

ChatBot Langchain pour l'UQAC

Réalisé par :

Mathis Aulagnier - AULM12040200

Ryan Collobert - COLR28120200

Samuel Madrigal - MADS23060200

Cours : 8INF974 – ATELIER PRATIQUE EN INTELLIGENCE ARTIFICIELLE II

1 Introduction

1.1 État des lieux

L'Université du Québec à Chicoutimi (UQAC) regroupe plus de 6500 étudiants répartis dans plus de 200 programmes d'études. Pour assurer sa gestion administrative, l'université dispose d'un manuel de gestion qui regroupe l'ensemble des règlements, politiques et procédures. Face au volume croissant de ce manuel, les employés rencontrent des difficultés pour retrouver rapidement l'information nécessaire.

Dans le cadre du cours 8INF974 - Atelier pratique en intelligence artificielle II, notre équipe de trois étudiants a été mandatée pour développer une preuve de concept d'un chatbot. Ce projet s'inscrit dans une démarche d'innovation, exploitant les récentes avancées en traitement du langage naturel (NLP) et en apprentissage automatique (ML).

1.2 Objectifs

Notre projet vise à créer un assistant conversationnel capable de répondre efficacement aux questions des employés concernant le manuel de gestion. Pour atteindre cet objectif, nous explorons la mise en œuvre de chatbots avec LangChain, intégrons l'intelligence artificielle dans une application web, et approfondissons nos connaissances dans le développement d'IA.

Notre solution s'appuie sur la technique RAG (Retrieval Augmented Generation), qui combine deux étapes clés :

- Le Retrieval (Récupération) : recherche des informations pertinentes dans une base de données en fonction de la question posée
- L'Augmented Generation (Génération Augmentée) : utilisation des informations récupérées pour enrichir le contexte et générer une réponse précise via un modèle de langage

En combinant ces deux étapes, le RAG peut produire des réponses plus précises et contextuellement appropriées.

Pour des raisons de confidentialité et de coûts, notre solution utilise un modèle de langage hébergé localement plutôt qu'un service cloud.

1.3 Gestion du projet

Nous avons organisé entre une et deux réunions par semaine afin de faire le point sur les avancées depuis la dernière réunion, poser nos questions, définir les prochaines étapes et répartir les tâches à venir. En complément, nous avons communiqué via Discord, et le code du projet est hébergé sur Git.

2 Scraping des Pages Web

Notre approche de scraping a évolué en trois phases distinctes pour optimiser la qualité des données extraites :

1. Scraping Initial

La première approche a été réalisée avec un script Python basique effectuant du drill-down sur le site. Cette méthode a révélé des limitations importantes dues à la structure répétitive des pages (table des matières et sommaire redondants).

2. Optimisation du Script

Amélioration du script pour :

- Supporter l'extraction des fichiers PDF
- Cibler spécifiquement les balises "Article"
- Filtrer les éléments HTML indésirables

3. Solution Finale avec Crawl4AI

La découverte du dépôt `Crawl4AI` sur GitHub (<https://github.com/unclecode/crawl4ai>) a été notre choix final pour le scraping. Cette bibliothèque, spécialement conçue pour l'extraction de données destinées aux modèles de langage, offre plusieurs avantages cruciaux :

- **AsyncWebCrawler** : Cette bibliothèque facilite le traitement parallèle des requêtes, ce qui diminue considérablement le temps nécessaire pour extraire les données. Sa capacité à gérer les opérations de manière asynchrone est particulièrement utile pour traiter le grand nombre de pages de notre manuel de gestion. Cependant, nous avons dû changer notre adresse IP à plusieurs reprises car le site de l'UQAC nous a bannis en raison de l'envoi soudain d'un trop grand nombre de requêtes.
- **DefaultMarkdownGenerator** : Ce composant assure une conversion fidèle du contenu HTML/PDF en Markdown, préservant la structure des documents tout en les rendant plus accessibles pour notre chaîne de traitement RAG.
- **Le nettoyage du contenu** est optimisé grâce aux filtres préconfigurés de Crawl4AI. Notre script `getArticle.py` utilise notamment le `PruningContentFilter` avec un seuil de 0.4, permettant de:
 - Isoler le contenu pertinent dans les balises "article"
 - Supprimer les éléments de navigation redondants
 - Éliminer les publicités et contenus non pertinents

Finalement, notre script `getURL.py` utilise le `sitemap.xml` du site pour cartographier en profondeur le site. Le `sitemap` nous a notamment permis d'identifier plus de **293** pages pertinentes à traiter.

3 Conception de notre RAG

3.1 Contexte et Objectifs

La mise en place d'un système RAG (Retrieval-Augmented Generation) nécessite une recherche approfondie sur les méthodes de construction et les optimisations possibles. Notre démarche a débuté par l'identification des formats de données les plus adaptés aux modèles de langage. Le format Markdown s'est imposé naturellement, offrant une structure claire tout en restant simple à manipuler. Cette décision a guidé notre choix d'utiliser `crawl4AI` pour l'extraction des données, nous permettant ainsi d'obtenir directement un format optimal pour la suite du traitement.

3.2 Comparaison des Architectures de Chatbots

3.2.1 Le Chatbot Traditionnel

Le chatbot basé sur un graphe représente l'approche la plus conventionnelle. Il fonctionne avec un ensemble prédéfini de questions et réponses stockées dans une base de données relationnelle. Bien que cette méthode garantisse des réponses rapides et précises pour les cas anticipés, elle souffre d'un manque crucial de flexibilité. L'impossibilité de traiter des questions non prévues limite considérablement son utilité dans des contextes dynamiques.

