CY IA : L'intelligence étudiante

Pipeline RAG pour la centralisation intelligente
des données web et mail de CY Tech

Auteurs : Bilal El Biyadi, Yvan Louamba, Ziyad Hadine
Tuteur : Souhila Arib
Spécialité : Ing 3 IA

15 Janvier 2025

Sommaire

1 Introduction

Dans le contexte d'un enseignement supérieur , disposer d'un accès rapide et fiable à l'information est devenu indispensable pour les étudiants. L'école d'ingénieur CYTECH met à leur disposition un site web particulièrement riche, regroupant presque un millier de pages de contenu. En parallèle, les boîtes mail institutionnelles des étudiants sont quotidiennement saturées de messages contenant des informations cruciales sur les plans administratif, pédagogique ou logistique.

Cette densité informationnelle, bien qu'indispensable au bon fonctionnement de la vie étudiante, devient paradoxalement un facteur de désorganisation et de surcharge cognitive. Il devient difficile de savoir où chercher une information, de ne pas en oublier une importante, ou tout simplement de gagner du temps dans la gestion quotidienne de ses démarches scolaires. L'usage des moteurs de recherche classiques (Ctrl+F, recherche mot-clé dans la boîte mail, navigation manuelle sur le site) atteint rapidement ses limites face à cette masse de données non structurées.

Dans ce contexte, ce projet est né d'un constat simple : les étudiants ont besoin d'un assistant intelligent, capable de centraliser les informations issues à la fois du site internet de l'école et de leur boîte mail, et de les restituer de façon pertinente, contextualisée et naturelle. L'objectif du projet est donc de concevoir une application 2-en-1, reposant sur l'intelligence artificielle de type Retrieval-Augmented Generation (RAG), pour offrir une expérience de recherche optimale. Cette solution vise à réduire la dispersion de l'information, à améliorer l'efficacité des recherches et à prévenir les oublis critiques liés à une mauvaise visibilité des contenus. Elle s'inscrit ainsi dans une logique de gain de temps, de sérénité et d'aide à la réussite académique.

2 Définitions et Terminologies Clés

2.1 LLM (Large Language Model)

Un Large Language Model (LLM) est un modèle de traitement du langage naturel (NLP) entraîné sur d'immenses quantités de texte. Grâce à l'apprentissage profond (deep learning), ces modèles parviennent à comprendre (dans une certaine mesure) les requêtes des utilisateurs. En effet, ils sont capables de cerner la structure, la grammaire et le sens général de la langue. En outre, ils sont capables de générer du texte cohérent, produire des réponses, des résumés ou des traductions en s'appuyant sur ce qu'ils ont appris.

Concrètement, ils acquièrent une compréhension statistique du langage en analysant et en mémorisant des patterns, ce qui leur permet de générer des phrases cohérentes ou d'analyser et de classer du contenu. Un LLM assimile les règles de syntaxe, de grammaire et de sémantique à partir de ce vaste corpus d'entraînement, ce qui se traduit par la capacité de répondre à des questions ou de rédiger des textes avec une fluidité remarquable. Toutefois, ces modèles s'appuient sur une base de connaissances fondamentalement figée à la date où ils ont été entraînés. Dès lors, ils ne peuvent pas, sans mise à jour spécifique (réentraînement ou fine-tuning), intégrer les faits ou informations postérieurs à leur date de coupure. De plus, comme mentionné précédemment, il arrive parfois qu'un LLM "invente" des faits ou réponde de manière erronée (hallucination), faute de mécanismes internes robustes de vérification. Ainsi, les LLM bénéficient d'une puissance de génération et une polyvalence impressionnantes, mais présentent des limites en termes de données récentes, de fiabilité et d'explicabilité.

2.2 RAG (Retrieval-Augmented Generation)

Le *R*etrieval-Augmented Generation (RAG) vise à pallier certaines limites des LLM en y intégrant un module de recherche capable de récupérer, en temps réel, des informations depuis une base de connaissances externe. Le fonctionnement repose d'abord sur la requête de l'utilisateur, qui déclenche un processus de recherche dans des documents. Les informations les plus adaptées, contextualisées et à jour sont ensuite extraites et transmises à la partie "génération" du modèle. Ainsi, au lieu de s'appuyer uniquement sur un bagage statique appris au préalable, le RAG exploite des ressources externes pour affiner et justifier ses réponses. Cette approche offre une actualisation permanente des contenus, réduit considérablement le risque d'hallucinations (réponses inventées) et permet de mieux tracer et justifier la provenance des informations fournies en les sourçant très clairement. Le RAG se révèle donc très efficace pour les entreprises ou organisations souhaitant utiliser leurs données internes sans devoir entièrement réentraîner un grand modèle pour chaque mise à jour.

2.3 Pipeline RAG

2.3.1 Le Concept d'Embedding

Les embeddings jouent un rôle crucial dans la pipeline RAG (Retrieval-Augmented Generation) en permettant une représentation vectorielle des mots, phrases ou documents dans un espace mathématique multidimensionnel. Contrairement à une représentation traditionnelle basée sur l'orthographe des mots ou leur indexation dans un dictionnaire, un embedding encode chaque entité textuelle sous la forme d'un vecteur de nombres réels. Par exemple, le mot « océan » peut être représenté par un vecteur tel que

$$(0.12, -0.03, 0.95, \dots)$$

Cette représentation vectorielle permet de capturer les relations sémantiques entre les différents éléments textuels. Par exemple, les phrases « J'adore les pommes » et « Je préfère les clémentines » auront des embeddings proches, reflétant

leur similarité thématique, tandis qu'une phrase sur le nucléaire sera spatialement éloignée en raison de son contenu sémantiquement distinct.

L'utilisation des embeddings est essentielle pour le fonctionnement efficace de la pipeline RAG, en particulier pour la phase de récupération des informations pertinentes. Le *Retriever*, première composante de la pipeline, convertit la requête de l'utilisateur en un embedding. Cette transformation permet de comparer la question posée avec les documents de la base de connaissances en mesurant la similarité entre leurs vecteurs correspondants, souvent à l'aide de métriques telles que la distance cosinus ou le produit scalaire. Les documents dont les vecteurs sont les plus proches dans l'espace vectoriel sont alors identifiés comme les plus pertinents.

Cette capacité de comparer les significations plutôt que les simples correspondances lexicales permet une recherche plus intelligente et flexible. Par exemple, pour une requête telle que « Quelle est la recette d'un bon gâteau au chocolat ? », le *Retriever* peut identifier des documents pertinents même si ceux-ci utilisent des termes différents, comme un article de blog sur la pâtisserie, un extrait de livre de recettes ou un tutoriel de cuisine. Les embeddings permettent ainsi de capturer les nuances sémantiques, rendant la recherche plus robuste et contextuellement appropriée.

2.3.2 Étape 1 : Le Retriever (Récupération d'Informations)

La pipeline RAG se compose de deux étapes principales : la récupération d'informations pertinentes (*Retriever*) et la génération de la réponse (*Generator*).

