RESUME

Ce projet porte sur la conception et l'implémentation d'un chatbot juridique spécialisé dans le droit du travail sénégalais. Son objectif est de fournir aux travailleurs, employeurs et étudiants un outil technologique local, fiable et accessible, capable de répondre aux questions juridiques tout en citant les sources officielles.

Pour atteindre cet objectif, une approche RAG (Retrieval-Augmented Generation) a été mise en œuvre. La base de connaissances, composée du Code du travail sénégalais et du Manuel du travailleur, a été prétraitée puis découpée en passages (chunks). Ces données ont été indexées dans Milvus, une base vectorielle open source, en utilisant une collection hybride combinant :

- des vecteurs denses, pour la représentation sémantique des textes,
- et des vecteurs clairsemés (sparse), pour la représentation lexicale.

Les embeddings ont été générés grâce à BGEM3EmbeddingFunction, permettant d'exploiter simultanément les deux types de représentations. Les résultats de recherche sont ensuite fusionnés à l'aide de la méthode Reciprocal Rank Fusion (RRF), garantissant une meilleure pertinence des extraits sélectionnés.

La partie génération repose sur un LLM exécuté localement via Ollama. Plusieurs modèles ont été testés, notamment Qwen2.5 et Phi-3.5 Mini, avec une préférence pour Qwen2.5 en raison des contraintes matérielles. Un prompt juridique spécialisé a été conçu afin de guider le modèle vers des réponses précises, contextualisées et sourcées, tout en limitant les hors-sujets.

L'interface utilisateur a été développée avec Streamlit et propose deux modes d'interaction :

- la saisie clavier, de type chatbot classique,
- l'entrée vocale push-to-talk, intégrant la reconnaissance vocale Vosk STT, qui permet de dicter une question et d'obtenir automatiquement sa transcription avant envoi.

Une tentative d'intégration de la lecture vocale via Piper TTS a également été explorée, mais mise en pause pour des raisons techniques. Toutefois, l'architecture du système reste ouverte à cette extension future.

En définitive, ce projet démontre la faisabilité d'un assistant juridique local, fiable et multimodal, combinant rigueur technique (Milvus, embeddings hybrides, RRF), efficacité

pratique (LLM exécuté en local) et ergonomie (interface clavier + voix). Il constitue une avancée concrète dans l'amélioration de l'accessibilité au droit du travail sénégalais grâce aux technologies de l'intelligence artificielle.

LISTE DES SIGLES, ACRONYMES ET ABRECIATIONS

- RAG : Retrieval-Augmented Generation
- LLM : Large Language Model
- NLP : Natural Language Processing
- OCR : Optical Character Recognition
- PDF : Portable Document Format
- RRF : Reciprocal Rank Fusion
- CPU : Central Processing Unit
- API : Application Programming Interface
- GPU : Graphics Processing Unit

LISTES DES FIGURES

SOMMAIRE

INTRODUCTION

Le droit du travail occupe une place essentielle dans l'organisation des relations professionnelles au Sénégal. Il constitue l'ensemble des règles juridiques qui encadrent les rapports entre employeurs et travailleurs, en garantissant à la fois la protection des salariés et la régulation des pratiques patronales. Cependant, l'accès effectif à cette information juridique demeure un défi : la majorité des travailleurs, notamment dans le secteur informel, ne disposent pas des connaissances nécessaires pour comprendre et interpréter les dispositions légales qui les concernent.

Dans ce contexte, le numérique s'impose aujourd'hui comme un levier puissant d'accessibilité à l'information juridique. Les technologies de l'intelligence artificielle, et en particulier les systèmes de question-réponse automatisés, permettent de vulgariser le droit et de le rendre directement exploitable par les citoyens. Les chatbots, en combinant des bases de connaissances spécialisées et des modèles de génération de langage, offrent une solution interactive, rapide et adaptée aux besoins des utilisateurs.

C'est dans cette perspective que s'inscrit le présent projet, qui vise la conception et l'implémentation d'un chatbot juridique basé sur le droit du travail sénégalais. L'objectif principal est de développer un assistant capable de répondre de manière précise aux questions des utilisateurs, en citant les textes officiels du Code du travail et du Manuel du travailleur. Pour ce faire, une approche RAG (Retrieval-Augmented Generation) hybride a été adoptée, combinant la recherche dense et la recherche lexicale dans une base vectorielle Milvus, puis la génération de réponses avec un modèle de langage exécuté localement.

Les missions réalisées se sont articulées autour de plusieurs étapes clés : la préparation et l'indexation des données juridiques, la mise en place d'une recherche hybride dense-sparse avec Milvus, l'intégration d'un LLM local via Ollama pour la génération de réponses, et enfin le développement d'une interface utilisateur avec Streamlit. Celle-ci offre une double modalité d'interaction : la saisie par clavier et la reconnaissance vocale grâce au moteur Vosk, renforçant ainsi l'accessibilité du système.

Ce rapport présentera dans un premier temps le contexte théorique et technologique du projet, avant de détailler la méthodologie et l'implémentation technique mise en œuvre. Enfin, une dernière partie sera consacrée à l'analyse des résultats obtenus, aux limites identifiées et aux perspectives d'évolution de la solution proposée.

CHAPITRE I : État de l'art

A. Chatbots et Traitement Automatique du Langage Naturel (NLP)

1. Définition et évolution des chatbots

Les chatbots, également appelés agents conversationnels, sont des systèmes capables d'interagir avec des utilisateurs en langage naturel, de manière écrite ou orale. Leur histoire remonte aux années 1960 avec le développement d'ELIZA, un programme conçu par Joseph Weizenbaum qui simulait un psychothérapeute en reformulant les phrases de l'utilisateur. Bien que rudimentaire, ELIZA a marqué un tournant en démontrant qu'un ordinateur pouvait donner l'illusion de comprendre le langage humain.

Au fil des décennies, les chatbots ont connu plusieurs évolutions. Les premiers étaient basés sur des règles prédéfinies et des correspondances de motifs (*pattern matching*), ce qui les rendait très limités. Ils ne pouvaient répondre qu'à des requêtes spécifiques prévues à l'avance. Avec l'essor de l'intelligence artificielle et du machine learning, les chatbots se sont transformés en systèmes capables d'apprendre à partir de données, d'améliorer leur compréhension et d'adapter leurs réponses.

Aujourd'hui, les chatbots modernes s'appuient sur le traitement automatique du langage naturel (NLP) et, plus récemment, sur les modèles de langage de grande taille (LLMs) pour fournir des réponses plus précises, nuancées et contextuelles.

2. Le NLP comme moteur des chatbots modernes

Le NLP constitue le cœur technologique des chatbots intelligents. Il englobe un ensemble de techniques permettant aux machines de comprendre, de représenter et de générer du langage humain. Les étapes classiques du NLP incluent la tokenisation (découpage du texte en unités), la lemmatisation, l'analyse syntaxique, mais surtout la représentation vectorielle des mots et phrases.

Pendant longtemps, les représentations linguistiques étaient basées sur des modèles statistiques simples comme le TF-IDF [1]ou les sacs de mots (*bag-of-words*), qui ne prenaient pas en compte le contexte. Des progrès significatifs ont ensuite été réalisés avec des modèles

[1] Le TF-IDF (Term Frequency – Inverse Document Frequency) est une mesure statistique qui évalue l'importance d'un terme dans un document relativement à un corpus.

d'embeddings continus tels que Word2Vec [2]et GloVe[3], qui projetaient les mots dans un espace vectoriel continu où la proximité entre vecteurs reflète des relations sémantiques.

Cependant, ces modèles restaient limités car ils ne tenaient pas compte du contexte dynamique d'un mot. L'émergence des modèles transformers, notamment BERT (2018) et GPT (2019), a révolutionné le NLP. Ces architectures permettent de capturer le sens d'un mot en fonction de son contexte, ouvrant la voie à des performances inédites dans la compréhension et la génération de texte.

Dans le cadre des chatbots, ces avancées offrent une compréhension fine des intentions des utilisateurs, une capacité de raisonnement contextualisé et une génération de réponses plus naturelles.

3. Applications et limites des chatbots

Les chatbots sont aujourd'hui utilisés dans une variété de secteurs. Dans le commerce électronique, ils assurent le service client, aident à la recherche de produits et facilitent les transactions. Dans le domaine de la santé, ils fournissent une assistance médicale de premier niveau, tandis qu'en éducation, ils accompagnent les apprenants avec des explications personnalisées.

Dans le domaine juridique, plusieurs chatbots émergent, comme DoNotPay, qui propose des services d'assistance pour contester des contraventions ou rédiger des documents juridiques. Toutefois, la plupart de ces solutions sont développées en anglais et restent peu adaptées aux systèmes juridiques africains, notamment au Sénégal. De plus, les chatbots basés uniquement sur des LLMs posent des problèmes de fiabilité, car ils peuvent inventer des réponses ou produire des informations sans fondement dans les textes de loi.

