Projet LLM 2025/2026

Daniel Vaz Correia

Théo Koessler

Introduction

Dans le cadre de notre troisième année de BUT, nous avions pour objectif de transformer un CV traditionnel en un chatbot interactif. En effet, plutôt que de lire un document PDF, le recruteur pourrait ainsi discuter directement avec un robot qui connaît notre parcours, compétences et expériences.

Pour réaliser cela, nous n'avons pas seulement utilisé un modèle de LLM. Nous nous sommes servis d'une technique appelée RAG (Retrieval Augmented Generation). Cela permet de donner accès à nos données privées à l'intelligence artificielle.

RAG

Les modèles d'IA sont très puissants mais ne connaissent pas nos informations personnelles (stage, notes, projets...). Le principe du RAG fonctionne en deux grandes étapes que j'ai implémentées dans mon code :

L'Indexation des données

D'abord, il y a l'indexation : il faut préparer les données et les organiser dans un espace dédié que nous nommerons « data ». J'ai rédigé mon parcours dans des fichiers Markdown séparés par thèmes : CV.md, Stage.md, Bilan.md. C'est là que le chatbot va devoir trouver les informations me concernant.

Ensuite, une fois nos informations rangées, le système doit pouvoir les lire. Le script indexer.py découpe ces fichiers en petits morceaux (chunks) afin d'avoir une meilleure organisation de l'information. Ces textes sont ensuite transformés en vecteurs (des suites de nombres) qui seront interprétables pour la machine. Enfin, les données sont envoyées vers Upstash, ma base de données vectorielle en ligne.

Architecture et Développement

Une fois les données en ligne, nous avons développé deux scripts principaux pour faire fonctionner le chatbot.

L'Intelligence du système (agent.py)

Le fichier agent.py constitue le moteur intelligent de mon application. Il fait le lien entre l'utilisateur, la base de données et l'intelligence artificielle.

Concrètement, ce script exécute deux actions principales :

- Il utilise une fonction de recherche (tool) pour interroger Upstash et récupérer les passages du portfolio qui correspondent à la question posée.
- Il intègre ces informations dans le prompt qu'il envoie au modèle GPT-4.1-nano.

C'est cette étape cruciale qui force l'IA à utiliser uniquement les données issues de ‘data’ pour construire sa réponse, garantissant ainsi qu'elle ne nous donne pas d'informations fausses.

L'Interface Utilisateur (app.py)

Le fichier app.py représente la partie visible du projet : c'est l'interface utilisateur. Développé avec la librairie Streamlit, ce script transforme le code Python en une application web interactive.

Il a pour mission de gérer l'affichage de la conversation (les bulles de chat), de récupérer la question posée par le recruteur via une zone de saisie, et de maintenir l'historique des échanges à l'écran pour que la discussion reste fluide.

Conclusion

Pour conclure, ce projet nous a permis de passer de la théorie à la pratique en créant une véritable application d'Intelligence Artificielle.

Nous avons réussi à mettre en place une architecture RAG complète : partir de données brutes (fichiers Markdown), les stocker dans une mémoire vectorielle (Upstash) et les connecter à un modèle de langage (OpenAI) via une interface web (Streamlit).

Ce travail nous a permis de comprendre concrètement le fonctionnement des outils modernes de développement et de maîtriser la chaîne complète de traitement de la donnée, de l'indexation jusqu'à la génération de la réponse.