La première étape de la pipeline RAG, le *Retriever*, consiste à transformer la requête de l'utilisateur en un embedding. Cette représentation vectorielle permet de comparer efficacement la question avec les documents de la base de connaissances. En calculant la similarité entre les vecteurs, le *Retriever* identifie les documents les plus pertinents. Par exemple, pour la question « Quelle est la recette d'un bon gâteau au chocolat ? », le *Retriever* pourrait sélectionner des documents tels qu'un article de blog sur la pâtisserie, un extrait de livre de recettes ou un tutoriel de cuisine. Ces documents sont ensuite récupérés en fonction de leur proximité vectorielle avec la requête initiale.

2.3.3 Étape 2 : Le Generator (Génération de la Réponse)

Une fois les documents pertinents récupérés, la deuxième étape implique le *Generator*, généralement un modèle de langage avancé comme GPT. Les textes extraits des documents sélectionnés sont concaténés ou résumés pour créer un contexte riche et pertinent. Le *Generator* utilise ce contexte, en conjonction avec la question de l'utilisateur, pour élaborer une réponse cohérente et précise.

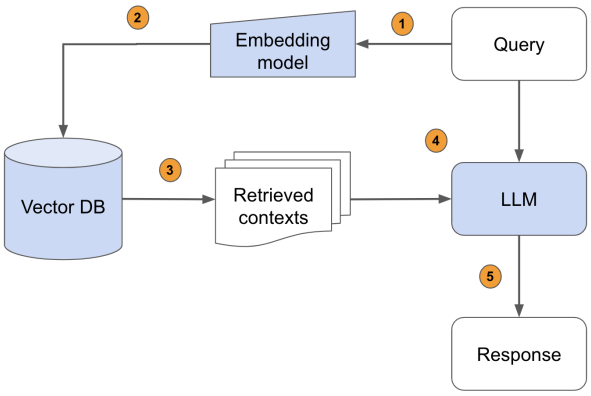

Figure 1: Schéma conceptuel d'une pipeline RAG

2.4 Les techniques de RAG avancés

Lorsqu'on met en place un système RAG (Recherche + IA Générative) de manière « classique », on se contente généralement d'extraire des passages via une simple recherche sémantique, puis de les transmettre au modèle de langage. Bien que cela fournisse déjà un certain niveau de pertinence, on constate souvent que cette approche seule est insuffisante pour couvrir un large éventail de cas d'usage et pour garantir des résultats de haute qualité.

En effet, il est fréquent de rencontrer des problèmes comme des réponses hors contexte, des informations manquantes ou, à l'inverse, une surabondance de contenu qui noie l'information utile. Pour remédier à ces limitations, on déploie des techniques de RAG avancées, qui permettent d'améliorer la précision, la pertinence et la concision des réponses générées.

Il existe de nombreuses stratégies avancées dans ce domaine. Toutefois, pour illustrer les principaux mécanismes d'amélioration, deux d'entre elles ont été utilisé pour le projet : le Reranking et le Multy Query.

Chacune de ces techniques vise un objectif précis, qu'il s'agisse d'affiner le découpage initial, de réordonnancer les résultats renvoyés, ou de compléter la recherche vectorielle par d'autres méthodes lexicales. Leur mise en œuvre fournit à la fois plus de robustesse et de flexibilité à un pipeline RAG, en offrant des moyens de répondre de façon plus fiable aux questions posées.

2.4.1 Reranking

Le *reranking* consiste à ajouter une étape supplémentaire de ré-ordonnancement des documents (ou des segments) renvoyés par le module de recherche sémantique. Concrètement :

- **Utilisation d'un second modèle de langage :** après la recherche initiale basée sur des *embeddings*, un autre modèle, généralement plus spécialisé ou plus finement entraîné, évalue la pertinence de chaque segment au regard de la requête.
- **Combinaison de scores :** ce second modèle ne se limite pas à la dimension sémantique stricte. Il peut par exemple prendre en compte un *score d'affinité linguistique*, c'est-à-dire la proximité lexicale ou la façon dont les mots clés apparaissent dans le segment, pour affiner davantage le classement.
- **Réduction du bruit :** cette étape permet de prioriser les résultats réellement utiles et de mettre à l'écart ceux qui, bien que sémantiquement voisins, n'offrent

pas la meilleure réponse. Le *reranking* agit ainsi comme un filtre intelligent, garantissant un niveau de confiance plus élevé pour les segments les mieux notés.

En complétant la recherche initiale, le *reranking* augmente la pertinence des résultats finaux présentés au module de génération de texte.

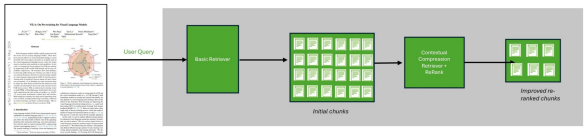

Figure 2: Schéma conceptuel du concept de reranking

2.4.2 MultiQuery

Le *multi-query* complète la recherche sémantique en générant, à partir de la requête initiale, plusieurs reformulations parallèles qui explorent des perspectives lexicales et conceptuelles différentes. Un large modèle de langage produit ainsi diverses paraphrases, chacune soulignant un aspect particulier de la question ; cette diversité élargit la couverture sémantique et augmente la probabilité de récupérer les segments réellement pertinents, même lorsque les mots exacts diffèrent entre la requête et les documents. Les résultats issus de ces requêtes multiples sont ensuite fusionnés : les doublons sont éliminés et un ensemble enrichi, mais cohérent, est transmis aux étapes suivantes (le *reranking* ou directement le module de génération). En multipliant les angles d'interrogation, le *multi-query* améliore le rappel, réduit le risque de manquer une information clé et fournit au générateur de texte un contexte plus exhaustif et nuancé.

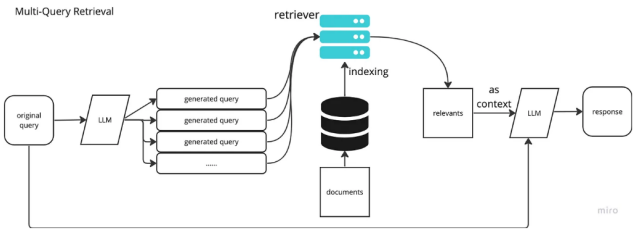

Figure 3: Schéma conceptuel du processus de multi-query

2.5 Pourquoi un RAG plutôt qu'un LLM Basique ?

À la différence d'un LLM basique, qui se limiterait à ses propres connaissances internes, un RAG rend possible une adaptation quasi immédiate à des documents nouveaux ou spécialisés. Il fournit une meilleure robustesse face aux interrogations sur des textes précis, limitant ainsi les risques de réponses fantaisistes ou inexactes. Dans des secteurs comme le juridique ou l'industrie lourde, les textes de référence et réglementations changent régulièrement. Pouvoir mettre à jour la base documentaire sans réentraîner tout un modèle est un atout considérable.

Par ailleurs, le RAG facilite la mise en place de processus de vérification et de traçabilité des sources, offrant ainsi une transparence accrue sur l'origine des informations et renforçant la confiance dans la qualité des données fournies. En outre, cette approche permet de personnaliser le contenu documentaire en fonction des besoins spécifiques des utilisateurs ou des secteurs d'activité.

Enfin, l'architecture modulaire d'un système RAG favorise l'intégration harmonieuse avec d'autres outils d'analyse et de gestion de contenu, ce qui ouvre la voie à des solutions sur mesure et évolutives, adaptées aux exigences toujours changeantes du marché et aux défis spécifiques de chaque industrie.