Voici, le schéma présentant l'architecture du chatbot traditionnel :

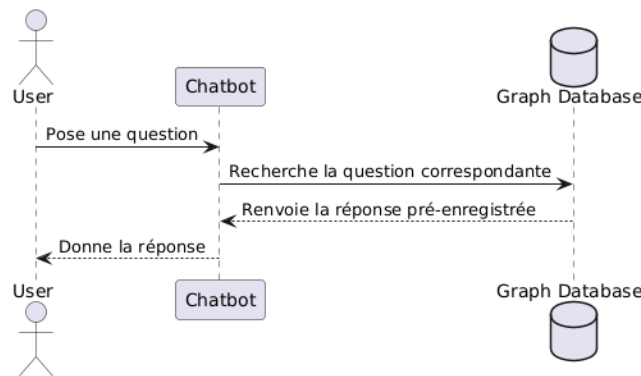


Figure 1: Architecture d'un chatbot basé sur un graphe

3.2.2 L'Évolution vers les LLM

L'introduction des modèles de langage (LLM) a marqué une évolution majeure pour les chatbots. Ces systèmes génèrent des réponses en s'appuyant sur leur compréhension du langage naturel, acquise lors de leur entraînement. Bien que plus flexibles que leurs prédécesseurs, ils présentent une limitation importante : l'absence de connexion à des sources d'information externes actualisées. Cette caractéristique peut conduire à des réponses imprécises ou obsolètes, particulièrement dans des domaines spécialisés.

Voici, le schéma présentant l'architecture du chatbot connecté à un LLM :

3.2.3 L'Innovation RAG

Le système RAG représente une synthèse innovante, combinant les avantages des deux approches précédentes. En associant la puissance générative d'un LLM à une base de connaissances externe, il permet de produire des réponses à la fois contextuelles et factuellement précises. Cette architecture permet une mise à jour continue des connaissances sans nécessiter de réentraînement du modèle.

Voici, le schéma présentant l'architecture d'un RAG :

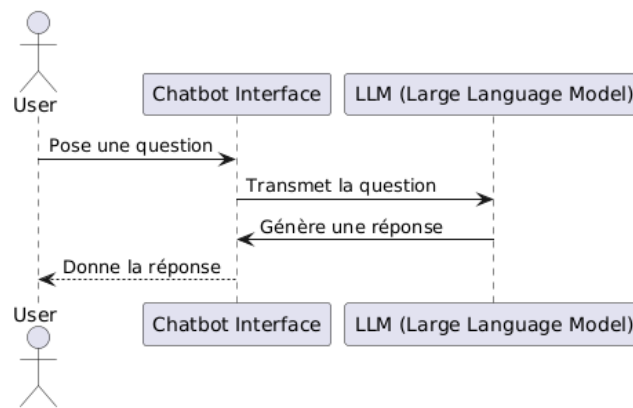


Figure 2: Architecture d'un chatbot basé sur un LLM

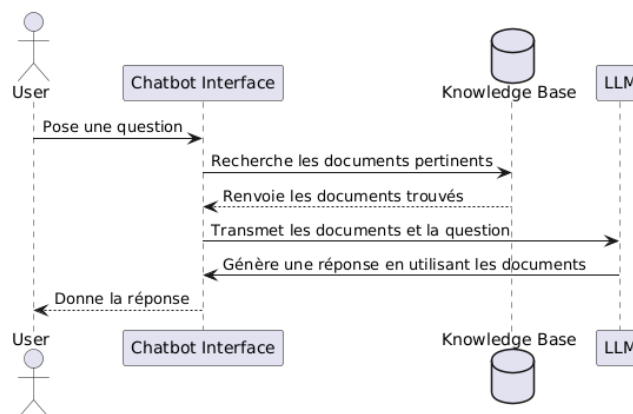


Figure 3: Architecture d'un chatbot basé sur un RAG

3.3 Système de Recherche Intelligent

3.3.1 Défis de la Recherche Documentaire

La recherche efficace dans une base documentaire pose un défi majeur. Une approche par mots-clés simples s'avère souvent insuffisante, produisant trop de résultats sans garantie de pertinence contextuelle. Notre solution repose sur l'utilisation d'embeddings, permettant une compréhension plus profonde du contexte des requêtes.

3.3.2 Vectorisation Sémantique

Notre système transforme aussi bien les questions que les documents en vecteurs numériques grâce au modèle Llama3. Ces vecteurs de dimension 4096 capturent les nuances sémantiques du texte, permettant des comparaisons précises basées sur la similarité conceptuelle plutôt que sur la simple correspondance de mots.

3.4 Innovations Techniques

3.4.1 Traitement et Segmentation des Données

La gestion efficace des données extraites constitue un défi majeur dans notre architecture RAG. Les documents Markdown, bien que correctement structurés, sont souvent trop volumineux pour être traités directement par les modèles de langage. Pour résoudre ce problème, nous avons implémenté un système de découpage intelligent utilisant `MarkdownTextSplitter` de Langchain.

Cette approche permet de diviser les documents en segments (chunks) tout en préservant leur cohérence sémantique. Le système maintient un recouvrement (overlap) entre les segments consécutifs, garantissant ainsi la conservation du contexte global. Cette méthode s'est révélée particulièrement efficace pour respecter les contraintes de tokens des modèles tout en préservant l'intégrité de l'information.

3.4.2 Enrichissement par Métadonnées

Chaque chunk généré est enrichi avec un ensemble de métadonnées cruciales :

- L'URL source du document original
- La position relative dans le document
- Les informations contextuelles essentielles

Cette structuration permet non seulement la traçabilité des informations mais facilite également la reconstitution du contexte complet lorsque nécessaire.