Ainsi, le défi central pour un chatbot juridique fiable réside dans sa capacité à fournir des réponses précises, adossées à des sources légales authentiques et vérifiables. C'est dans ce contexte que l'approche RAG apparaît comme une solution particulièrement pertinente.

[2] Word2Vec est un modèle de représentation vectorielle qui apprend les relations sémantiques entre mots en analysant leurs contextes d'apparition.
[3] GloVe (Global Vectors) est une méthode de représentation vectorielle qui capture les cooccurrences globales de mots dans un corpus pour modéliser leurs relations sémantiques.

B. L'approche RAG (Retrieval-Augmented Generation)

1. Limites des modèles de langage seuls

Les modèles de langage de grande taille (LLMs), tels que GPT, Mistral ou LLaMA, sont capables de générer du texte fluide et cohérent. Toutefois, leur fonctionnement repose sur l'apprentissage statistique à partir de larges corpus. Cela entraîne deux limites majeures :

- Ils ne disposent pas d'une mémoire actualisée : un modèle entraîné en 2023 ne connaît pas les textes de loi adoptés en 2024, sauf mise à jour.
- Ils souffrent du problème des hallucinations : ils peuvent inventer des références, des articles ou des jurisprudences inexistants, ce qui est particulièrement problématique dans un domaine sensible comme le droit.

Ces limites rendent l'utilisation directe des LLMs risquée pour des applications juridiques.

2. Principe du RAG

L'approche Retrieval-Augmented Generation (RAG) a été proposée pour pallier ces faiblesses. Elle combine deux étapes complémentaires :

- Retrieval (recherche d'information) : à partir d'une question utilisateur, le système recherche les passages les plus pertinents dans une base documentaire (par exemple, le Code du travail sénégalais ou des manuels explicatifs). Cette recherche repose sur des techniques de dense retrieval (via des embeddings vectoriels) et/ou de sparse retrieval (via des méthodes lexicales comme BM25[4]).
- Augmented Generation (génération augmentée) : les passages trouvés sont injectés dans le prompt du modèle de langage, qui génère alors une réponse contextualisée et adossée à ces extraits.

Ainsi, le modèle n'invente pas une réponse : il s'appuie sur des documents vérifiables, ce qui améliore à la fois la précision et la fiabilité des réponses produites.

3. Variantes du RAG

- Dense retrieval : utilise des représentations vectorielles continues issues de modèles comme BERT, Sentence-BERT ou BGE-M3. Ces méthodes capturent la similarité

[4] BM25 : algorithme de recherche textuelle basé sur la pondération des termes, souvent utilisé comme évolution du TF-IDF pour mesurer la pertinence entre une requête et un document.

sémantique et permettent de retrouver des passages pertinents même si les mots ne sont pas identiques.

- Sparse retrieval : repose sur des méthodes lexicales classiques (TF-IDF, BM25). Ici, la pertinence est basée sur la fréquence et la correspondance exacte des mots.
- Hybrid retrieval : combine dense et sparse retrieval. Cette méthode exploite la complémentarité entre recherche sémantique et recherche lexicale, et permet d'obtenir des résultats plus robustes.

Dans le cadre de ce projet, l'hybrid retrieval avec fusion RRF (Reciprocal Rank Fusion) a été privilégié, car il garantit un équilibre entre pertinence sémantique et rigueur lexicale.

4. Avantages du RAG dans le domaine juridique

Le recours au RAG est particulièrement adapté au secteur juridique, où la précision est cruciale. Ses principaux atouts sont :

- Fiabilité : les réponses sont ancrées dans des extraits authentiques (articles de loi, manuels).
- Traçabilité : le chatbot peut fournir les citations des sources consultées, renforçant la confiance de l'utilisateur.
- Flexibilité : les bases documentaires peuvent être mises à jour (ajout d'articles, nouvelles lois) sans nécessiter un réentraînement complet du modèle.

Ainsi, le RAG constitue une avancée déterminante pour la conception d'un chatbot juridique centré sur le droit du travail sénégalais.

Ce chapitre a permis de poser les bases conceptuelles du projet. Nous avons d'abord présenté l'évolution des chatbots et le rôle central du NLP dans leur développement, en soulignant leurs applications et leurs limites, notamment dans le domaine juridique. Nous avons ensuite introduit l'approche RAG, qui constitue une réponse aux limites des LLMs en intégrant une étape de recherche d'information avant la génération de texte. Cette méthodologie, en particulier dans sa variante hybride, offre une meilleure précision et une plus grande fiabilité, ce qui en fait un choix pertinent pour un chatbot juridique.

CHAPITRE II : Cahier des charges

1. Objectifs du projet

L'objectif général de ce projet est de concevoir et d'implémenter un chatbot juridique spécialisé dans le droit du travail sénégalais, afin de faciliter l'accès à l'information juridique pour un public non expert. Le système vise à réduire la complexité souvent associée à la lecture et à l'interprétation des textes législatifs, tout en proposant une expérience utilisateur fluide et intuitive.

Plus spécifiquement, plusieurs objectifs opérationnels ont été définis :

- Développer une architecture RAG (Retrieval-Augmented Generation) permettant de combiner une recherche documentaire efficace avec la puissance des modèles de génération.
- Mettre en place une recherche hybride (dense et sparse) pour assurer une meilleure couverture des documents juridiques et améliorer la pertinence des résultats.
- Assurer la fiabilité et la transparence des réponses générées par le chatbot, notamment par l'intégration d'un système de citations automatiques des articles du Code du travail ou des sections de documents de référence.
- Proposer une interface multimodale intégrant à la fois la saisie clavier et la commande vocale via la reconnaissance automatique de la parole (STT).
- Concevoir un système totalement exécutable en local, en exploitant uniquement des outils et bibliothèques open source, afin de garantir la confidentialité des données et l'accessibilité du projet même en contexte de ressources limitées.

Ces objectifs traduisent une volonté de concilier innovation technologique et utilité sociale, en rendant le droit du travail plus compréhensible et accessible.

2. Contraintes et périmètre

Le projet est soumis à un ensemble de contraintes qui orientent sa conception et sa réalisation.

2.1. Contraintes techniques

- Exécution en local uniquement : le système doit fonctionner sans dépendance à des services cloud afin de préserver la confidentialité des données et d'éviter des coûts liés à des solutions propriétaires.
- Ressources matérielles limitées : le projet est développé sur une machine disposant uniquement d'un processeur (CPU) avec 8 Go de RAM. Cela exclut l'utilisation de modèles trop volumineux et impose un choix raisonné des outils et modèles employés.
- Technologies open source : seuls des frameworks, bibliothèques et modèles libres sont utilisés (Ollama, Vosk, Piper, Streamlit, Milvus, etc.).

2.1. Contraintes fonctionnelles

- Langue unique : le chatbot est exclusivement dédié à la langue française, afin de correspondre aux textes législatifs et au public ciblé.
- Spécialisation thématique : le domaine de compétence est volontairement restreint au droit du travail sénégalais afin de garantir la pertinence et la fiabilité des réponses.
- Expérience utilisateur simple : l'interface doit rester accessible aux non spécialistes de l'informatique ou du droit.

2.2. Périmètre

Le système couvre :

- Le Code du travail sénégalais et certains documents complémentaires (par exemple, manuels d'explication ou textes réglementaires liés).
- Les fonctionnalités de base d'un chatbot juridique : recherche d'information, génération augmentée, citation des sources.
- L'intégration d'une interface multimodale combinant texte et voix.

En revanche, le système n'a pas vocation à couvrir l'ensemble du droit sénégalais ni à fournir des conseils juridiques personnalisés : il se limite à l'information juridique brute et contextualisée.

3. Fonctionnalités prévues

Sur la base des objectifs et contraintes définis, les fonctionnalités attendues du système sont les suivantes :

3.1.Recherche documentaire hybride

- Intégration de deux approches complémentaires : la recherche dense (basée sur les embeddings vectoriels) et la recherche sparse (basée sur BM25 ou équivalent).
- Combinaison des résultats via la méthode RRF (Reciprocal Rank Fusion) pour maximiser la pertinence.

3.2.Génération augmentée (RAG)

- Construction d'un prompt enrichi à partir des passages sélectionnés.
- Utilisation d'un modèle local (Ollama avec Qwen, Mistral, Gemma, etc) pour générer une réponse synthétique, claire et juridiquement structurée.

3.3.Citations automatiques

- Intégration des références dans la réponse finale (articles du Code, sections de documents).
- Système garantissant la traçabilité des informations fournies.

3.4.Interface multimodale

- Saisie clavier : l'utilisateur peut poser une question par texte.
- Entrée vocale : activation via un bouton micro, reconnaissance de la voix avec Vosk.

3.5.Historique des conversations

- Conservation des échanges dans la session.
- Possibilité pour l'utilisateur de consulter ses questions précédentes et les réponses associées.

3.6.Extensibilité

- Prévoir une structure permettant d'ajouter d'autres domaines juridiques ou de nouveaux jeux de documents à l'avenir.