3 Stack Technique

3.1 Frontend

Pour le développement de la partie frontend de CYIA, le choix a été porté sur Next.js, TypeScript et Tailwind CSS pour leurs avantages complémentaires. Next.js offre des fonctionnalités puissantes comme la génération de pages statiques et dynamiques, essentielles pour optimiser les performances et le référencement. TypeScript garantit une meilleure robustesse du code grâce au typage statique, réduisant ainsi les erreurs lors du développement. Tailwind CSS, quant à lui, permet une conception rapide et réactive grâce à son système de classes utilitaires, tout en maintenant une grande flexibilité pour la personnalisation du design. Le frontend est déployé sur la plateforme Vercel, optimisée pour les applications Next.js.

3.2 Backend

Le backend se divise entre la logique métier et la gestion des données. L'application Node.js (ou les routes internes de Next.js) pilote les tâches d'authentification, de routage et de coordination avec l'infrastructure IA. La base de données relationnelle est hébergée à la fois localement avec PostgreSQL pour le développement, et sur le cloud avec NeonDB pour l'environnement de production.

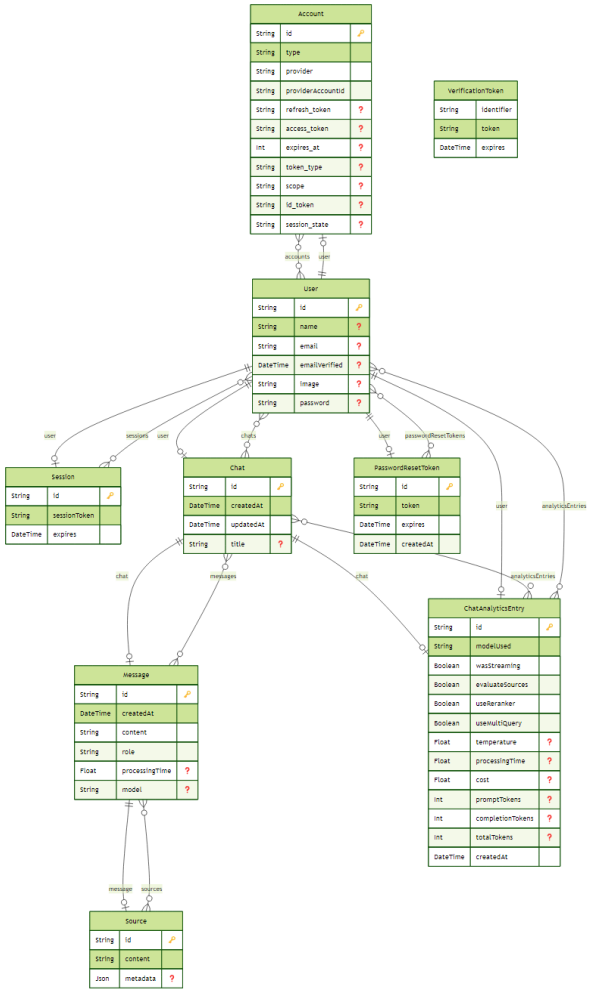

Figure 4: Schéma conceptuel de la base de données

Concernant le schéma de la base de donnée, il contient :

- **Table `User`** : centralise les informations des utilisateurs, avec des champs pour l'identifiant unique, le nom, l'email (unique), l'image de profil, la date de vérification de l'email, ainsi que les relations vers les comptes, sessions et chats associés.
- **Table `Account`** : stocke les informations des comptes liés à l'authentification OAuth (Google, GitHub), incluant les tokens d'accès, le type de fournisseur et l'identifiant utilisateur. Elle est reliée à la table `User`.
- **Table `Session`** : gère les sessions actives des utilisateurs authentifiés, avec un token de session unique, une date d'expiration et une liaison vers l'utilisateur concerné.
- **Table `VerificationToken`** : conserve les jetons de vérification temporaire pour des actions comme la validation d'email ou la réinitialisation de mot de passe. Chaque jeton est associé à un identifiant unique et possède une date d'expiration.
- **Table `Chat`** : enregistre les conversations initiées par les utilisateurs. Chaque chat contient un titre, une date de création/mise à jour, une référence utilisateur et une liste de messages associés.
- **Table `Message`** : représente chaque message dans un chat, avec un contenu textuel, un rôle (utilisateur ou assistant), le temps de traitement éventuel, le modèle utilisé et les sources liées. Elle est liée à un chat.
- **Table `Source`** : relie un message aux sources documentaires utilisées pour générer la réponse. Elle stocke le contenu source ainsi que des métadonnées associées sous forme de JSON (ex. titre, URL, auteur).

3.3 Stack IA

La stack d'intelligence artificielle repose tout d'abord sur les bases de données vectorielles permettant le stockage des embeddings. En environnement local, ChromaDB assure le stockage des embeddings facilitant ainsi le prototypage et les tests hors ligne. En production, ces données sont migrées vers PineconeDB, une solution cloud hautement scalable, optimisée pour la recherche vectorielle à faible latence.

Concernant la génération des embeddings, le modèle `text-embedding-ada-002` d'OpenAI est utilisé en local avec ChromaDB, tandis que le modèle `llama-text-embed-v2` est utilisé dans le cloud avec PineconeDB.

Cette répartition des rôles offre une architecture flexible et évolutive : les ressources allouées au traitement vectoriel (via PineconeDB) peuvent évoluer indépendamment de celles du backend ou de la base relationnelle, simplifiant la maintenance et améliorant la résilience globale du système.

Le cœur de la pipeline RAG a été développé avec Langchain, un framework spécialisé dans la gestion de chaînes de traitement d'IA. Celui-ci orchestre l'ensemble du processus, depuis la récupération de documents pertinents jusqu'à la génération de réponses enrichies. Les modèles de langage (LLM) sont quant à eux accessibles à travers l'API OpenRouter, qui permet de sélectionner dynamiquement, à chaque requête, le modèle le plus adapté au besoin de l'utilisateur (Mistral 8b, Claude 3.7 Sonnet, Deepseek R1, Grok3-mini-beta, GPT-4o, GPT-4 turbo, GPT-4.1, Gemini 2.0 Flash Lite, Qwen 2.5 7B).