3.4.3 Architecture de Stockage

Le choix de Chroma DB comme base de données vectorielle représente un élément clé de notre architecture. Cette solution offre des capacités d'indexation et de recherche optimisées pour les vecteurs de haute dimension, garantissant des temps de réponse rapides même sur de grands volumes de données. Le processus complet suit une séquence logique :

1. Prétraitement et segmentation des documents
2. Vectorisation des chunks via Llama3
3. Stockage structuré dans Chroma DB

3.4.4 Gestion Contextuelle

Notre système maintient la cohérence des informations grâce à une gestion sophistiquée des métadonnées. Chaque fragment de texte conserve ses liens avec le document source et son contexte d'origine, permettant une reconstitution fidèle de l'information complète lorsque nécessaire.

3.4.5 Optimisation des Performances

L'architecture a été conçue pour équilibrer précision et rapidité. Le système de chunking intelligent, combiné à l'indexation vectorielle, permet de maintenir des temps de réponse optimaux tout en garantissant la pertinence des résultats. La taille des chunks a été choisie afin d'avoir assez d'information et assez courts pour éviter des erreurs de token overflow.

3.5 Architecture du RAG

Voici l'architecture finale de notre chatbot :

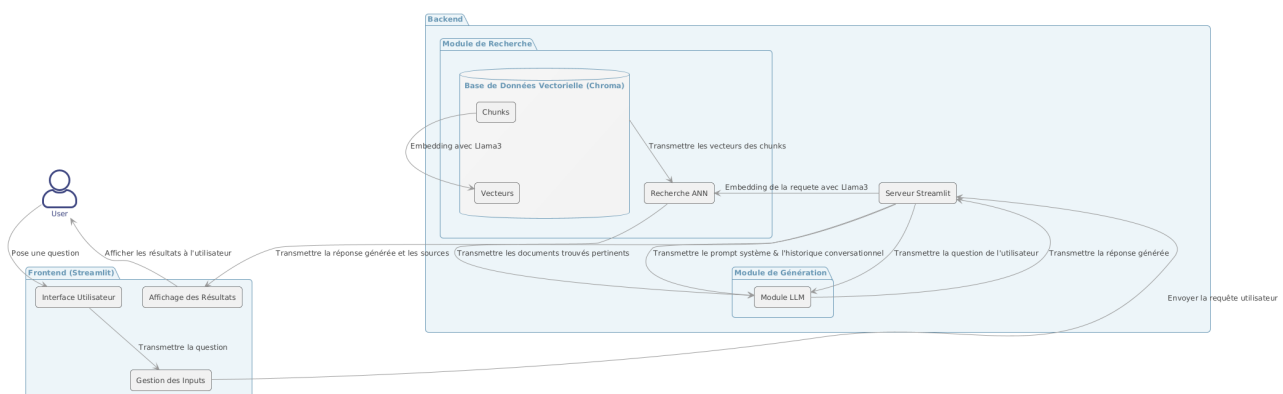


Figure 4: Architecture d'un chatbot basé sur un RAG

4 Implémentation du Chatbot

4.1 Choix du Modèle Local

4.1.1 GPT-3.5 et GPT-2

Notre objectif était d'utiliser un modèle fonctionnant en local, ce qui impliquait de choisir des modèles de petite taille. Nous avons d'abord testé GPT-3.5, mais celui-ci présentait une limitation à 500 tokens, ce qui le rendait quasi inutilisable dans le cadre de ce projet. Nous sommes ensuite passés à GPT-2.0, mais celui-ci posait également des problèmes, notamment parce qu'il avait été principalement entraîné en anglais. Or, l'UQAC fournit les fichiers de documentation en français, ce qui entraînait des difficultés de compréhension et de génération.

4.1.2 Modèles Francophones

Nous nous sommes alors tournés vers des modèles spécifiquement développés pour le traitement du français ou, au moins, capables de le comprendre. Cependant, ces modèles étaient dépassés en termes de performances, car ils ne faisaient plus partie des modèles *state-of-the-art* (ex : FlauBERT, un dérivé de BERT).

4.1.3 DeepSeek 8B

Avec l'apparition de DeepSeek, nous avons migré vers ce LLM, car il permettait d'obtenir des réponses de meilleure qualité et en français. Cependant, nous avons encore rencontré des problèmes : DeepSeek ignorait parfois les instructions et générait des réponses aléatoires.

DeepSeek présente également des problèmes d'affichage, car il s'agit d'un modèle qui "réfléchit". Cette réflexion est générée à l'intérieur de balises `<think>`, mais celles-ci ne sont pas toujours bien générées, ce qui peut entraîner l'affichage du raisonnement directement dans la réponse présentée à l'utilisateur.

4.1.4 Choix Final : Llama 3.0

Après plusieurs tests comparatifs, nous avons retenu LLaMA 3.0. Il est utilisé à la fois pour les embeddings et la génération, ce qui évite de charger plusieurs modèles en mémoire. Ses réponses sont plus pertinentes que celles des autres modèles testés. Sa fenêtre de 4 000 tokens permet également une gestion efficace de la mémoire contextuelle.

4.2 Interface Utilisateur

Pour l'Interface Utilisateur, nous avons utilisé Streamlit, car nous codons en Python. Cette solution permet de générer une interface web sans changer de langage, assurant ainsi une cohérence de langage dans le projet.

Notre objectif initial était de concevoir une interface visuellement proche des pages de l'UQAC. Cependant, nous avons rencontré plusieurs difficultés, notamment avec le thème par défaut de Streamlit, qui reste sombre à moins que l'utilisateur ne le modifie lui-même, ainsi qu'avec le header de Streamlit, qui masque notre propre en-tête, cachant des informations que nous souhaitons donner à l'utilisateur.