4. Architecture générale du système

L'architecture générale repose sur plusieurs modules interconnectés, chacun ayant un rôle spécifique dans le fonctionnement global du chatbot.

4.1.Base documentaire

- Les documents juridiques (Code du travail, textes complémentaires) sont découpés en passages (chunks).
- Chaque chunk est indexé à la fois de manière vectorielle (dense embeddings) et avec une méthode éparse (sparse).
- L'indexation est gérée via Milvus, un moteur de base de données vectorielle.

4.2.Moteur de recherche hybride

- Lorsqu'une question est posée, le moteur effectue une recherche dense et une recherche sparse.
- Les résultats sont fusionnés via RRF pour fournir un ensemble de passages pertinents.

4.3.LLM génératif (Ollama)

- Les passages pertinents servent à construire un prompt enrichi.
- Le modèle choisi (Qwen2.5 ou Mistral 7B, optimisé pour CPU) génère la réponse.
- Les citations sont intégrées dans la sortie finale.

4.4.Couche STT

- STT (Speech-to-Text) : Vosk capte la voix de l'utilisateur et la transforme en texte.

4.5.Interface utilisateur (Streamlit)

- Interface simple avec champ de saisie texte + bouton micro.
- Historique des conversations affiché dans une barre latérale.

Cette architecture modulaire permet d'assurer la clarté du système, sa maintenabilité et sa capacité à évoluer.

CHAPITRE III : Déroulement du projet

Dans le cadre du projet, une méthodologie agile a été privilégiée afin de favoriser une progression incrémentale et un suivi structuré des réalisations. Avec Trello, le travail a été organisé en plusieurs sprints, chacun correspondant à une étape clé du développement du chatbot juridique. Cette organisation a permis d'itérer rapidement, de tester les solutions mises en place au fur et à mesure et d'adapter les choix techniques en fonction des contraintes rencontrées.

Ainsi, le Sprint 1 a été consacré à la constitution et au traitement de la base documentaire, étape essentielle pour disposer d'un corpus fiable et exploitable. Le Sprint 2 quant à lui s'est focalisé sur l'indexation et la mise en place de la recherche hybride, combinant des vecteurs denses et clairsemés pour améliorer la pertinence des résultats. Enfin, sur cette base solide, les Sprints 3 et 4 ont permis de franchir un cap important : l'intégration d'un modèle de langage génératif et la mise en place d'une interface multimodale accessible

L'ensemble de ces sprints constitue un cheminement progressif allant de la préparation des données à l'intégration d'une interface interactive et multimodale, en passant par l'implémentation des mécanismes de recherche et de génération de réponses.

A. Sprint 1 : Préparation & Extraction

Le premier sprint du projet avait pour objectif principal de constituer une base documentaire exploitable pour l'entraînement et l'alimentation du futur chatbot. Dans le cadre de ce projet, deux sources essentielles ont été identifiées :

- Le Code du travail sénégalais, qui constitue la totalité des articles du code du travail sénégalais.
- Le Manuel du travailleur, qui représente une source complémentaire, plus explicative et vulgarisée, visant à éclairer les droits et obligations des travailleurs.

L'objectif était donc de transformer ces documents, initialement en format PDF, en un corpus textuel structuré et découpé en unités cohérentes appelées *chunks*. Ces chunks serviront de briques pour la recherche d'information et la génération augmentée par récupération (*Retrieval-Augmented Generation* – RAG).

1. Étapes réalisées sur le Code du travail

Le Code du travail était disponible sous la forme d'un PDF textuel, c'est-à-dire qu'il contenait déjà du texte sélectionnable et extractible. Il a donc été traité directement sans nécessiter de reconnaissance optique de caractères (OCR).

Le pipeline appliqué a suivi les étapes suivantes :

1. Extraction du texte : lecture page par page du fichier et récupération du contenu textuel brut.
2. Nettoyage : suppression des espaces multiples, normalisation des sauts de ligne et uniformisation du format.
3. Découpage en chunks (chunking) :
    - Le texte a été segmenté en morceaux de taille fixe (512 mots).
    - Un mécanisme de *recouvrement* (*overlap*) de 50 mots a été ajouté pour garantir qu'aucune information ne soit perdue à la jonction entre deux chunks. Cela permet d'assurer la continuité des phrases et d'améliorer la qualité de la recherche ultérieure.

Au terme de ce processus, le Code du travail a été découpé en 70 chunks. Ces unités constituent une base fiable et cohérente pour alimenter la mémoire documentaire du chatbot.

2. Étapes réalisées sur le Manuel du travailleur

Contrairement au Code du travail, le Manuel du travailleur était fourni sous forme d'un PDF numérisé, c'est-à-dire une image scannée dépourvue de texte exploitable. Dans ce cas, une simple extraction textuelle était impossible.

Pour rendre ce document utilisable, il a fallu recourir à la Reconnaissance Optique de Caractères (OCR).

Le pipeline mis en place a suivi les étapes suivantes :

- Conversion en images : chaque page du PDF a été convertie en une image haute résolution (300 dpi[5]) à l'aide de la bibliothèque *pdf2image*[6].

[5] dpi (dots per inch) : unité de mesure de la résolution d'une image ou d'un document imprimé, indiquant le nombre de points par pouce.
[6] pdf2image : bibliothèque Python permettant de convertir des pages PDF en images (par exemple PNG ou JPEG), utile pour l'annotation ou la prévisualisation de documents

- Application de l'OCR : l'outil *Tesseract OCR* a été utilisé pour reconnaître automatiquement les caractères présents dans les images et les convertir en texte brut.
- Correction d'encodage : étant donné que le document contenait de nombreux caractères français (accents, cédilles), un module de correction a été appliqué pour éviter les erreurs fréquentes de transcription (par exemple "Ã©" corrigé en "é").
- Assemblage et structuration : les pages reconnues ont été fusionnées dans un fichier texte unique, en conservant une séparation claire entre les pages.
- Nettoyage et chunking : comme pour le Code du travail, le texte a été normalisé puis découpé en chunks de 512 mots avec un overlap de 50 mots.

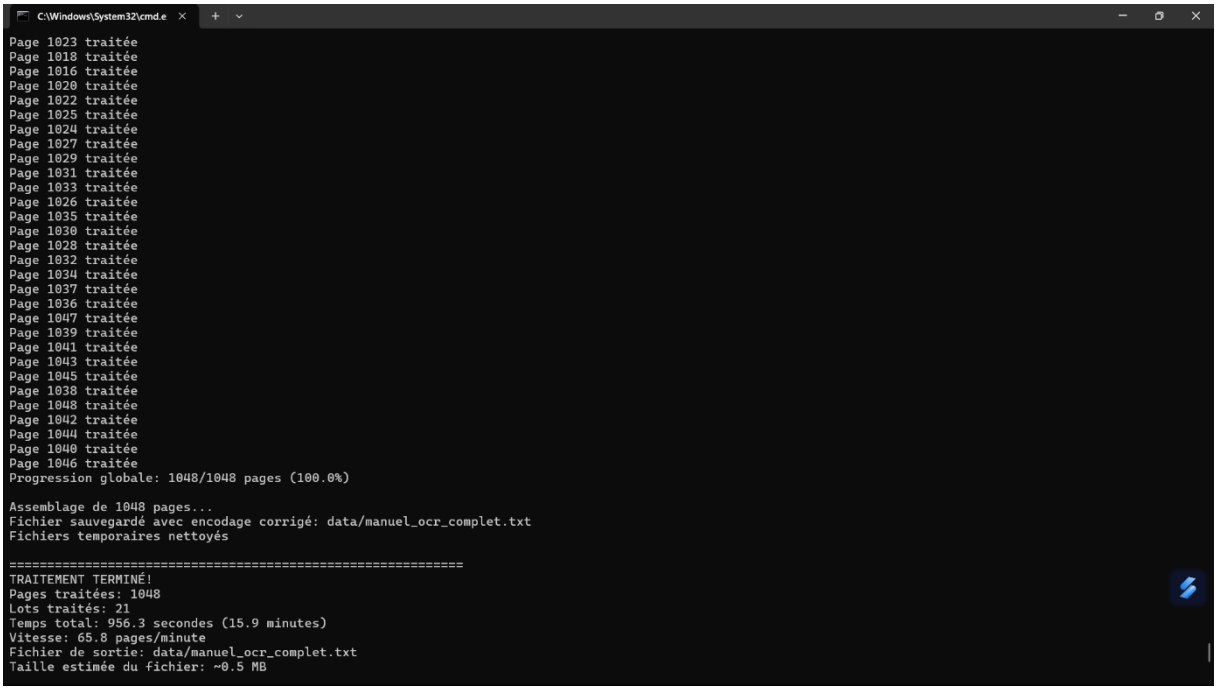

Figure 1 : OCR sur Le Manuel du Travailleur

Ce processus a permis d'obtenir 605 chunks exploitables à partir du Manuel du travailleur.