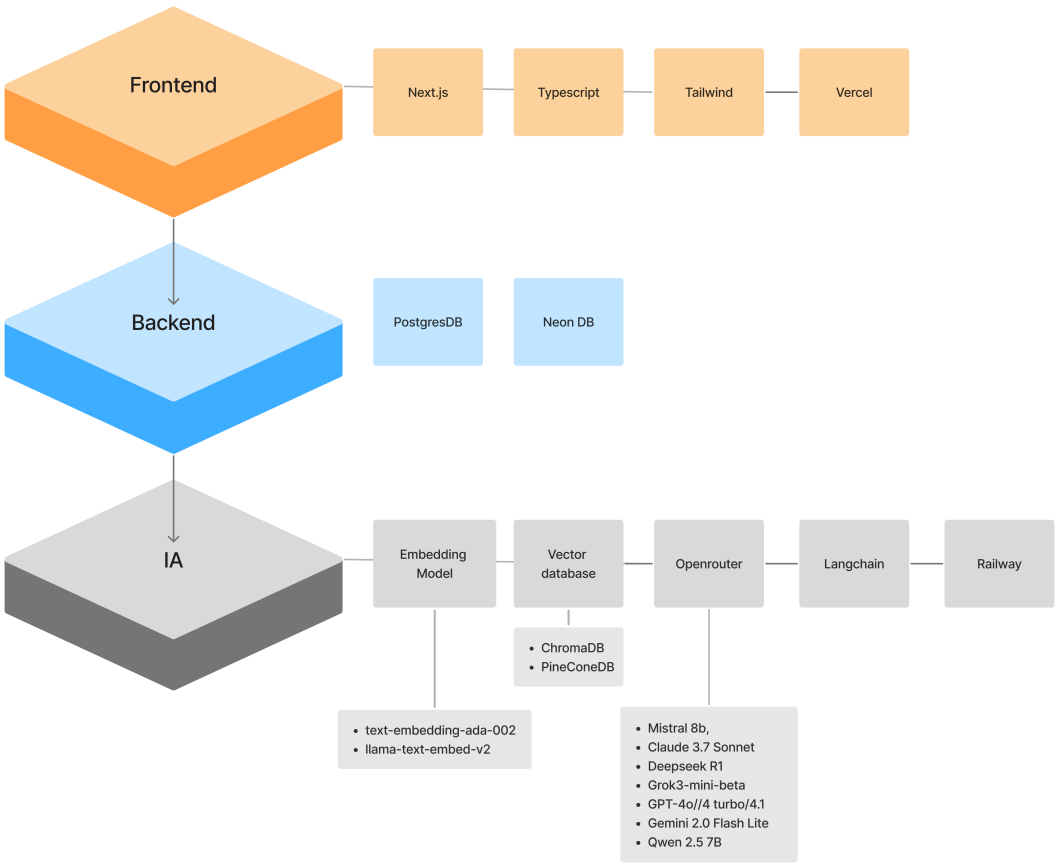

Figure 5: Schéma récapitulatif de la stack technique CYIA

4 Fonctionnalités développées

L'application repose sur deux piliers fondamentaux : l'intelligence artificielle et une interface web moderne assurant l'accessibilité, la personnalisation et la persistance des interactions.

4.1 Application web

Côté client, une interface responsive assure un rendu optimal sur ordinateurs comme sur smartphones. Le module d'authentification intègre : inscription classique (courriel + mot de passe), connexion, réinitialisation sécurisée, ainsi que l'accès direct via Gmail ou GitHub. Chaque profil conserve son historique de conversations, permettant de reprendre un échange en contexte.

Une page Contact offre un formulaire simplifié ; les messages sont transmis à l'équipe projet avec les métadonnées nécessaires au diagnostic (navigateur, identifiant de session, etc.).

Enfin, la landing page présente clairement les objectifs et les principales fonctionnalités de l'application, constituant une porte d'entrée privilégiée pour les nouveaux utilisateurs.

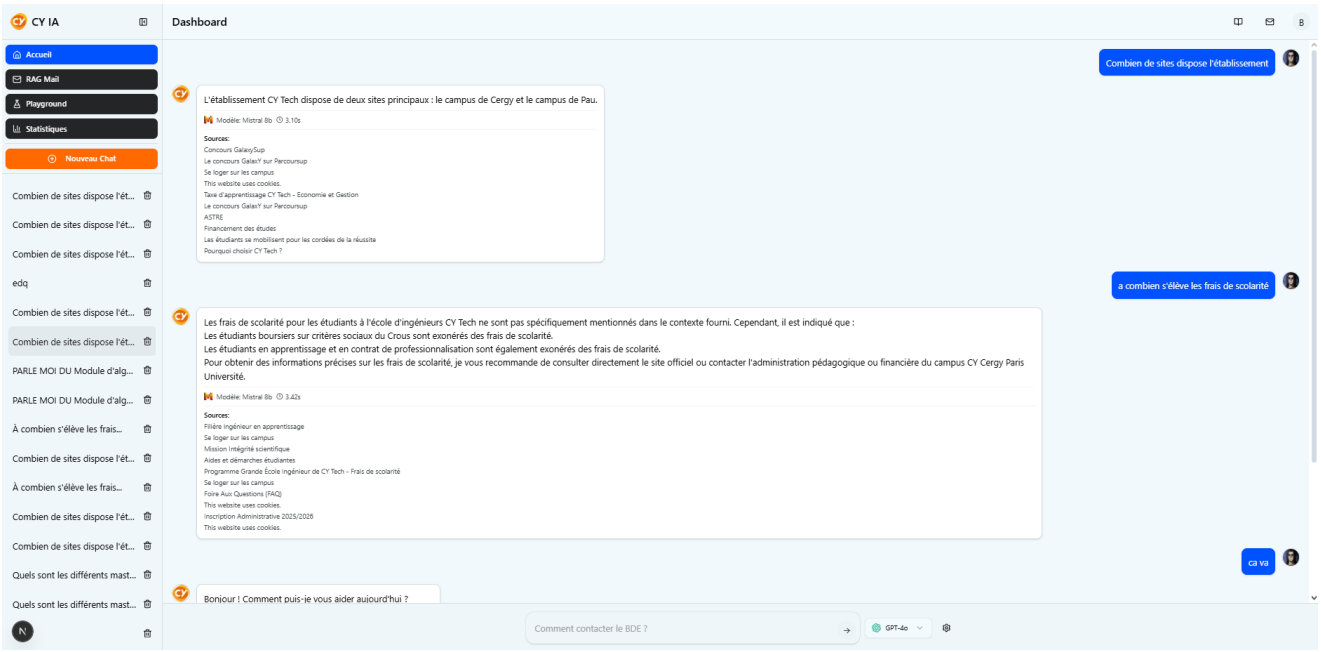

Figure 6: Screenshots de l'interface utilisateur (page dashboard)

4.2 Intelligence artificielle

Côté serveur se cache tout le coeur du projet : le mécanisme de RAG. Tout d'abord, il y a les données du site internet CY Tech. L'intégralité du site a été aspirée, segmentée puis indexée. L'utilisateur interroge ainsi toutes les pages institutionnelles en langage naturel, sans se soucier de leur emplacement exact.

Pour accroître la pertinence, la recherche s'appuie sur un enchaînement multi-query (diversification automatique des requêtes) suivi d'un reranking qui réordonne les segments selon des critères linguistiques et contextuels plus fins. Le moteur est également multimodal : grâce à l'API OpenRouter, il est possible de sélectionner à la volée le modèle le mieux adapté (GPT-4o, Claude 3, Mistral-Large, Grok, Gemini, etc.).

En outre, l'interface expose un paramétrage avancé des hyper-paramètres (température, top-p, top-k, pénalités de répétition, graine aléatoire). Un playground distinct permet d'expérimenter ces réglages sans qu'aucune requête ne soit enregistrée en base, favorisant les tests libres.

Enfin, les données sont stockées en base de donnée. Une page Analytics exploite cette base en fournissant un tableau de bord détaillé : nombre de requêtes, modèles les plus utilisés, distribution des paramètres et estimation du coût des appels.