Face à ces contraintes, nous avons finalement choisi de conserver une interface Streamlit simple, sans ajout de CSS personnalisé pour que l'utilisateur ne perde pas d'information et qu'il n'a "rien à faire" d'autres que de poser des questions au chatbot.



Figure 5: Exemple de page du site de l'UQAC

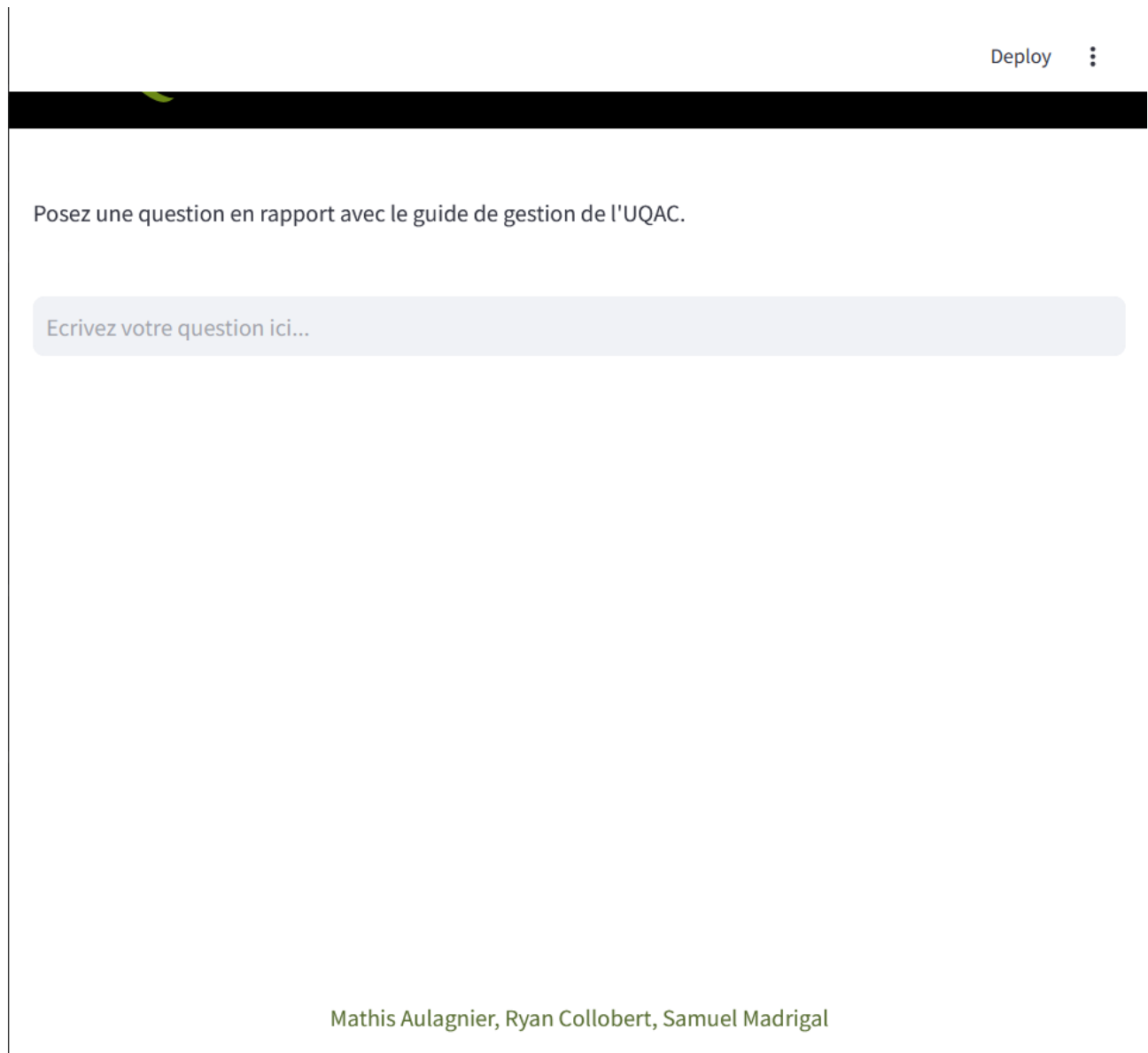


Figure 6: Version du chatbot dans le style de l'UQAC

Nous pouvons clairement constater que notre header ainsi que toutes ces informations sont cachées pour l'utilisateur.



Figure 7: Version finale du chatbot

Dans cette dernière version de notre Chatbot, nous avons également ajouté une option pour effacer tout l'historique conversationnel afin de repartir de zéros, pour, entre autres, obtenir de meilleurs résultats pour des recherches qui n'ont rien en commun.

5 Résultats

5.1 Performance du Système

L'application finale utilise Llama 3 (8B paramètres, nécessitant 8GB de RAM) ainsi qu'une base de données Chroma DB (occupant 1GB de RAM). Les autres composants de l'application ont un impact négligeable sur les performances. Ainsi, lors de son exécution, l'application consomme un total d'environ 9GB de RAM.