3. Justification et définition de l'OCR

3.1. Définition de l'OCR

L'OCR (Optical Character Recognition), ou reconnaissance optique de caractères, est une technologie qui permet de convertir une image contenant du texte imprimé ou manuscrit en un contenu numérique éditable. Concrètement, elle analyse la forme des lettres et les associe à des caractères de l'alphabet ou des symboles.

3.2. Fonctionnement technique

Un système OCR repose généralement sur plusieurs étapes :

- Prétraitement de l'image : amélioration du contraste, suppression du bruit visuel, correction d'inclinaison de la page.
- Segmentation : découpage de l'image en zones de texte, lignes, puis caractères individuels.
- Reconnaissance des caractères : utilisation d'algorithmes de classification (historiquement basés sur des règles, aujourd'hui souvent sur des réseaux de neurones) pour identifier chaque symbole.
- Post-traitement : correction des erreurs à l'aide de dictionnaires de langue et de règles grammaticales.

3.3. Outil utilisé

Dans ce projet, l'outil retenu a été Tesseract OCR, un logiciel libre développé par Google, reconnu pour sa performance dans la reconnaissance multilingue, notamment du français. Il a été associé à *pdf2image* pour la conversion des pages et au module *PyMuPDF* [7]pour l'analyse des métadonnées PDF.

3.4. Limites de l'OCR

Malgré son efficacité, l'OCR présente certaines limites :

- Qualité des documents sources : un scan de mauvaise qualité, trop sombre ou flou, dégrade la reconnaissance.
- Polices et mises en page complexes : certains tableaux ou documents juridiques avec colonnes peuvent être mal interprétés.
- Erreurs linguistiques : l'OCR peut confondre certains caractères similaires (par exemple « l » et « 1 »).

  Ces limites expliquent la nécessité d'un nettoyage post-OCR, comme cela a été réalisé dans ce projet.

[7] PyMuPDF : bibliothèque Python permettant de manipuler des fichiers PDF (lecture, extraction de texte, d'images ou de métadonnées) ainsi que d'autres formats de documents et images.

3.5. Justification de l'usage dans ce projet

L'usage de l'OCR s'imposait pour le Manuel du travailleur, car sans cette technologie, le contenu n'aurait pas pu être exploité. Grâce à ce traitement, un document de plus de 1000 pages a pu être transformé en un corpus textuel structuré, prêt à être utilisé pour l'entraînement du chatbot.

4. Résultats obtenus

Au terme du Sprint 1, les résultats suivants ont été atteints :

- Code du travail : 70 chunks générés.
- Manuel du travailleur : 605 chunks générés via OCR.

Ces deux bases de données textuelles constituent désormais la mémoire documentaire initiale du chatbot. Le recouvrement de 50 mots entre les chunks garantit la cohérence et évite toute perte d'information aux jonctions.

Ce premier sprint a permis de poser les fondations du projet en constituant une base documentaire fiable et exploitable. Les deux documents de référence — le Code du travail et le Manuel du travailleur — ont été transformés en un format adapté à l'indexation et à la recherche d'information.

La prochaine étape (Sprint 2) consistera à intégrer ces chunks dans un système vectoriel à l'aide de modèles d'embeddings. L'objectif sera de permettre une recherche sémantique efficace afin que le chatbot puisse retrouver les articles et passages pertinents en fonction des questions de l'utilisateur.

B. Sprint 2 : Indexation & RAG Basique

Ce deuxième sprint du projet de chatbot juridique avait pour objectif de mettre en place l'étape d'indexation et de réaliser une première version du mécanisme de recherche de documents, étape indispensable avant d'intégrer une génération de réponses. Concrètement, il s'agissait de stocker les textes du *Code du travail* et du *Manuel du travailleur* dans une base vectorielle, de leur associer des représentations numériques appelées *embeddings*, puis de permettre la recherche dans ces documents en combinant deux approches : une recherche sémantique et une recherche lexicale. Cette combinaison, appelée recherche hybride, a ensuite été testée sur un jeu de questions représentatives.

1. Présentation de Milvus et justification du choix

Pour répondre aux besoins d'indexation et de recherche, nous avons choisi d'utiliser Milvus, une base de données vectorielle open source spécialisée dans la gestion de données haute dimension. Une base vectorielle se distingue d'une base de données classique par sa capacité à stocker et interroger des vecteurs, c'est-à-dire des représentations numériques d'objets tels que des textes, des images ou des sons.

Milvus est particulièrement adaptée au projet pour plusieurs raisons. D'abord, elle supporte à la fois les vecteurs denses et les vecteurs épars, ce qui permet de mettre en œuvre une recherche hybride. Ensuite, elle offre des performances élevées, notamment grâce à des index spécialisés tels que HNSW [8]ou IVF[9], qui réduisent considérablement le temps de recherche. De plus, Milvus est bien intégré dans l'écosystème Python à travers la bibliothèque pymilvus, ce qui nous a permis de l'utiliser directement dans notre pipeline. Enfin, Milvus dispose d'une communauté active et d'une documentation claire, garantissant un bon support technique.

2. Installation et configuration de Milvus

La mise en place de Milvus nécessite l'utilisation de Docker, un outil qui permet de déployer facilement des applications dans des environnements isolés appelés *containers*. Dans notre cas, nous avons opté pour le mode standalone, plus simple à configurer que le mode cluster et suffisant pour un projet académique.

L'installation s'est déroulée en plusieurs étapes. Dans un premier temps, nous avons téléchargé l'image officielle de Milvus depuis DockerHub, en veillant à utiliser la version v2.6.1. Ensuite, nous avons lancé Milvus grâce au script standalone.bat fourni par le projet, qui démarre en parallèle les composants nécessaires : le moteur Milvus, ETCD pour la gestion de la configuration, et MinIO pour le stockage des données. Une fois Milvus démarré, nous avons vérifié son bon fonctionnement en utilisant la commande docker ps pour s'assurer que les containers étaient actifs. Enfin, nous avons testé la connexion via Python avec pymilvus, en confirmant que le service répondait bien sur le port 19530.

[8] HNSW (Hierarchical Navigable Small World) : algorithme d'indexation de recherche de similarité qui construit un graphe hiérarchique pour retrouver rapidement les vecteurs proches, très efficace pour les grandes bases vectorielles.

[9] IVF (Inverted File Index) : méthode qui partitionne l'espace des vecteurs en plusieurs cellules (clusters) et recherche uniquement dans les cellules les plus proches du vecteur requête, réduisant ainsi le temps de recherche.

Durant cette étape, certains problèmes techniques ont été rencontrés, comme une erreur de connexion à DockerHub ou des refus de connexion au port gRPC[10]. Ces difficultés ont été résolues en s'authentifiant correctement à DockerHub et en redémarrant le container de Milvus.

3. La notion de collection et de schéma

Dans Milvus, les données ne sont pas stockées de manière brute, mais organisées au sein de ce que l'on appelle des collections. Une collection peut être rapprochée d'une table dans une base de données relationnelle. Elle regroupe un ensemble de données ayant la même structure, définie par un schéma. Le schéma précise les champs de la collection, leurs types (entier, texte, vecteur dense, vecteur épars, etc.), ainsi que d'éventuelles contraintes comme la taille maximale d'un texte ou la dimension d'un vecteur.

Dans notre projet, nous avons créé une collection appelée chatbot_chunks_hybrid. Cette collection contient six champs :

- Un identifiant (id) généré automatiquement et servant de clé primaire ;
- Le champ source, qui indique l'origine du texte (Code du travail ou Manuel du travailleur) ;
- Le champ chunk_index, qui correspond au numéro du passage dans le document ;
- Le champ text, qui stocke le contenu textuel du passage ;
- Le champ dense, qui contient l'embedding dense, c'est-à-dire la représentation sémantique du texte ;
- Enfin le champ épars (sparse), qui contient l'embedding épars, représentant la dimension lexicale du texte.

Une fois la collection créée, il a fallu définir des index afin d'optimiser les recherches. Nous avons choisi l'index HNSW (Hierarchical Navigable Small World) pour les vecteurs denses, car il est reconnu pour sa rapidité dans les recherches de proximité. Pour les vecteurs épars, nous avons opté pour un index inversé (*sparse inverted index*), plus adapté aux représentations de type lexical.

[10] gRPC : framework open source développé par Google qui permet la communication distante entre services (Remote Procedure Call) en utilisant le protocole HTTP/2 et la sérialisation Protocol Buffers, garantissant rapidité, efficacité et compatibilité multi-langages.

4. Génération des embeddings

L'étape suivante consistait à générer les embeddings des textes. Un embedding est une représentation vectorielle d'un texte : il traduit le contenu linguistique dans un espace numérique, permettant de comparer facilement deux textes selon leur proximité. Pour cela, nous avons utilisé le modèle BGE-M3, via l'interface BGEM3EmbeddingFunction. Ce modèle présente l'avantage de produire simultanément deux types de vecteurs : un vecteur dense, qui capture la signification générale du texte, et un vecteur épars, qui préserve la correspondance exacte des mots.