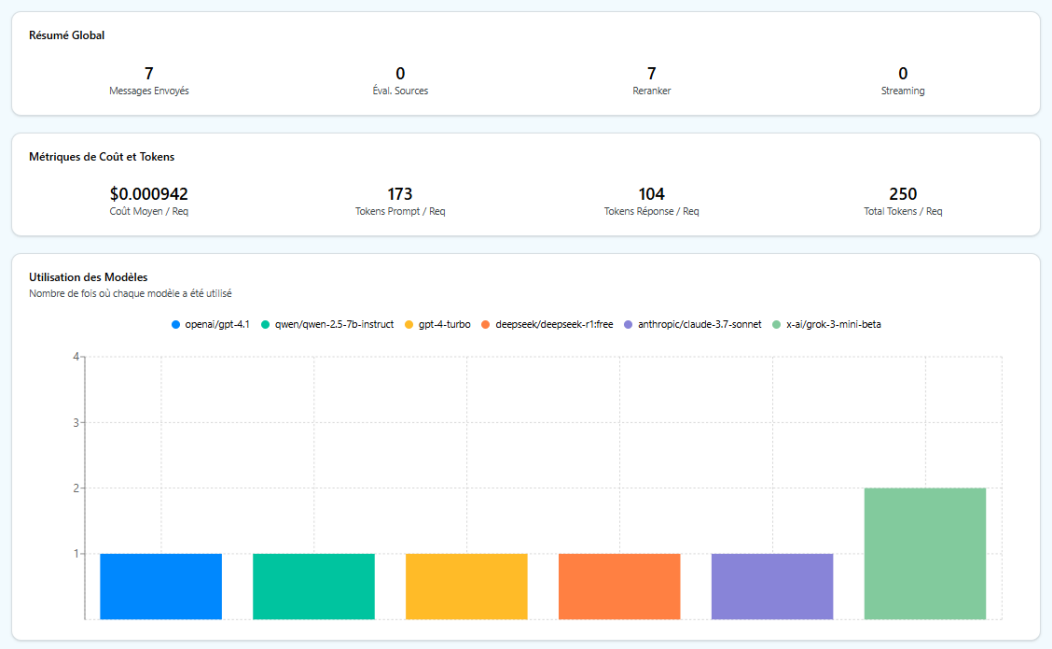

Figure 7: Screenshots de l'interface utilisateur (page statistique)

L'ensemble forme un écosystème cohérent : un backend robuste, une logique IA avancée et une interface accessible, le tout aligné sur l'objectif initial : offrir un accès immédiat, pertinent et centralisé aux informations de CY Tech, qu'elles proviennent du site web ou de la messagerie.

L'infrastructure a été conçue pour être entièrement déployée dans le cloud : l'interface web, la base de données relationnelle et les traitements d'intelligence artificielle sont chacun hébergés sur des environnements spécialisés, garantissant à la fois performance, scalabilité et sécurité.

Toutefois, l'architecture reste pleinement fonctionnelle en local, ce qui permet le développement, les tests ou l'usage autonome sans dépendance à une connexion externe.

5 La Pipeline de CY IA

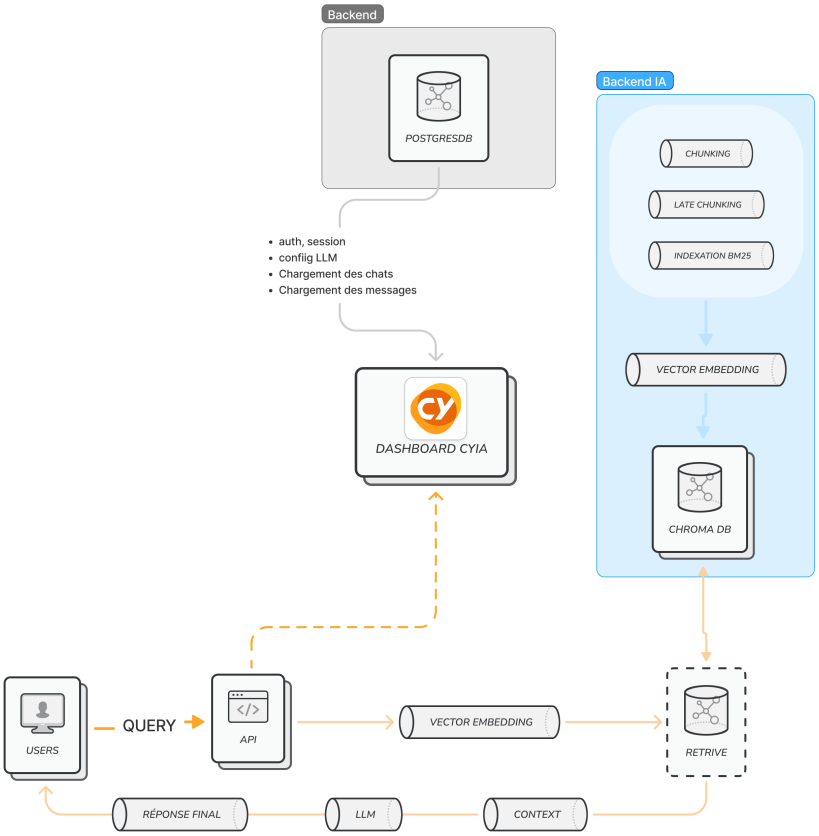

Figure 8: Schéma conceptuel de la pipeline CY IA

5.1 Base de donnée

La collecte des données publiques de CY Tech repose sur une chaîne entièrement automatisée, pensée pour couvrir l'ensemble des pages sans surcharger le serveur d'origine et sans dépasser les limites mémoire de la machine de traitement. L'objectif est d'obtenir, pour chaque URL, une version textuelle propre (markdown) et un *snapshot* de métadonnées utilisables lors des étapes de nettoyage et d'indexation.

5.1.1 Légalité du scraping et respect des règles d'accès

Le scraping de données publiques sur un site web est généralement légal dès lors que certaines conditions sont respectées. En particulier, il est essentiel de s'assurer que les pages collectées sont librement accessibles (sans authentification ni barrière payante) et que l'usage des données respecte les conditions d'utilisation du site. La norme `robots.txt` joue un rôle clé dans ce cadre. Ce fichier, situé à la racine du site (`cytech.cyu.fr/robots.txt`), permet aux administrateurs web de spécifier les parties du site que les robots sont autorisés ou non à explorer.

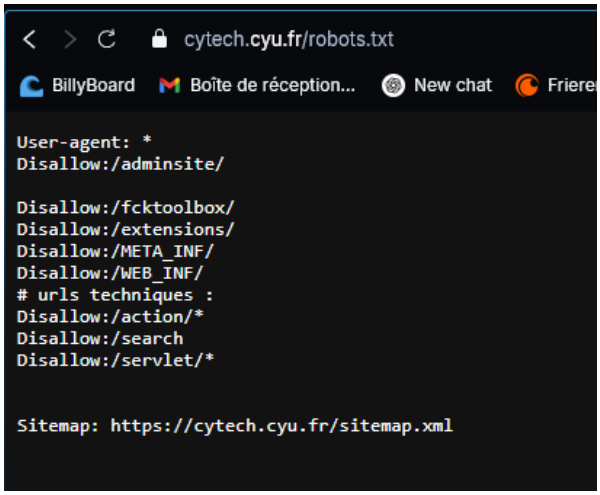

Figure 9: Screenshot de la page cytech.cyu.fr/robot.txt

5.1.2 Scraping du site internet CYTECH

Dans la présente pipeline, l'exploration repose exclusivement sur les URLs fournies par le fichier `sitemap.xml`, qui liste explicitement les pages destinées à être indexées. Ce choix garantit que seules les pages rendues publiques par l'administration du site sont traitées, et limite le risque de collecte non souhaitée.