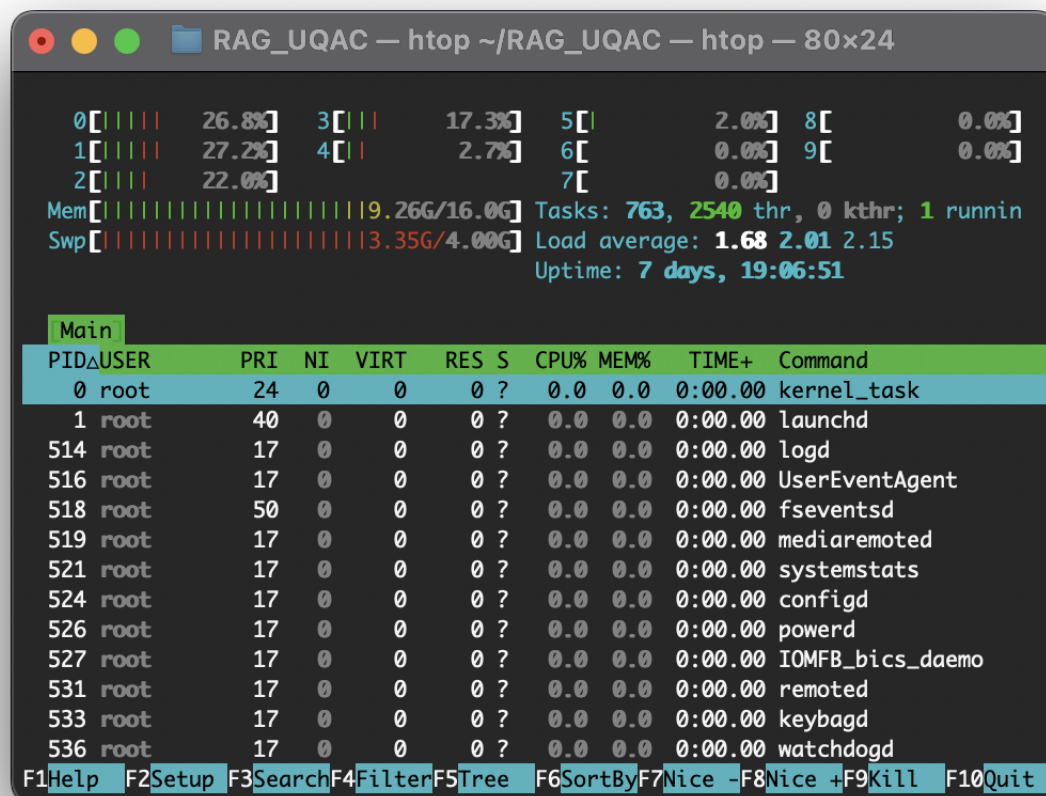


Figure 8: Performances de l'application

5.2 Exemples de Conversations

Un exemple de conversation avec le chatbot est disponible à la fin de ce rapport.

6 Défis et Apprentissages

6.1 Problèmes Rencontrés

Au cours de ce projet, nous avons été confrontés à plusieurs défis techniques et conceptuels :

- **Problèmes de scraping** : La première version du scraping ne récupérait pas les bons fichiers, ce qui a nécessité des ajustements. De plus, notre version finale a entraîné un bannissement temporaire du site de l'UQAC en raison d'un trop grand nombre de requêtes.
- **Problèmes d'affichage du chatbot** : L'intégration du chatbot dans une interface inspirée du style de l'UQAC a posé des difficultés, notamment avec le thème de Streamlit, qui restait sombre par défaut ainsi que le header de Streamlit, qui masquait notre propre en-tête.
- **Réponses en anglais du LLM** : Certains modèles de langage répondaient en anglais alors que nous avions besoin de réponses en français.
- **Gestion des flux de réponse** : Notre approche reposait sur deux flux de sortie : les réponses aux questions et les sources associées. Cependant, cela entraînait un problème d'affichage, l'une des deux réponses n'apparaissant pas toujours correctement.
- **Coupures dans les réponses** : Certaines réponses étaient tronquées en raison de la limite de tokens imposée par certains modèles de langage.
- **Gestion de la mémoire** : La limite du nombre de tokens nous empêchait de conserver l'intégralité de la conversation en mémoire.
- **Besoins en contexte des LLMs** : Certains modèles nécessitaient plus de contexte que d'autres pour fournir des réponses pertinentes, ce qui nous a poussés à ajuster la manière dont nous leur fournissions les données.

6.2 Solutions Apportées

- **Scraping** :
 - **Version 1** : Remplacement du scraping initial par l'utilisation de `crawl4ai`.
 - **Version 2** : Changement d'adresse IP à chaque bannissement pour contourner les restrictions du site de l'UQAC.
- **Interface** : Mise en pause du développement de l'interface pour prioriser d'autres aspects critiques du projet.
- **Gestion des réponses en anglais** :
 - **Changement de LLM** : Passage de GPT-3.5 à DeepSeek (qui répond nativement en français), puis à Llama 3 (méthode hybride).
 - **Méthode hybride pour Llama 3** :
 1. Première requête pour obtenir la réponse avec les sources (générées en anglais).
 2. Seconde requête pour traduire la réponse en français.

- **Gestion du multi-flux** : Le code suivant permet de récupérer simultanément la réponse et ses sources :

```
qa_chain = ConversationalRetrievalChain.from_llm(  
    llm=llm,  
    retriever=retriever,  
    memory=memory,  
    return_source_documents=True,  
    output_key="answer"  
)
```

Ainsi, la réponse du LLM est récupérée via `output_key="answer"` et les sources via `return_source_documents=True`, ce qui permet de traiter les deux flux simultanément.

- **Limite de tokens et gestion de la mise en mémoire de l'historique** : Passage de GPT-3.5 (capacité de 500 tokens) à DeepSeek (capacité de 8000 tokens), puis à Llama 3 (capacité de 4000 tokens). Cette dernière limitation est compensée par une gestion optimisée de la mémoire.

En raison de la limite de tokens, nous utilisons Llama3 pour générer un résumé de la conversation. Ce résumé est ensuite injecté dans le prompt système afin de fournir un contexte historique au modèle, sans nécessiter un rechargement complet de la base de données, ce qui serait trop coûteux en termes de temps.