Les documents ont été d'abord découpés en *chunks* lors du Sprint 1 (70 pour le Code du travail et 605 pour le Manuel du travailleur, soit 675 au total). Ces chunks ont ensuite été encodés en lot (*batch*) de 16 afin de limiter la consommation mémoire. Les vecteurs denses obtenus avaient une dimension de 1024, conformément au schéma de la collection. Quant aux vecteurs épars, ils ont été transformés en dictionnaires {index: valeur} pour être compatibles avec le format attendu par Milvus.

5. Pipeline d'insertion dans Milvus

L'insertion dans Milvus a suivi une logique progressive. Après avoir chargé les fichiers JSON contenant les chunks, nous avons généré les embeddings correspondants. Les données ont ensuite été préparées sous forme d'entités comprenant la source, l'index du chunk, le texte original, le vecteur dense et le vecteur épars.

Pour éviter de surcharger Milvus et réduire le risque d'erreurs RPC, nous avons adopté une insertion par batchs de 50 chunks. Chaque lot était inséré via la commande collection.insert, suivi d'un flush permettant d'écrire les données de manière permanente et d'un load pour rendre la collection immédiatement exploitable pour les recherches.

À la fin du processus, la collection contenait l'ensemble des 675 chunks correctement indexés et prêts à être interrogés.

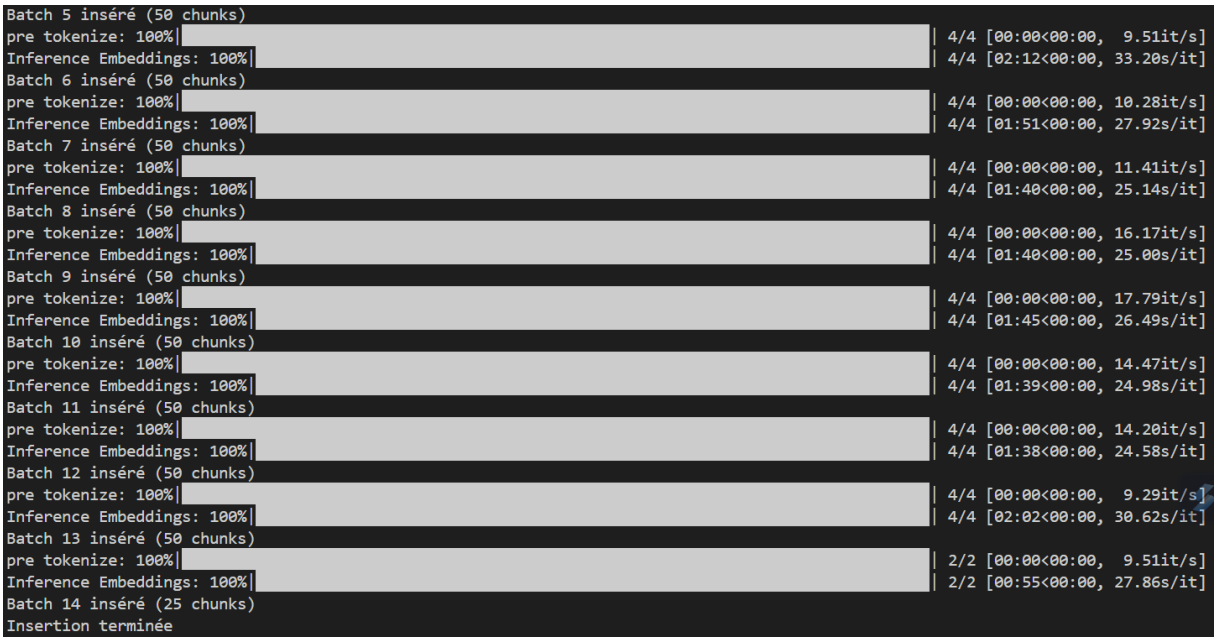

Figure 2 : Insertion des chunks dans Milvus

6. Recherche hybride et fusion des résultats

Une fois les données insérées, nous avons mis en place une première fonction de recherche hybride. La démarche consistait à générer, pour une requête donnée, ses embeddings denses et épars. Une recherche dense était alors effectuée sur le champ dense, et une recherche éparse sur le champ sparse.

Les deux résultats de recherche étaient ensuite fusionnés en utilisant une méthode appelée Reciprocal Rank Fusion (RRF). Le principe de cette méthode est de combiner les listes de résultats issues des deux recherches en attribuant à chaque document un score basé sur sa position dans les classements partiels. Les documents apparaissant en tête dans les deux listes sont ainsi favorisés dans le classement final.

Cette stratégie permettait de bénéficier des avantages complémentaires des deux approches : la recherche dense apporte une compréhension sémantique, tandis que la recherche éparse garantit la prise en compte des termes exacts.

7. Résultats observés

L'expérimentation de cette recherche hybride a été réalisée sur un petit jeu de cinq questions représentatives, couvrant des thématiques telles que le licenciement, la maladie du travailleur ou encore les indemnités de congé. Les résultats obtenus ont montré une bonne pertinence : pour chaque question, la liste finale contenait des passages issus soit du Code du travail soit du Manuel, apportant des éléments de réponse concrets.

Par exemple, sur la question des conditions de licenciement, la recherche dense a permis de retrouver des passages décrivant la procédure générale, tandis que la recherche éparse a ramené des articles mentionnant explicitement le terme « licenciement ». La fusion des résultats a donc produit un classement équilibré, intégrant à la fois le contexte et la précision terminologique.

Le temps de recherche est resté raisonnable, de l'ordre d'une à deux secondes par requête, malgré une exécution uniquement sur CPU. Certaines redondances ont toutefois été observées, un même passage pouvant apparaître deux fois s'il était retrouvé par les deux recherches.

Le Sprint 2 a constitué une étape décisive dans la construction du chatbot juridique. Nous avons non seulement mis en place une collection hybride adaptée dans Milvus, mais aussi indexé l'ensemble des documents de référence avec des vecteurs denses et épars. Nous avons également conçu un pipeline d'insertion robuste et une fonction de recherche hybride reposant sur la fusion RRF.

Les tests réalisés ont confirmé la pertinence de cette approche, en montrant que la combinaison des deux types de recherche permet de retrouver des informations à la fois précises et contextuelles. Ce résultat ouvre la voie à l'intégration d'un modèle génératif lors du Sprint 3, afin de formuler des réponses rédigées et accessibles aux utilisateurs.

C. Sprint 3 : Génération de réponses et intégration des citations

1. Objectif du sprint

Après la mise en place de la recherche hybride au cours du Sprint 2, l'étape suivante consistait à doter le chatbot de la capacité à formuler des réponses complètes, structurées et contextualisées à partir des extraits juridiques retrouvés. L'objectif principal de ce sprint était donc de connecter la couche RAG (Retrieval-Augmented Generation) à un modèle de langage génératif (LLM), capable non seulement de comprendre les résultats renvoyés par la recherche mais également de les transformer en réponses adaptées aux besoins des utilisateurs. Pour ce faire, la plateforme Ollama a été choisie, car elle permet d'exécuter localement des modèles open source récents et performants, sans dépendre d'une API externe.

2. Choix du modèle de langage (LLM)

Dès le début de ce sprint, plusieurs modèles ont été envisagés. Le modèle Phi-3.5 Mini (3.8B), développé par Microsoft, a retenu l'attention grâce à sa taille modérée et à sa bonne réputation en matière de génération de texte. De même, Mistral 7B Instruct, reconnu comme l'un des modèles open source les plus performants de sa catégorie, représentait une option attrayante pour obtenir des réponses plus riches et nuancées.

Toutefois, ces deux alternatives présentaient un inconvénient majeur dans notre contexte : elles nécessitaient des ressources matérielles importantes, en particulier une mémoire RAM supérieure à 16 Go et idéalement un GPU dédié. Dans notre cas, l'environnement de travail était limité à une machine dotée de 8 Go de RAM et sans GPU, ce qui excluait d'emblée les modèles les plus volumineux.

Le choix s'est donc porté sur Qwen2.5-3B, un modèle offrant un compromis satisfaisant entre légèreté et qualité des réponses. Sa taille plus réduite (3 milliards de paramètres) le rend exploitable dans un environnement contraint tout en conservant des capacités suffisantes pour la tâche de génération de réponses juridiques contextualisées

3. Présentation du modèle Qwen2.5-3B

Le modèle Qwen2.5-3B appartient à la famille des modèles Qwen (Tongyi Qianwen), développée par Alibaba Group. Cette série de modèles open source a été conçue pour répondre à une large gamme de tâches en traitement automatique du langage naturel (NLP), allant de la compréhension de texte à la génération de réponses contextualisées.

