RAPPORT TECHNIQUE - Assistant Portfolio IA

Assistant Conversationnel Portfolio

Auteur : Hoda Kharbouche
Technologies : Python, Streamlit, OpenAI, Upstash

1. OBJECTIF DU PROJET

Ce projet permet de transformer un CV classique en assistant virtuel intelligent. Au lieu de lire un document statique, les recruteurs peuvent interagir en posant des questions naturelles et obtenir des réponses personnalisées basées sur les données réelles du portfolio.

Fonctionnalités principales

L'application offre plusieurs fonctionnalités clés. D'abord, elle permet une recherche sémantique : l'utilisateur pose des questions en langage naturel et reçoit des réponses précises. Ensuite, toutes les réponses sont sourcées, c'est-à-dire qu'elles s'appuient uniquement sur les données du portfolio, garantissant ainsi leur véracité. Le système propose également des suggestions contextuelles pour approfondir certains sujets. Enfin, l'historique des conversations est sauvegardé, permettant un échange fluide qui se souvient du contexte.

2. ARCHITECTURE RAG

Le projet utilise une architecture RAG (Retrieval-Augmented Generation). Cette approche combine deux mécanismes : la recherche d'information pertinente et la génération de texte. Concrètement, cela permet de donner accès à des données privées (le portfolio) à un LLM générique, sans avoir besoin de réentraîner le modèle.

Flux de traitement d'une requête

Quand un utilisateur pose une question, le traitement se déroule en 6 étapes séquentielles. Tout commence par la capture de la question dans l'interface Streamlit, où le système vérifie l'existence d'une session utilisateur via un UUID. Ensuite, la question est transformée en embedding (vecteur numérique de 1536 dimensions) grâce à l'API OpenAI. Cette étape permet de la comparer mathématiquement avec les textes du CV.

Le système interroge alors Upstash Vector pour trouver les 5 paragraphes du CV ayant la plus grande similarité cosinus avec la question. Ces extraits sont ensuite assemblés avec les instructions système et la question pour former un prompt enrichi. Ce prompt est envoyé à GPT-4.1-nano qui génère une réponse naturelle en français, à la première personne. Enfin, l'échange complet est sauvegardé dans Upstash Redis pour maintenir le contexte conversationnel.

3. COMPOSANTS DU PROJET

indexer.py - Ingestion des données

Ce module est responsable de l'ingestion initiale des données. Il lit tous les fichiers Markdown du dossier Data/, les découpe en chunks via le chunker, génère les embeddings correspondants, puis envoie le tout vers Upstash Vector. Ce script doit être exécuté manuellement après chaque modification du contenu du portfolio.

chunker.py - Segmentation intelligente

Le chunker découpe les documents longs en segments cohérents. Cette étape est critique car elle impacte directement la qualité de la recherche. La stratégie utilisée consiste à créer un nouveau chunk à chaque titre Markdown (# ou ##), garantissant ainsi que chaque segment correspond à une idée complète. La taille cible est de 200 à 500 mots par chunk.

agent.py - Configuration comportementale

Ce fichier définit le comportement de l'IA via le Prompt Système. On y trouve les instructions permanentes : identité ('Tu es la version virtuelle de Hoda Kharbouche'), tonalité (professionnel, enthousiaste), règles linguistiques (toujours à la première personne), et format de réponse (listes à puces pour la clarté). Pour modifier le ton de l'assistant, il suffit d'éditer la variable 'instructions' dans ce fichier.

agent_tool.py - Recherche vectorielle

Cette fonction permet à l'IA d'interroger la base de connaissances. Elle transforme la question en embedding, effectue une requête de similarité cosinus sur Upstash Vector avec un top_k de 5, puis retourne les textes des résultats. Le temps de recherche moyen est inférieur à 100 millisecondes.

app.py - Interface Streamlit

C'est le fichier principal qui gère l'interface utilisateur. Il orchestre plusieurs fonctionnalités : la gestion des sessions via UUID et localStorage, le chargement et la sauvegarde de l'historique dans Redis (avec un TTL de 7 jours), la détection de mots-clés pour afficher des suggestions contextuelles (mode drill-down), et l'insertion automatique d'un lien LinkedIn tous les 5 messages.

RAPPORT TECHNIQUE - Assistant Portfolio IA

4. INFRASTRUCTURE ET INSTALLATION

Services Cloud

Le projet s'appuie sur trois services Cloud. Upstash Vector stocke les embeddings avec leurs textes associés. C'est une solution serverless qui offre des performances de recherche inférieures à 100ms et un tier gratuit jusqu'à 10 000 requêtes par mois. Upstash Redis gère la persistance des conversations avec un TTL de 7 jours. Enfin, l'API OpenAI est utilisée pour deux opérations : la génération d'embeddings (modèle text-embedding-3-small) et la génération de réponses (GPT-4.1-nano).

