervention humaine directe ni contrôle centralisé. Cependant, dans le contexte d'un système distribué, cette

autonomie reste encadrée par des règles, des protocoles et des contraintes fonctionnelles strictes qui garantissent la cohérence globale du système.

Autrement dit, même si un agent agit de façon indépendante, il ne peut pas se « rebeller » contre les objectifs, règles ou mécanismes de coordination définis dans le système. Cette limitation découle notamment de :

- Les protocoles de communication et de coordination qui imposent des contraintes comportementales, assurant la collaboration et évitant les conflits destructeurs.
- Les règles métier et politiques intégrées dans la conception des agents, qui restreignent leurs choix d'action à un cadre acceptable.
- La nécessité d'assurer la stabilité et la fiabilité globale du système distribué, ce qui empêche tout comportement imprévu ou malveillant pouvant compromettre l'ensemble.

En résumé, l'autonomie d'un agent dans un système distribué est relative : elle garantit l'indépendance opérationnelle, mais sous des limites fonctionnelles précises qui empêchent tout comportement anarchique ou rebelle. Ceci est essentiel pour maintenir la confiance, la sécurité et la robustesse dans les environnements multi-agents complexes.

Annexe 3 : Interface utilisateur Streamlit et aspects pratiques

3.1 Aperçu de l'interface

- **Captures d'écran** annotées montrant les principales fonctionnalités :
 - Zone de saisie (question)
 - Import de documents
 - Affichage de l'historique
 - Boutons de feedback ("Utile" / "Pas utile")
 - Affichage des réponses et des sources
- **Légendes** expliquant brièvement chaque composant de l'interface.

3.2 Workflow utilisateur

- **Scénario d'utilisation type** (ex. : de l'import d'un document à la génération de la réponse)
- Diagramme de séquence (optionnel) ou schéma du parcours utilisateur.

3.3 Extrait(s) de code Streamlit

- **Code simplifié ou snippets-clés** (ex. : gestion de l'upload, affichage dynamique des réponses, gestion du feedback utilisateur, etc.)
- Commentaires expliquant la logique de chaque partie.

Exemple :

```
python

import streamlit as st

# Import du document
uploaded_file = st.file_uploader("Importer un document", type=["pdf", "txt", "docx"])

# Saisie de la question utilisateur
question = st.text_input("Posez votre question...")

# Appel de la fonction assistant (pipeline)
if st.button("Envoyer"):
    response = ask_assistant(question, uploaded_file)
    st.write(response)
    st.button("Utile")
    st.button("Pas utile")
```

3.4 Personnalisation et limites de l'interface

- Points forts (simplicité, traçabilité, ergonomie...)
- Limites identifiées ou axes d'amélioration (ex : support de nouveaux formats, anonymisation, sécurité...)

3.5 Reproductibilité et déploiement

- **Instructions d'installation rapide** (requirements.txt, dépendances principales)
- Commande de lancement Streamlit (`streamlit run app.py`)
- Spécificités pour l'intégration locale (ex. : modèles, index, API internes)