Avec environ 3 milliards de paramètres, Qwen2.5-3B constitue une version compacte de la gamme Qwen. Son objectif est d'offrir un équilibre entre performance et consommation de ressources, le rendant particulièrement adapté à des environnements CPU-only et à des machines disposant d'une mémoire limitée. Contrairement à des modèles plus lourds comme Mistral 7B ou LLaMA 13B, Qwen2.5-3B peut être exécuté sur des ordinateurs dépourvus de GPU puissant, ce qui correspond exactement à nos contraintes.

Sur le plan technique, Qwen2.5-3B est un modèle instruct-tuned, c'est-à-dire affiné pour répondre de manière conversationnelle à des instructions ou des questions. Il peut ainsi générer des explications structurées, précises et adaptées à un style de dialogue.

Un autre atout majeur réside dans son support multilingue, incluant le français. Étant donné que les textes juridiques sénégalais sont rédigés en français, cette caractéristique rend le modèle particulièrement pertinent pour notre projet. Enfin, le fait qu'il soit disponible open

source via Ollama simplifie grandement son intégration locale, tout en garantissant l'indépendance vis-à-vis d'API externes souvent payantes ou limitées.

4. Construction du prompt juridique

La performance d'un modèle génératif dépend en grande partie de la qualité du prompt qui lui est fourni. Dans ce sprint, une attention particulière a donc été portée à la construction du prompt juridique. Celui-ci a été conçu pour guider le modèle vers des réponses précises, fiables et structurées.

Le prompt comprend plusieurs éléments :

- Le contexte, constitué des extraits de textes juridiques retrouvés par la recherche hybride. Chaque passage est accompagné de sa source et de son index, afin de pouvoir être cité explicitement.
- La question de l'utilisateur, formulée telle quelle.
- Les instructions spécifiques, orientées vers la précision juridique : répondre uniquement à partir du contexte fourni, citer les articles ou sections correspondants, éviter toute spéculation, et utiliser un langage clair mais juridiquement correct.

Ce dispositif permet de contraindre le modèle à s'appuyer uniquement sur les données disponibles, renforçant ainsi la crédibilité et la vérifiabilité du chatbot.

5. Résultats des tests

Le système a été évalué à travers une dizaine de questions inspirées du droit du travail sénégalais. Parmi celles-ci figuraient :

- « Quels sont les droits du travailleur malade ? »
- « Quelles sont les conditions de licenciement ? »
- « Que se passe-t-il en cas de décès de l'agent ? »

Dans la majorité des cas, les réponses générées se sont révélées pertinentes, correctement structurées et accompagnées de citations directes issues du corpus juridique indexé. Les résultats ont démontré que l'approche RAG combinée à Qwen2.5-3B était capable de restituer une information fiable, tout en améliorant l'accessibilité à la réglementation du travail.

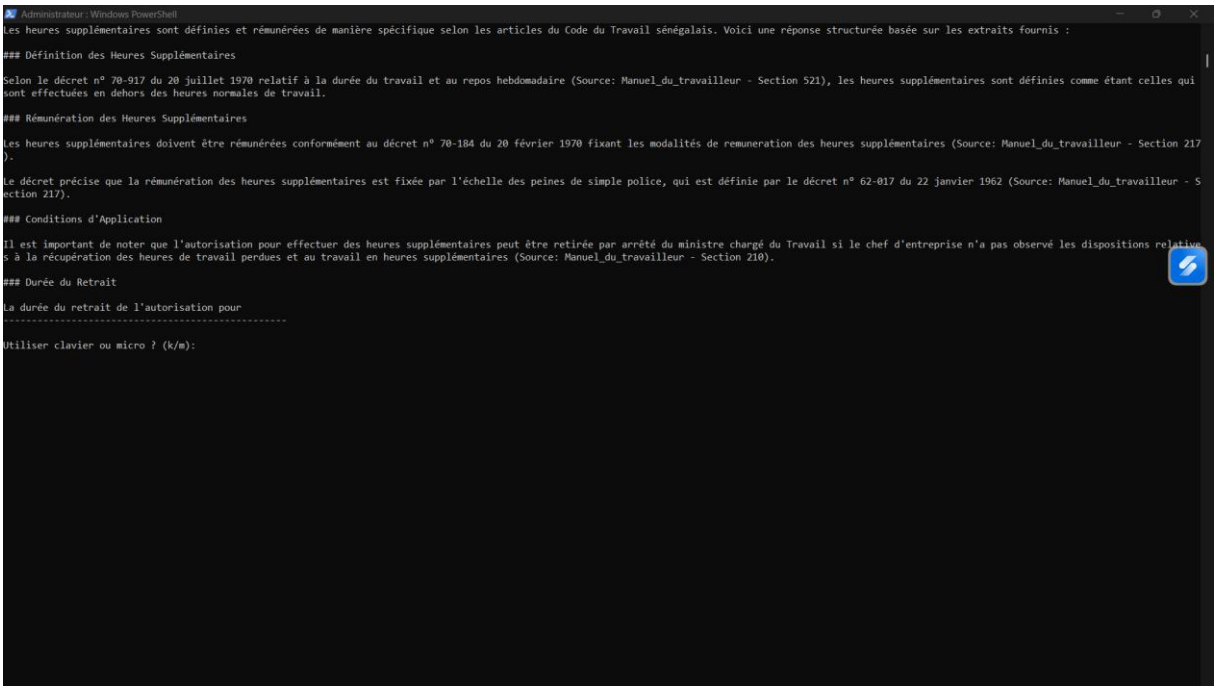

Figure 3 : Test réalisé en ligne de commande

6. Limites rencontrées

Malgré ces résultats positifs, une limite majeure a été observée : la lenteur de génération des réponses. Même avec un modèle compact comme Qwen2.5-3B, l'absence de GPU a fortement impacté les performances. Chaque requête nécessitait parfois plusieurs dizaines de secondes avant de produire une réponse complète.

Cette lenteur s'explique par la nature même des modèles de langage, qui reposent sur des calculs matriciels massivement parallèles. Sur GPU, ces calculs sont optimisés et rapides, tandis que sur CPU ils s'exécutent de manière séquentielle, entraînant une baisse considérable de la vitesse d'inférence.

Cette contrainte constitue une limite importante pour l'expérience utilisateur. Elle met en évidence la nécessité, dans les perspectives futures, de recourir soit à un environnement matériel plus performant (GPU), soit à des modèles encore plus légers et optimisés pour CPU, afin de garantir une interaction fluide avec l'utilisateur.

D. Sprint 4 : Développement de l'interface utilisateur multimodale

1. Objectif du sprint

Après avoir validé la capacité du système à générer des réponses juridiques cohérentes et citées, l'étape suivante consistait à rendre le chatbot accessible et utilisable par un public non technique. L'objectif principal de ce sprint était donc de développer une interface graphique conviviale qui permette à l'utilisateur de dialoguer naturellement avec le système. L'accent a été mis sur la simplicité, l'ergonomie et la possibilité d'intégrer plusieurs modalités d'entrée afin de rendre l'outil accessible à un large éventail d'usagers.

2. Choix techniques et outils utilisés

Pour répondre à ces besoins, le framework Streamlit a été retenu. Ce choix se justifie par plusieurs atouts :

- Sa facilité de mise en œuvre, qui permet de prototyper rapidement une interface interactive.
- Sa compatibilité avec Python, langage déjà utilisé pour la couche RAG.
- Son intégration native de composants interactifs (zones de saisie, boutons, sidebar, etc.).

En parallèle, pour l'entrée vocale, l'outil Vosk a été utilisé. Entièrement open source, Vosk offre plusieurs avantages décisifs :

- Il est compatible avec les environnements CPU-only, donc utilisable même sans GPU.
- Il supporte le français, indispensable pour le domaine juridique ciblé.
- Il permet un mode batch, c'est-à-dire que la transcription n'apparaît qu'une fois l'utilisateur arrêté de parler, évitant ainsi des textes partiels ou bruités.

Ces choix techniques ont donc permis de concilier contraintes matérielles et besoins fonctionnels.

3. Présentation de l'interface

L'interface utilisateur a été conçue autour de deux modes d'entrée :

- La saisie au clavier : l'utilisateur tape directement sa question dans une zone de texte, comme dans un chatbot classique.

- L’entrée vocale : un bouton micro a été intégré directement dans la zone de saisie. Lorsqu’il est activé, Vosk enregistre la voix de l’utilisateur, effectue la transcription et insère automatiquement le texte reconnu dans un box et l’utilisateur pourra copier ce dernier dans la zone de saisie et valider pour soumettre sa question.

L’expérience utilisateur a donc été pensée pour être fluide et flexible : chacun peut choisir entre parler ou taper, sans avoir à changer d’interface.

Un autre ajout important a été l’historique des conversations. Celui-ci apparaît dans une barre latérale, affichant l’ensemble des questions posées et des réponses générées. Cette fonctionnalité apporte une continuité et une traçabilité des échanges, facilitant le suivi de la consultation juridique.