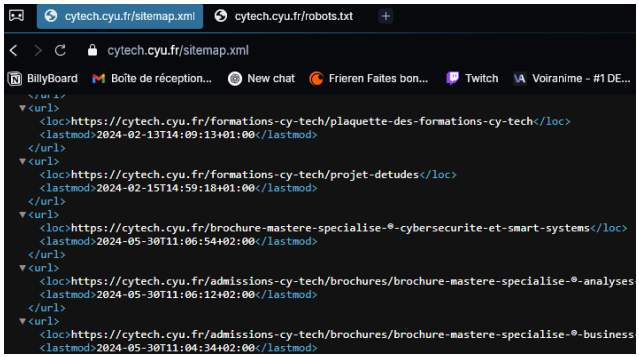

Figure 10: Screenshot de la page cytech.cyu.fr/sitemap.xml

La première étape est donc d'interroger ce fichier `sitemap.xml` du site cytech afin de récupérer la liste exhaustive des adresses publiées. Le sitemap est analysé comme un document XML ; toutes les balises sont extraites, fournissant un inventaire de plusieurs centaines d'URL à traiter. Ce mécanisme assure que la pipeline suit la structure officielle du site, tout en évitant les explorations arbitraires.

Les URL sont ensuite traitées par un *web crawler* asynchrone piloté par un navigateur sans interface graphique. En effet, le navigateur est instancié en mode headless, avec des paramètres de démarrage visant à réduire l'empreinte GPU et à interdire tout composant superflu (sandbox, cache, etc.). Ensuite, les pages sont rendues par lots (concurrence configurable, typiquement dix requêtes simultanées) où chaque lot réemploie la même instance du navigateur afin de minimiser le temps de lancement et la consommation CPU. Pour chaque page, le moteur de rendu fournit un extrait markdown normalisé (balises HTML nettoyées, tableaux convertis, images omises) qui servira de matière première pour le découpage en *chunks* de notre base de donnée vectorielle.

Le processus surveille en continu la mémoire vive du système. Un relevé est effectué avant et après chaque lot de pages. Cette télémétrie permet d'identifier les pics de consommation et de prévenir les débordements éventuels, garantissant la robustesse du scraping sur de longues sessions.

Pour chaque URL, deux fichiers sont produits :

1. un fichier markdown contenant le contenu textuel complet
2. un fichier JSON récapitulant les métadonnées essentielles (URL, horodatage du crawl, titre de la page, type de contenu, message d'erreur le cas échéant, etc.)

Les noms de fichiers sont dérivés de l'URL et sécurisés (caractères interdits remplacés) afin d'éviter les collisions et de rester compatibles avec tous les systèmes d'exploitation.

À la fin de l'exécution, la pipeline fait le bilan du nombre de pages *crawlé*es avec succès et du nombre d'échecs, puis ferme proprement le navigateur. Ce rapport constitue une trace d'audit car il permet de rejouer sélectivement la collecte sur les URL en échec ou d'anticiper les besoins d'actualisation lorsque de nouvelles pages apparaissent. Cette phase de scraping fournit ainsi un corpus textuel complet, structuré et daté, prêt à être transformé en *embeddings* pour l'indexation vectorielle dans Chroma/Pinecone DB.

5.2 Prétraitement du corpus

Le script de prétraitement parcourt ces fichiers et les répartit dans deux dossiers : un pour les fichiers qui dépassent deux cents caractères, et un autre lorsqu'ils sont plus concis. Cette séparation simplifie la suite du traitement : les textes longs seront découpés en segments, tandis que les courts pourront, au besoin, être regroupés.

Le nettoyage consiste à éliminer tout ce qui relève de la navigation ou de la mise en page : bandeaux de cookies, menus « Aller au contenu », boutons de partage, pieds de page redondants, ainsi que les images et leurs légendes. Après cette épuration, le programme identifie le titre principal, à défaut en construit un à partir du nom de fichier. Il extrait ensuite le corps du texte, c'est-à-dire la portion située entre ce titre et la première occurrence d'un élément non pertinent comme une image. Si ce bloc s'avère vide, toute la partie suivant le titre est conservée. Les phrases dupliquées sont supprimées et les encarts « En savoir plus » sont retirés afin d'obtenir un contenu linéaire et clair. Les métadonnées issues du crawl (URL, date, type de page, etc.) sont fusionnées avec le résultat du nettoyage. Le script ajoute le nombre de caractères restants et un indicateur signalant si le texte est court ou long. L'ensemble est enregistré : un fichier markdown révisé pour le contenu et un fichier JSON pour les métadonnées enrichies.

Tout au long du processus, chaque opération est consignée dans un fichier de journal. À la fin, un bilan indique le nombre total de pages traitées, la proportion de textes longs et courts, le taux de réussite et les éventuelles erreurs rencontrées. Le corpus ainsi obtenu est prêt pour le découpage en segments et la conversion en embeddings, étape préalable à l'indexation dans Chroma DB.

5.3 Création de la base de données vectorielle

Tous les fichiers classés comme « longs » sont chargés. Chaque document est ensuite découpé par un séparateur récursif Langchain `RecursiveCharacterTextSplitter` en blocs de 1000 caractères, avec un chevauchement fixe de 200 caractères. Cette segmentation, arrêtée dès la construction de l'index, permet de conserver la continuité du contexte tout en maintenant des unités textuelles adaptées aux requêtes RAG. Au moment du découpage, chaque segment reçoit un petit dictionnaire de métadonnées : le chemin complet du fichier markdown d'origine, son nom, ainsi que le titre et l'URL si ces éléments figurent dans le fichier JSON produit lors du prétraitement. Ces informations seront utiles pour afficher la source exacte et créer des liens vers la page correspondante du site CY Tech.

Une fois le texte segmenté, chaque bloc est transformé en

vecteur dense à l'aide d'un modèle `text-embedding-ada-002` d'OpenAI ou llama-text-embed-v2 de Llama. Le même modèle sert aussi bien à l'ingestion qu'à l'interrogation, de sorte que la métrique cosinus appliquée plus tard reste strictement cohérente entre l'indexation et la recherche. Les couples formés par les chunks et leurs métadonnées sont finalement stockés dans une unique collection Chroma.

5.4 Interaction avec les LLM (Query)

L'utilisateur commence par soumettre un prompt tout en choisissant les valeurs des hyperparamètres qu'il souhaite affecté à la pipeline. Celui-ci est immédiatement converti en vecteur avec le même modèle d'embedding utilisé lors de l'indexation, puis envoyé à Chroma/Pinecone pour interrogation. En réponse, Chroma ou Pinecone renvoie les segments les plus proches tirés de la collection vectorielle. Ces extraits constituent un premier contexte brut, prêt à alimenter la génération. Enfin, si l'option reranking est activé, un tri supplémentaire réordonne ces segments. Lorsqu'il est activé, un autre modèle LLM évalue la pertinence relative de chaque passage et place en tête ceux qui répondent le mieux à la requête. Seuls ces extraits priorisés sont transmis à l'étape suivante.