Dépendances Python

Le projet nécessite plusieurs librairies Python : openai-agents pour l'orchestration des agents, upstash-vector et upstash-redis pour les connexions aux bases de données, streamlit pour l'interface web, et python-dotenv pour la gestion sécurisée des secrets.

Configuration des secrets (.env)

Il faut créer un fichier .env à la racine du projet contenant 5 clés d'accès. OPENAI_API_KEY (commence par 'sk-') permet d'utiliser ChatGPT. UPSTASH_VECTOR_REST_URL et UPSTASH_VECTOR_REST_TOKEN donnent accès à la base vectorielle. UPSTASH_REDIS_REST_URL et UPSTASH_REDIS_REST_TOKEN permettent la connexion à Redis. Ce fichier ne doit jamais être partagé publiquement ou commité sur Git.

Installation complète

L'installation se fait en 7 étapes. D'abord, télécharger le projet sur votre ordinateur. Ensuite, créer un environnement virtuel avec 'python -m venv .venv' pour isoler le projet. L'activer ensuite (source .venv/bin/activate sur Mac/Linux, ou .venv\Scripts\activate sur Windows). Le prompt affiche (.venv) quand c'est activé.

Installer ensuite toutes les librairies avec 'pip install -r requirements.txt'. Créer le fichier .env avec les 5 clés mentionnées précédemment. Préparer la base de données en lançant 'python src/indexer.py' et vérifier qu'un message de succès s'affiche. Enfin, lancer l'application avec 'streamlit run src/app.py' et ouvrir http://localhost:8501 dans le navigateur.

5. UTILISATION QUOTIDIENNE

Pour démarrer l'application, lancer 'streamlit run src/app.py' dans le terminal avec l'environnement activé. Une page web s'ouvre automatiquement sur http://localhost:8501.

L'interface présente une zone de chat au centre pour l'historique des messages, un champ de saisie en bas pour poser les questions, des boutons de suggestions qui apparaissent selon le contexte, et un bouton pour réinitialiser l'historique en haut.

On peut par exemple poser une question générale comme 'Présente-toi en quelques mots' pour obtenir une vue d'ensemble, ou une question technique comme 'Quels sont tes frameworks Python ?' pour une liste ciblée. Quand on mentionne un projet spécifique, le système affiche automatiquement 3 boutons de drill-down (Technologies, Description, Apports).

6. MAINTENANCE DU PROJET

Mise à jour du contenu

Pour ajouter ou modifier une expérience, un projet ou une compétence, il suffit d'éditer les fichiers .md dans le dossier Data/. Il est important de respecter la structure Markdown avec des titres (#, ##). Une fois les modifications sauvegardées, lancer 'python src/indexer.py' pour réindexer. Le système affiche un message de succès une fois terminé. On peut ensuite tester en posant une question sur le nouveau contenu.

Modification du comportement

Pour changer le ton de l'IA ou ses règles de réponse, ouvrir le fichier src/agent.py et localiser la variable 'instructions'. Modifier le texte selon les besoins (par exemple, passer d'un ton enthousiaste à un ton formel), puis sauvegarder. Il suffit ensuite de relancer l'application. Aucune réindexation n'est nécessaire pour ce type de modification.

Résolution de problèmes

Si l'IA répond systématiquement 'Je ne sais pas', cela signifie généralement que l'indexation n'a pas été lancée ou a échoué. La solution est de relancer 'python src/indexer.py' et de vérifier les logs.

En cas d'erreur d'authentification ou de timeout de connexion, vérifier que les 5 clés dans le fichier .env sont correctes. Tester également la connexion internet.

Si l'historique ne se charge pas entre les sessions, cela peut provenir de credentials Redis invalides ou d'un problème de cache. Vérifier UPSTASH_REDIS_REST_URL et TOKEN dans .env, et essayer de vider le cache du navigateur (Ctrl+Shift+R).

Évolutions possibles

Le projet peut être enrichi de plusieurs manières. On pourrait ajouter un support multilingue avec détection automatique de la langue. Un système d'analytics permettrait de tracker les questions les plus fréquentes pour identifier les sections du portfolio à améliorer. Une fonctionnalité d'export permettrait de télécharger la conversation en PDF. On pourrait également intégrer une authentification pour des analytics avancés par utilisateur, ou ajouter une synthèse vocale (TTS) pour l'accessibilité.