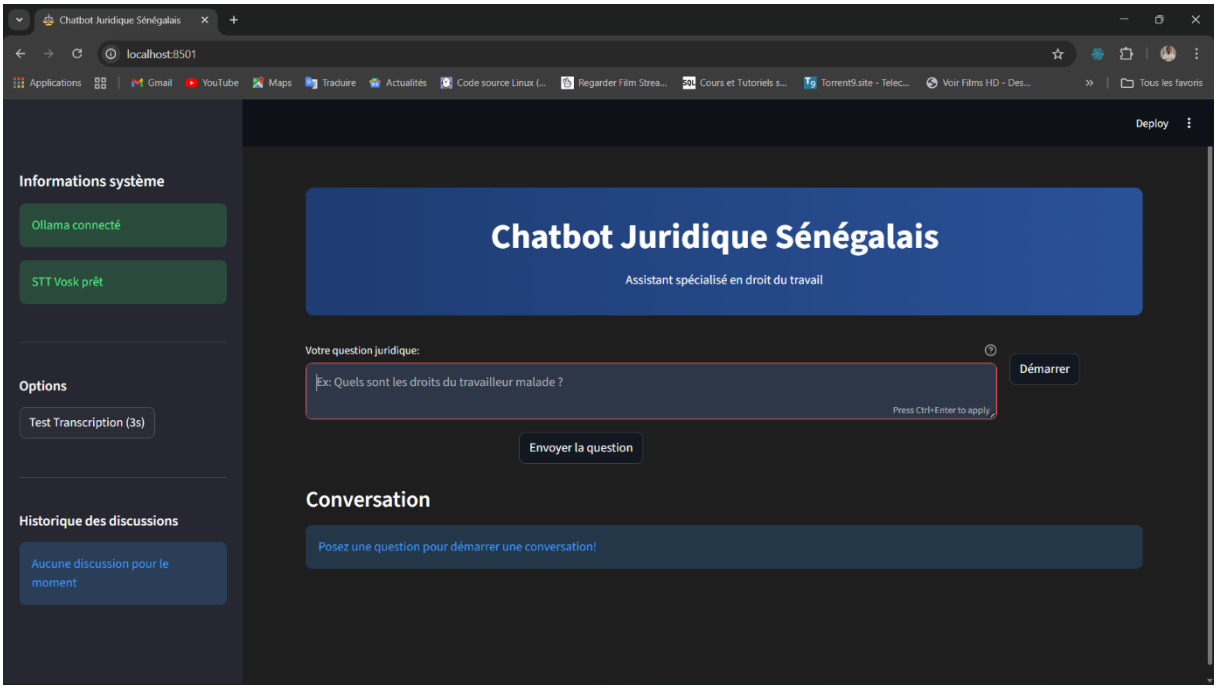

Figure 4 : Présentation de l’interface utilisateur

4. Résultats et tests

Lors des tests, l’interface a permis une interaction fluide avec le chatbot. L’utilisateur pouvait :

- poser des questions par écrit et obtenir des réponses pertinentes générées par Qwen2.5-3B,
- utiliser le micro pour dicter sa question, qui était ensuite automatiquement transcrite et,
- consulter l’historique des conversations pour garder une trace des échanges.

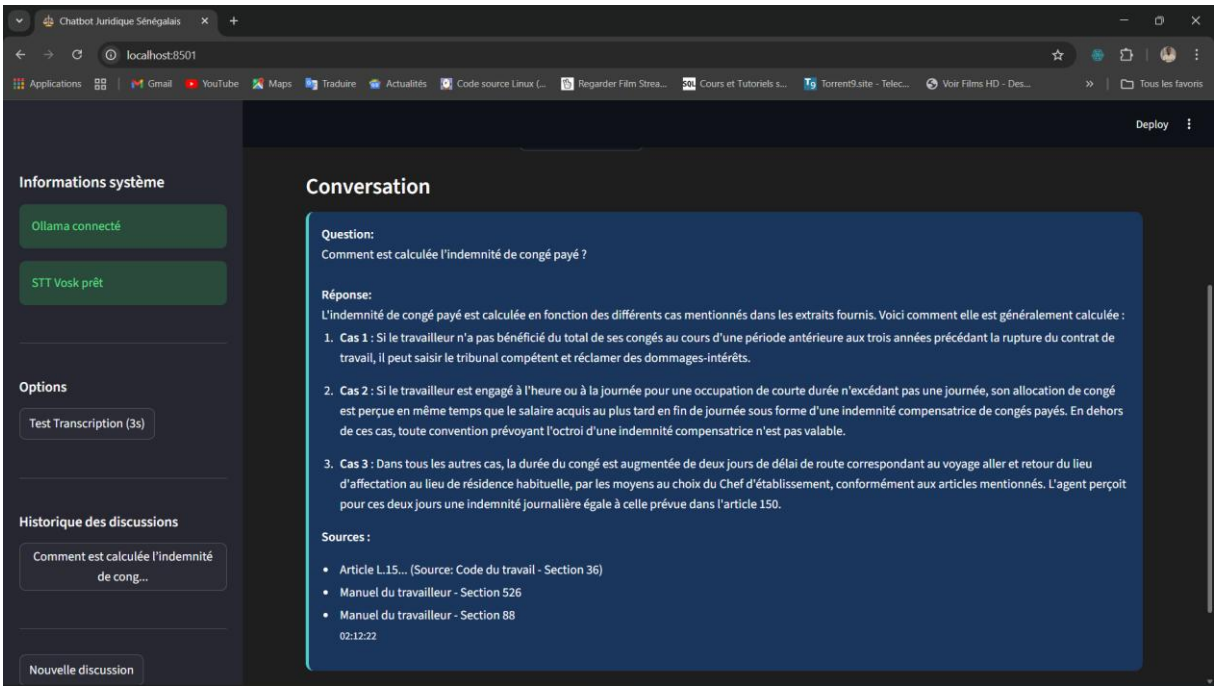

Figure 5 : Test de l'option saisie au clavier

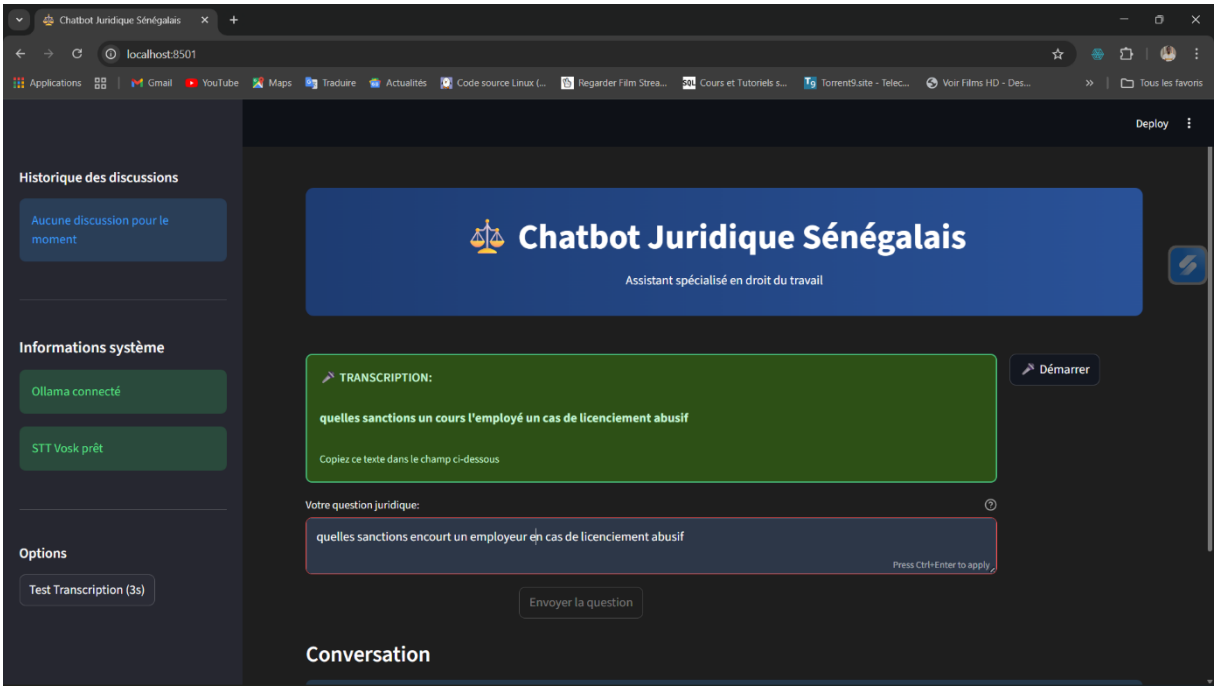

Figure 6 : Test de l'option entrée vocale

Cette configuration a permis de valider la dimension multimodale du projet, en combinant efficacement texte et voix.