L'interface offre également un panneau de contrôle où l'utilisateur peut, avant chaque requête, ajuster finement les paramètres du modèle de langage et de la pipeline. Voici les réglages disponibles :

5.5 Génération de la réponse et gestion de l'historique

Le système assemble alors le prompt final : un rôle d'assistant (system prompt), la concaténation des extraits et de leurs sources (context prompt), la question de l'utilisateur et, le cas échéant, des instructions expertes ajoutées par l'utilisateur.

Paramètre	**Rôle**	**Plage ou effet typique**
`model`	Choix du LLM interrogé	Mistral 8b, Claude 3.7 Sonnet, Deepseek R1, Grok3-mini-beta, GPT-4o//4 turbo/4.1, Gemini 2.0 Flash Lite, Qwen 2.5 7B
`temperature`	Contrôle la créativité	0 = très déterministe ; 2 = très inventif (défaut 0.7)
`top_p`	Nucleus sampling (probabilité cumulée)	0.2 = réponse concentrée ; 0.9 = réponse diversifiée
`top_k`	Nombre de tokens candidats pris en compte	1 = maximum de prédictibilité ; >10 = plus de variété
`frequency_penalty`	Pénalise les tokens déjà fréquents	> 0 réduit la récurrence des mêmes mots
`presence_penalty`	Décourage la répétition d'idées	> 0 pousse le modèle à introduire de nouveaux concepts
`repetition_penalty`	Limite les séquences répétées	Valeur > 1 évite les boucles verbales
`seed`	Initialise l'aléatoire	Entier ; fixe la reproductibilité d'une réponse
`max_tokens`	Bornes de longueur de sortie	Coupe la réponse au-delà du seuil défini
`use_reranker`	Classement secondaire des passages	Booléen ; améliore la pertinence finale
`use_multi_query`	Génération de requêtes parallèles	Booléen ; augmente le rappel des informations
`evaluate_sources`	Attribution d'un score de fiabilité	Booléen ; affiche la confiance pour chaque extrait cité

Table 1: Hyperparamètres RAG ajustables par l'utilisateur sur la plateforme CY IA

Le LLM sélectionné produit la réponse, laquelle est immédiatement renvoyée à l'interface utilisateur. Pour finir, l'ensemble de l'échange (question, contexte utilisé, réponse générée, paramètres du modèle) est enregistré dans une base de donnée. Cette persistance alimente les tableaux de bord analytiques et assure une traçabilité complète.

6 Etude de cas

Cette section reprend les quatre questions initiales posées avec les paramètres par défaut, puis montre comment la réponse évolue quand on modifie un seul réglage à la fois : changement de modèle, variation de `top_k`, activation du multi-query ou du reranking. Les blocs de dialogue sont présentés sous forme de citations; juste après, un commentaire explique la différence observée.

6.1 Questions de Base

Question	**Réponse**
Quels sont les frais de scolarité ?	Les frais de scolarité à CY Tech sont indicatifs et susceptibles de modification, sans montant spécifique dans mes sources. Exemptions : boursiers CROUS (notification de bourse requise) ; étudiants en apprentissage ou contrat de professionnalisation (contrat requis). Consultez https://www.messervices.etudiant.gouv.fr/envole/ pour les bourses et https://cytech.cyu.fr/. **Modèle** : grok-3-mini-beta

Question	**Réponse**
Quelles sont les spécialités enseignées à CY Tech ?	CY Tech propose : Informatique (technologies et programmation), Mathématiques appliquées (contextes pratiques), Mathématiques Économie Finance (approche analytique), Biotechnologies & Chimie (sciences de la vie), Génie Civil (ingénierie des infrastructures). Mastères Spécialisés® pour expertise approfondie. Visitez https://cytech.cyu.fr/. **Modèle** : GPT-4o

Question	**Réponse**
Période à l'étranger possible à CY Tech ?	Oui, CY Tech propose des programmes d'échange et de mobilité internationale pour études ou stages à l'étranger, enrichissant l'expérience académique et interculturelle. Contactez le service des relations internationales ou visitez https://cytech.cyu.fr/. **Modèle** : Mistral 8b

Question	**Réponse**
Filières informatiques de CY Tech ?	CY Tech propose : Génie des Systèmes d'Information avec spécialisations en Intelligence Artificielle, Ingénierie du Cloud Computing, Cybersécurité, Informatique Embarquée, Visual Computing, Data Science, BI & Analytics, Systèmes d'Information d'Entreprise. **Modèle** : Mistral 8b

L'IA répond efficacement aux questions simples et bien définies, comme les spécialités ou les programmes d'échange, lorsque les informations sont disponibles sur le site de CY Tech. Cependant, elle rencontre des difficultés avec des questions vagues ou peu fréquentes, comme les frais de scolarité précis ou le nombre exact de campus, car ces données sont souvent absentes ou incomplètes dans les sources accessibles.

6.2 Importance du Choix de Modèle

Question	**Réponse**
Combien de sites dispose l'établissement ?	Je ne sais pas. Les informations fournies ne précisent pas le nombre de sites ou de campus de l'établissement CY Tech. **Modèle** : GPT-4o

Question	**Réponse**
Combien de sites dispose l'établissement ?	CY Tech dispose de plusieurs sites, notamment à Cergy-Pontoise et à Pau. **Modèle** : Mistral 8B (Ministral)

Le choix du modèle a un impact significatif sur la qualité de la réponse. GPT-4o, bien que performant, ne parvient pas à fournir des informations précises, se limitant à une réponse vague. Mistral 8B (Ministral) offre une réponse plus détaillée en identifiant deux sites (Cergy-Pontoise et Pau), mais sans garantir l'exactitude complète des informations. Cela met en évidence l'importance de sélectionner un modèle capable de combiner extraction d'informations pertinentes et fiabilité.

6.3 Influence du Nombre de Documents Récupérés

Question	**Réponse**
Combien de sites possède l'école ? (k=10)	CY Tech est implantée sur plusieurs campus, mais les détails spécifiques ne sont pas mentionnés dans les sources. **Modèle** : GPT-4 Turbo

Question	**Réponse**
Combien de sites possède l'école ? (k=20)	CY Tech dispose de campus à Cergy, Pontoise, et Saint-Germain-en-Laye. **Modèle** : GPT-4 Turbo

Augmenter `top_k` de 10 à 20 permet de récupérer des informations plus complètes, passant d'une réponse vague à une liste précise de trois campus (Cergy, Pontoise, Saint-Germain-en-Laye). Cela montre que l'augmentation du nombre de documents améliore le rappel et la qualité de la réponse.