De plus, pour maintenir un historique persistant entre différents utilisateurs, nous enregistrons les conversations en morceaux (chunks), similaires aux sources HTML de l'UQAC, dans la base de données. Cette approche, s'appuie sur l'architecture du RAG, et permet ainsi de conserver l'historique tout en limitant le nombre de tokens utilisés.

- **Optimisation du contexte** : Nous sommes passés de 3 à 10 chunks afin d'élargir le contexte fourni au LLM. La recherche utilise une approche contextuelle via l'embedding et l'utilisation d'un *approximate Nearest Neighbor* (aNN) pour trouver des chunks similaires à la question posée.

Le aNN est plus rapide qu'un kNN et, combiné avec un plus grand nombre de chunks, garantit une réponse pertinente. Dans l'éventualité où ce n'est pas le cas, il est possible de reposer la question afin d'essayer de trouver de meilleurs chunks, ce qui n'est pas réalisable avec un kNN.

6.3 Améliorations Possibles

- **Optimisation du scraping** : Trouver un moyen de limiter la vitesse du scraping afin d'éviter le bannissement du site de l'UQAC, ou d'automatiser un changement d'IP dès qu'un bannissement est détecté.
- **Refonte du frontend** : Abandonner Streamlit pour l'interface utilisateur et développer une page web en utilisant le même langage que celui des sites de l'UQAC. Cela permettrait une séparation plus nette entre le frontend et le backend. Actuellement, avec Streamlit, ces deux parties sont imbriquées dans le même fichier, ce qui "complique" l'architecture du projet pour d'éventuelles évolutions (comme l'implémentation du chatbot sur le site de l'UQAC).
- **Gestion améliorée de la mémoire et du contexte** : Utiliser un package combinant Streamlit et LangChain afin d'optimiser la gestion de la mémoire et du contexte de conversation de manière plus efficace qu'actuellement.

7 Conclusion

7.1 Bilan du projet

Le développement du ChatBot pour l'UQAC a permis d'explorer et de mettre en œuvre une solution basée sur l'architecture RAG (Retrieval-Augmented Generation), offrant une approche hybride entre la récupération d'informations et la génération de réponses contextuelles. Ce projet a nécessité l'intégration de plusieurs technologies avancées, notamment LangChain, Chroma DB, Llama 3 et des techniques de vectorisation sémantique, ce afin d'optimiser la pertinence des réponses fournies.

Tout au long du processus, l'équipe a rencontré et surmonté divers défis, notamment en matière de scraping des données et de gestion du contexte conversationnel du chatbot. Les solutions mises en place, comme le découpage intelligent des documents et l'amélioration de la gestion de l'historique ont permis d'affiner la précision des résultats tout en assurant une utilisation efficace des ressources.

Cependant, des pistes d'amélioration restent envisageables, notamment en matière de gestion du scraping pour éviter les limitations imposées par l'UQAC, ainsi que l'évolution de l'interface utilisateur, qui pourrait être davantage personnalisée pour s'intégrer plus harmonieusement aux plateformes de l'université.

En définitive, ce projet a permis non seulement d'approfondir nos connaissances, mais aussi de démontrer la faisabilité d'une solution automatisée pour la consultation du manuel de gestion de l'UQAC. Cette preuve de concept constitue ainsi une base solide pour une éventuelle mise en production et une généralisation du chatbot au sein de l'université.

Malheureusement, GPT-4, ayant la capacité de rechercher sur internet, il peut consulter les sources directement sur le site de l'UQAC et formuler des réponses de manière plus efficace et pertinente que notre modèle. Cela permettrait également d'éviter d'héberger notre propre LLM.

7.2 Sources

Voici les sources utilisées lors de ce projet :

- <https://github.com/unclecode/crawl4ai?tab=readme-ov-file>
- <https://docs.crawl4ai.com/>
- <https://www.youtube.com/watch?v=T-D1OfcDW1M>
- <https://www.youtube.com/watch?v=Kvf-Pr1LRe4>
- <https://www.langchain.com/>
- <https://ollama.com/>
- <https://huggingface.co/>
- <https://www.uqac.ca/mgestion/>
- <https://streamlit.io/>

Chatbot spécialisé de la gestion à l'UQAC

Posez une question en rapport avec le guide de gestion de l'UQAC.

Nombre de documents dans la collection : 5228

Historique de la conversation :

Vous: Comment débiter une gestion de projet avec des documents universitaire ?

Chatbot: 1. Comment commencer une gestion de projet à l'aide des documents universitaires ?

Pour répondre à cette question, considérons plusieurs étapes clés :

1. Familiarisez-vous avec les exigences et objectifs du projet en utilisant des documents universitaires appropriés tels que les propositions de projets ou des articles de recherche. Ces documents doivent décrire l'étendue du projet, ses buts et les résultats attendus.
2. Identifiez les tâches nécessaires pour terminer le projet à partir des informations provenant des documents. Ceci peut inclure la définition de bornes spécifiques, la définition des livraisons pour chaque tâche et l'établissement d'un calendrier pour leur achèvement.
3. Développez un plan de projet exhaustif qui décrit la stratégie générale, les tâches, les rôles, les responsabilités, les calendriers et les ressources nécessaires pour terminer le projet avec succès. Cela peut être basé sur des méthodologies de gestion de projet communément utilisées telles que Agile ou Waterfall.
4. Communiquez clairement le plan du projet à tous les intervenants pertinents, y compris les membres de l'équipe de projet, les professeurs et l'administration universitaire. Cela assure que tout le monde impliqué comprend ses rôles, ses responsabilités et les attentes en matière de projet.
5. Suivez et contrôlez la progression du projet en suivant régulièrement la terminaison des tâches, les respectations des délais et l'adresse de tout problème ou obstacle qui peut survenir lors de son exécution.
6. Documentez toutes les activités, décisions, modifications et résultats du projet pour assurer la responsabilité et la transparence. Cela vous aidera à préparer un rapport final ou une présentation au terme du projet.