5. Limites rencontrées

Malgré ces réussites, certaines limites ont été identifiées :

- La lenteur du LLM reste le principal frein. Comme observé lors du Sprint 3, l'absence de GPU entraîne des temps de réponse élevés (jusqu'à plusieurs dizaines de secondes), ce qui peut dégrader l'expérience utilisateur malgré la convivialité de l'interface.
- Enfin, la gestion des ressources (CPU et RAM) reste critique : l'application consomme une part importante des capacités de la machine, ce qui pourrait poser des problèmes en cas de déploiement sur un environnement encore plus contraint.

CHAPITRE IV : Perspectives et améliorations futures

La réalisation de ce projet a permis de mettre en place un prototype fonctionnel de chatbot juridique spécialisé dans le droit du travail sénégalais. Toutefois, comme tout projet informatique, certaines limites subsistent et ouvrent la voie à de nombreuses perspectives d'amélioration. Ces perspectives concernent à la fois les aspects techniques, l'ergonomie de l'interface, l'élargissement fonctionnel et l'impact sociétal attendu.

1. Optimisation des performances

L'une des principales limites observées concerne la lenteur de génération des réponses par le modèle de langage. Malgré le choix d'un modèle relativement léger comme Qwen2.5-3B, l'exécution sur CPU dans un environnement limité à 8 Go de RAM entraîne des délais importants, parfois supérieurs à plusieurs dizaines de secondes. Cette contrainte technique s'explique par la nature des modèles de langage, qui reposent sur des calculs matriciels massivement parallèles, optimisés pour GPU mais bien plus lents sur CPU.

Pour pallier cette limite, plusieurs pistes peuvent être envisagées. La première consiste à déployer le chatbot sur une machine dotée d'un GPU, ce qui permettrait de tirer parti de l'accélération matérielle et de réduire considérablement les temps de réponse. Une seconde piste repose sur l'utilisation de modèles quantifiés (par exemple en int4 ou int8), qui réduisent l'empreinte mémoire et accélèrent l'inférence tout en maintenant une qualité de génération acceptable. Ces approches constituent des améliorations prioritaires pour rendre le système plus fluide et utilisable dans un cadre réel.

2. Amélioration de la recherche hybride

La recherche hybride, combinant embeddings denses et représentations clairsemées, a montré de bons résultats dans ce projet. Néanmoins, plusieurs axes d'amélioration restent possibles. Il serait pertinent d'ajouter un module de re-ranking, comme TILDEv2 ou BGE-Reranker, afin de réordonner les passages retrouvés par Milvus avant leur transmission au modèle génératif. Une telle étape permettrait d'améliorer la précision des résultats en mettant systématiquement en avant les extraits les plus pertinents.

Enfin, l'enrichissement du corpus documentaire constitue une perspective essentielle. L'intégration de nouvelles sources, comme les jurisprudences, les conventions collectives ou encore les textes réglementaires, renforcerait la couverture et la pertinence des réponses générées.

3. Enrichissement de l'interface utilisateur

L'interface développée avec Streamlit constitue une première étape réussie vers un chatbot accessible et ergonomique. Cependant, plusieurs améliorations sont envisageables pour enrichir l'expérience utilisateur. La première piste concerne la réintégration de la synthèse vocale (TTS). Bien qu'une tentative ait été faite avec Piper, des problèmes de compatibilité sous Windows ont conduit à suspendre cette fonctionnalité. Sa stabilisation permettrait au chatbot de restituer oralement les réponses, offrant une expérience plus naturelle et inclusive.

De plus, le passage du mode batch actuel de reconnaissance vocale à un mode streaming constituerait une avancée significative. L'utilisateur pourrait ainsi voir sa parole transcrite en temps réel, ce qui améliorerait l'interactivité et réduirait l'effet d'attente. D'autres améliorations ergonomiques pourraient également être explorées, comme la mise en évidence visuelle des citations juridiques dans les réponses, le surlignage des articles pertinents ou encore la possibilité d'exporter les échanges sous forme de fichiers PDF.

4. Extension des fonctionnalités

Au-delà des optimisations techniques et ergonomiques, le système peut évoluer vers des fonctionnalités plus avancées. L'une des pistes consiste à développer une gestion des cas complexes, où le chatbot serait capable de poser des questions de clarification lorsque la requête est trop vague ou ambiguë. Une autre piste serait la mise en place d'une API REST stable, permettant d'intégrer le chatbot dans d'autres environnements (applications mobiles, sites web institutionnels, plateformes de formation, etc.).

De plus, l'ajout d'un système d'authentification et de suivi personnalisé permettrait de différencier les utilisateurs et de conserver un historique adapté à chacun. Cela renforcerait la dimension pratique de l'outil et ouvrirait la voie à des applications professionnelles plus ciblées.

5. Déploiement et impact sociétal

Enfin, au-delà des aspects purement techniques, ce projet présente des perspectives d'impact social et institutionnel importantes. À terme, le chatbot pourrait être déployé en ligne afin de garantir un accès élargi au droit du travail sénégalais. Il pourrait également être enrichi en collaboration avec des juristes, avocats, syndicats ou institutions publiques, afin de valider la pertinence des réponses et d'assurer leur conformité juridique.

Ce projet pourrait ainsi jouer un rôle clé dans la vulgarisation juridique, en facilitant l'accès à des informations fiables pour les travailleurs, les employeurs et les citoyens en général.

L’efficacité du chatbot pourrait être évaluée à travers des enquêtes de satisfaction ou des mesures d’impact sur l’accès effectif au droit.

En somme, les perspectives de ce projet s’articulent autour de trois grands axes : l’optimisation technique (performance et précision), l’amélioration de l’expérience utilisateur (interface multimodale plus riche et fluide) et l’élargissement de l’impact (déploiement institutionnel et sociétal). Si les bases posées par ce prototype sont solides, ces perspectives constituent autant de pistes pour transformer ce projet expérimental en un véritable outil d’accès au droit, fiable, rapide et accessible à tous.

CONCLUSION

Ce travail s'inscrit dans une démarche visant à tirer parti des avancées récentes en intelligence artificielle pour répondre à un enjeu majeur : l'accès à l'information juridique au Sénégal, notamment dans le domaine du droit du travail. Dans un contexte où les textes législatifs et réglementaires sont souvent complexes et peu accessibles au grand public, la conception d'un chatbot juridique basé sur une approche RAG (Retrieval-Augmented Generation) offre une solution innovante, combinant rigueur juridique et facilité d'utilisation.

L'ensemble du projet a été conduit selon une méthodologie agile, découpée en sprints successifs. Cette approche incrémentale a permis de progresser étape par étape, depuis la préparation et l'indexation du corpus documentaire jusqu'à l'intégration d'une interface utilisateur multimodale. Les deux premiers sprints ont posé les bases en constituant un corpus juridique structuré, puis en mettant en place une recherche hybride dense et sparse avec Milvus. Le troisième sprint a marqué un tournant majeur avec l'intégration de la génération de réponses via le modèle Qwen2.5-3B, exécuté localement grâce à Ollama, et enrichi par un système de citations automatiques. Le quatrième sprint a permis de doter le chatbot d'une interface accessible et interactive grâce à Streamlit, combinant saisie clavier et saisie vocale (via Vosk), avec gestion de l'historique des conversations. Enfin, les perspectives explorées mettent en évidence le potentiel d'évolution du système, notamment en termes de performances, d'ergonomie et de déploiement sociétal.

Au regard des résultats obtenus, ce projet démontre la faisabilité d'un chatbot juridique spécialisé, capable de fournir des réponses contextualisées et sourcées à partir d'un corpus fiable. Toutefois, plusieurs limites ont été relevées, notamment la lenteur de génération des réponses, liée à l'absence de GPU et à l'exécution sur un environnement contraint (CPU et 8 Go de RAM). Ces contraintes techniques, bien qu'elles n'altèrent pas la validité du système, freinent son usage dans un contexte de production. Elles ouvrent ainsi des perspectives d'amélioration, qu'il s'agisse de l'optimisation des performances (GPU, quantification des modèles), de l'enrichissement de l'interface (TTS, streaming vocal), ou encore du déploiement institutionnel à grande échelle.

En définitive, ce mémoire met en lumière la capacité de l'intelligence artificielle à transformer l'accès au droit en offrant des outils pratiques, fiables et accessibles. Le chatbot développé constitue un prototype prometteur, qui, après des optimisations techniques et un élargissement

du corpus, pourrait devenir un véritable outil d'aide à la compréhension et à l'application du droit du travail sénégalais. Il ouvre la voie à une démocratisation de l'information juridique et à une meilleure protection des travailleurs et des employeurs par une connaissance plus large et plus directe de leurs droits et obligations.

WEBOGRAPHIE

- La documentation officielle de Milvus : https://milvus.io/fr
- Le site de Ollama pour téléchargement de modèles open source : https://ollama.com/
- Un article de chercheurs en IA qui parle des bonnes pratiques RAG : https://arxiv.org/html/2407.01219v1#A1.SS2
- Youtube: https://www.youtube.com/
- ChatGPT: https://chatgpt.com/
- Claude AI : https://claude.ai/

TABLES DES MATIERES