7. Réfléchissez sur les succès et les zones d'amélioration du projet, en utilisant cette connaissance pour informer vos efforts futurs de gestion de projet.

Des ressources utiles pour votre voyage dans la gestion de projet comprennent :

- Le Guide de la Commission de toponymie du Québec (pour les conventions de nomenclature liées à vos recherches)
- Le Grand dictionnaire terminologique de l'Office québécois de la langue Française (pour des renseignements terminologiques et linguistiques)
- Règlement relatif à la mission, à la vision et aux valeurs de l'UQAC (pour comprendre la mission et les valeurs de l'université)
- Procédure relative à la nomination (ou au renouvellement) d'une Personne Responsable d'une unité de recherche ou de recherche-crédation, Politique relative aux contrats et commandites de recherche, et Politique relative aux subventions de recherche (pour des conseils en matière de recherche et de gestion de projet à l'échelon de l'université)

Votre question :

Quelles sont les politiques d'admission ?

Réponse générée :

Politiques d'admission :

1. Examiner les exigences et objectifs du processus de demande en utilisant des documents académiques pertinents tels que des propositions ou articles de recherche décrivant le cadre, les objectifs et les résultats attendus du projet.
2. Identifier les tâches nécessaires à partir des documents, y compris la définition de buts spécifiques, l'établissement de dates limites pour chaque tâche et la création d'une chronologie de leur accomplissement.
3. Développer un plan de projet exhaustif détaillant la stratégie globale, les tâches, les rôles, les responsabilités, les calendriers et les ressources nécessaires pour terminer le projet avec succès. Ceci peut être fondé sur des méthodologies courantes de gestion de projets telles que l'Agile ou Waterfall.
4. Communiquer clairement le plan de projet à tous les parties prenantes pertinentes, y compris les membres de l'équipe, les professeurs et l'administration universitaire, pour assurer que tout le monde impliqué comprend son rôle, ses responsabilités et ses attentes en matière de projet.
5. Suivre la progression du projet en contrôlant régulièrement la complétion des tâches, l'adhésion aux dates limites et l'adressage de tout problème ou obstacle qui pourrait surgir lors de son exécution.
6. Documenter toutes les activités, décisions, modifications, résultats et autres informations liées au projet pour assurer la responsabilité et la transparence. Ceci aidera à préparer un rapport final ou une présentation à l'achèvement du projet.

7. Réfléchir sur les succès et les zones d'amélioration du projet, en utilisant cette connaissance pour informer les efforts futurs de gestion de projet.

Des ressources supplémentaires qui peuvent être utiles dans votre voyage de la gestion de projet comprennent :

- La Guide de la Commission de toponymie du Québec (pour les conventions liées à vos recherches de nommage)
- Le Grand dictionnaire terminologique de l'Office québécois de la langue Française (pour les informations sur les termes et linguistiques)
- La Politique Mission, Vision et Valeurs de l'UQAC (pour comprendre la mission et les valeurs de l'université)
- Les Procédures pour la Nomination (ou le Renouvellement) d'une Personne Responsable d'un Groupe de Recherche, les Contrats et Commissions de Recherche et les Bourses de Recherche (pour des conseils sur la recherche et la gestion de projet à l'échelle universitaire).

Sources utilisées :

Document 1

Contenu du chunk :

Dans la directive sur les sports élités, il est indiqué que la Réglementation du Recteurat de l'Université sera adoptée le 24 avril 2023 et révisée tous les trois ans. Le doyen ou la décanne des études est responsable de cette directive, avec conseil de la Direction des Activités Sportives et des Installations.

Le texte ne répond pas directement à une question car il s'agit d'une collection de règles et de lignes directrices plutôt qu'une réponse à une inquiétude spécifique. Si vous avez une question particulière sur ces règles ou ces lignes directrices, je serais heureux de vous aider à clarifier n'importe quel point!

- **Source :** Réponse déjà générée par le chatbot dans une autre conversation

Document 2

Contenu du chunk :

2. Si l'éducation préalable d'un candidat ne répond pas aux exigences du programme, il peut être obligé de suivre une propédeutique (cours préparatoire) pour être admis. La durée et les exigences en crédits de ces cours varient en fonction du niveau d'étude :
 - Deuxième cycle (Master): Durée maximale de trois trimestres et un maximum de 30 crédits
 - Troisième cycle (Doctorat): Durée maximale de trois trimestres et un maximum de 18 crédits

3. Il y a une directive sur la gouvernance des programmes d'excellence sportive à l'UQAC, qui a été adoptée le 24 avril 2023 par le Rectorat, en vertu de la réglementation. Cette directive est révisée tous les trois ans, et le responsable de cette directive est le Doyen ou les Doyens des études. Les parties prenantes comprennent le Service des activités et des aménagements sportifs (Services des sports et des installations).
- Source : Réponse déjà générée par le chatbot dans une autre conversation

Document 3

 Contenu du chunk :

Procédure relative à l’émission d’attestations

Adoption

Instance | Date | Décision
Conseil d’administration | 27 avril 1982 |
CAD-2980
Modification

Instance | Date | Décision | Commentaires
Conseil d’administration | 22 février 1994 | CAD-5484
Conseil d’administration | 16 avril 2002 | CAD-7197

Révision	Aux trois (3) ans
Responsable	Le ou la registraire
Parties prenantes	

1. Dispositions générales

1.1 Préambule

- Source : <https://www.uqac.ca/mgestion/chapitre-3/reglement-sur-les-programmes-detudes/2575-2/>

Document 4

 Contenu du chunk :

2.2. Principes spécifiques rattachés aux objectifs de la formation continue

En plus des principes généraux rattachés à l'offre, la formation continue doit avoir plus spécifiquement comme objectifs :

- d'être une réponse aux besoins individuels et collectifs, le tout en cohérence avec les milieux et les collectivités desservies;
 - de constituer de la formation qui s'adresse au personnel professionnel, aux personnes gestionnaires et aux autres personnes souhaitant du développement professionnel, un rehaussement de compétences ou une requalification;
 - d'être une porte d'entrée à l'Université pour les organisations, les entreprises et les ordres professionnels en quête de formation et d'accompagnement sur mesure;
 - d'être un incitatif à poursuivre ou à entreprendre des études universitaires.
- Source : <https://www.uqac.ca/mgestion/chapitre-3/reglement-sur-les-programmes-detudes/politique-sur-la-formation-continue/>

Document 5

 Contenu du chunk :

1.3****Champ d'application

La présente politique s'applique à toute personne, notamment le chercheur, l'étudiant, la Personne impliquée dans l'[activité de recherche](#) ou de création, le gestionnaire de fonds, impliquée dans des activités de recherche et de création ou de formation à la recherche et à la création au nom de l'UQAC, et ce, peu importe où ces activités se déroulent et qu'elles aient obtenu ou non un financement. De plus, l'UQAC s'attend à ce que les partenaires, organisations, collaborateurs, etc., qui collaborent à des activités de recherche et de création ou de formation à la recherche et à la création respectent les principes liés à la conduite responsable en recherche et en création, et ainsi se conforment à la présente Politique.

- Source : <https://www.uqac.ca/mgestion/chapitre-3/reglement-relatif-a-la-recherche-et-a-la-creation/politique-relative-a-la-conduite-responsable-en-recherche-et-en-creation/>

Document 6

Contenu du chunk :

Si le secrétaire général ou la secrétaire générale considère la plainte non recevable ou met fin à son traitement suivant l'article 4.3 de la politique, il ou elle informe la personne plaignante des motifs de non-recevabilité.

4.3 Suspension de traitement

Le secrétaire général ou la secrétaire générale peut mettre fin à son examen de la plainte s'il ou si elle estime notamment que :

1. l'objet de la plainte ne relève pas de la présente politique;
 2. la plainte met en cause le bien-fondé d'une politique ou d'une condition d'un programme de l'UQAC;
 3. la plainte est frivole.
- **Source :** <https://www.uqac.ca/mgestion/chapitre-2/reglement-sur-la-mission-et-les-valeurs-de-luqac/politique-linguistique/>

Document 7

Contenu du chunk :

2.4 Droit d'appel

L'étudiante ou l'étudiant qui se croit lésé par l'application de cette procédure pourra en appeler selon les modalités prévues à la procédure relative à la révision de notes.

3. Mise à jour

La présente procédure doit être mise à jour au besoin et minimalement, tous les trois (3) ans. La mise à jour est adoptée par la Commission des études, de la recherche et de la création.

4. Dispositions finales

La présente procédure entre en vigueur à compter de son adoption par la Commission des études de l'Université.

- **Source :** <https://www.uqac.ca/mgestion/chapitre-3/reglement-sur-les-programmes-detudes/procedure-relative-aux-regles-de-presentation-et-de-redaction-des-travaux-des-etudiants/>

Document 8

 Contenu du chunk :

1.4 Champ d'application

La présente politique s'applique aux étudiantes et aux étudiants des programmes d'études de cycles supérieurs, boursiers ou non, qui, tout en réalisant leur programme d'études à temps complet, exercent un emploi rémunéré à l'Université à même son budget de fonctionnement, sauf les subventions de recherche accordées aux professeurs et professeures par voie de concours à même le Fonds institutionnel de recherche.

1.5 Responsable de l'application

Le vice-recteur ou la vice-rectrice aux études, à la formation et à la réussite est responsable de son application.

1.6 Définitions

« Études de cycles supérieurs » : programmes d'études de deuxième et de troisième cycles.

- Source : <https://www.uqac.ca/mgestion/chapitre-3/reglement-sur-les-programmes-detudes/politique-relative-a-lencadrement-financier-des-etudiants-des-programmes-detudes-de-cycles-superieurs/>

Document 9

 Contenu du chunk :

3.1.1 Objectifs

Les objectifs de chaque programme d'études sont définis selon les finalités de leur [cycle](#) et prennent appui également sur les objectifs de programmes de niveaux ou de cycles antérieurs requis à des fins d'[admission](#).

3.1.2 Conditions d'admission

- Source : <https://www.uqac.ca/mgestion/chapitre-3/reglement-sur-les-programmes-detudes/>

Document 10

 Contenu du chunk :

Le *Rapport d'évaluation institutionnelle* comprend le(s) CPAP ainsi que les éléments manquants identifiés par le Décanat des études en fonction des critères et indicateurs énumérés dans la [Politique relative à l'évaluation institutionnelle de programme](#), soient :

- **Pertinence******: adaptation du programme par rapport aux besoins institutionnels, interinstitutionnels et sociétaux.
- **Efficacité******: capacité du programme à recruter, à sélectionner et à faire progresser les étudiants et étudiantes vers la réussite et la diplomation.
- **Cohérence******: qualité de la structuration du programme, progression des cours pour permettre l'atteinte des objectifs ou des compétences.
- **Source** : <https://www.uqac.ca/mgestion/chapitre-3/reglement-sur-les-programmes-detudes/procedure-relative-a-levaluation-institutionnelle-de-programme/>