6.4 Apport des Techniques RAG Avancées

Question	**Réponse**
Combien de sites possède l'école ? (Avant multi-query)	Aucune information sur le nombre de sites de CY Tech n'est disponible dans mes documents. **Modèle** : GPT-4 Turbo

Question	**Réponse**
Combien de sites possède l'école ? (Après multi-query)	CY Tech possède deux sites : Cergy-Pontoise (contact : `vie.campus.cytech@cyu.fr`, tél. 01...) et Pau (contact : `maeva.dikorebelle@cy-tech.fr`, tél. 05...). **Modèle** : GPT-4 Turbo

Question	**Réponse**
Frais de scolarité ? (Sans reranking)	Les frais de scolarité pour CY Tech ne sont pas détaillés dans les sources. Consultez `https://cytech.cyu.fr/` ou le service des admissions. **Modèle** : Mistral 8b

Question	**Réponse**
Frais de scolarité ? (Avec reranking)	Aucun montant précis des frais de scolarité. Indicatifs et modifiables, exonérations pour boursiers CROUS (notification requise) et étudiants en apprentissage/contrat pro (contrat requis). CVEC obligatoire, montant non précisé. Consultez `https://cytech.cyu.fr/`. **Modèle** : Claude 3.7 Sonnet

L'activation du multi-query permet de détecter des informations administratives (comme les contacts des campus de Cergy-Pontoise et Pau) en reformulant la requête pour explorer des pages non identifiées initialement. Le reranking améliore la pertinence en priorisant les extraits utiles, comme les exonérations et la CVEC, au lieu de réponses génériques. Ces techniques augmentent la couverture et la précision des réponses.

7 Limites et Perspectives

7.1 Performance

L'IA rencontre des difficultés à répondre aux questions lorsque l'information n'est pas répétée à plusieurs endroits sur le site de CY Tech. Par exemple, une question sur le classement de l'école selon le journal L'Étudiant reste souvent sans réponse, car cette donnée, bien que potentiellement présente, n'est pas suffisamment redondante ou explicite dans les sources accessibles. Malgré les expériences réalisées avec divers modèles et techniques RAG avancées (multi-query, reranking, variation de `top_k`), certaines informations, comme des détails administratifs précis ou des données spécifiques, échappent encore à l'extraction, même si elles figurent sur le site.
Pour améliorer le RAG, plusieurs pistes peuvent être envisagées :

- **Techniques de RAG avancées** : Développer des approches comme l'optimisation des embeddings ou l'utilisation de requêtes contextuelles pour mieux capturer des informations implicites ou peu répétées, en s'appuyant sur des modèles spécifiques au domaine éducatif.
- **Préprocessing plus poussé** : Améliorer le traitement des données brutes via un nettoyage approfondi, une segmentation intelligente des pages web et une extraction structurée des informations (par exemple, tableaux ou listes). Dans les architectures RAG, une grande partie des gains de performance provient d'un preprocessing efficace, qui rend les données plus accessibles et cohérentes pour le modèle.
- **Fusion multi-source** : Intégrer des sources externes fiables (comme des bases de données académiques ou des classements officiels) pour compléter les informations absentes ou rares sur le site principal.
- **Amélioration du reranking** : Optimiser les modèles de reranking pour prioriser les pages officielles ou les sections FAQ, souvent plus riches en données administratives.
- **Redirection vers un modèle alternatif** : Si un modèle ne parvient pas à répondre à une question, implémenter un système qui redirige automatiquement la requête vers un autre modèle mieux adapté, capable d'extraire ou de synthétiser l'information plus efficacement.

7.2 Gestion des Hallucinations

Le système RAG se révèle performant, car il filtre efficacement la plupart des requêtes lorsque l'information précise n'est pas présente dans le document source. Ce filtre a été réalisé à l'aide d'un simple prompt engineering. Toutefois, il peut survenir des cas d'hallucination partielle, où le modèle génère des informations plausibles mais incorrectes, comme l'inclusion de sites non confirmés. Pour réduire ces hallucinations, des mécanismes de validation croisée avec des sources externes ou des bases de données vérifiées pourraient être intégrés, ainsi qu'une pondération plus stricte des extraits récupérés pour privilégier les informations officielles.

7.3 Intégration de Données Multimodales

Une perspective d'amélioration serait d'incorporer des données non textuelles, comme des tableaux, graphiques ou PDF disponibles sur le site de CY Tech. Par exemple, les frais de scolarité ou les classements pourraient être extraits de documents officiels au format PDF, souvent ignorés par les modèles actuels. Cela nécessiterait des techniques d'extraction avancées (OCR, analyse de tableaux) pour enrichir le corpus de données.

7.4 Personnalisation des Réponses

Adapter les réponses au profil de l'utilisateur (étudiant, parent, chercheur) pourrait améliorer l'expérience. Par exemple, pour une question sur les frais, un étudiant boursier pourrait recevoir une réponse axée sur les exonérations, tandis qu'un étudiant international pourrait être dirigé vers des informations sur les frais spécifiques. Cela impliquerait d'intégrer un module de contextualisation basé sur des métadonnées utilisateur.

Ces limites et perspectives mettent en lumière le potentiel du RAG tout en identifiant des axes concrets pour renforcer sa robustesse et son adaptabilité aux besoins des utilisateurs.

8 RAG MAIL

8.1 Contexte et Objectifs

Contexte Les emails, bien qu'essentiels, forment un ensemble de données volumineux et complexe, rendant leur exploration manuelle inefficace. Les progrès en traitement du langage naturel, notamment avec des modèles comme CamemBERT pour la classification et des architectures RAG pour la recherche sémantique, permettent de transformer une boîte mail en une base de données interrogeable. Ce projet exploite ces technologies pour offrir une solution où les utilisateurs peuvent poser des questions simples et naturelles (par exemple, sur le contenu, les expéditeurs ou les catégories des emails) et obtenir des réponses rapides via une interface intuitive. En s'appuyant sur Google Cloud Platform (GCP) et PGAdmin pour gérer une base de plus de 4000 emails, le système intègre collecte, classification, et interrogation dans un cadre cohérent.
Objectifs Le projet cherche à concevoir un système capable de classer automatiquement les emails en neuf catégories (cours, réclamation, entreprise, spam, sondage, association, administratif, plateforme, autres), d'extraire des informations clés telles que les sujets, expéditeurs ou dates, de permettre une interrogation intuitive de la boîte mail comme une base de données via une interface conviviale, d'identifier les messages prioritaires comme les emails administratifs ou les réclamations. L'architecture repose sur un modèle CamemBERT pour la classification et un système RAG pour répondre aux requêtes des utilisateurs.

8.2 Méthodologie

Architecture Technique Le système CYIA repose sur une architecture modulaire intégrant plusieurs composants. Un script Python utilise l'API Gmail pour collecter les emails, qui sont ensuite classés en temps réel par un modèle CamemBERT entraîné sur des emails initialement annotés par GPT. Les emails sont stockés dans une base PostgreSQL gérée via PGAdmin, avec leurs métadonnées (identifiant, expéditeur, sujet, corps, date, catégorie). Une architecture RAG, basée sur LangChain, Chroma, et GPT-3.5-turbo, permet d'interroger les emails et de générer des réponses. Une interface Streamlit offre un accès interactif pour poser des questions.
Outils et Technologies Le développement s'appuie sur Python 3.10 pour le backend. Les services GCP incluent l'API Gmail pour la collecte et Cloud SQL pour la base PostgreSQL. Les outils NLP comprennent CamemBERT pour la classification, GPT-3.5-turbo pour la génération dans le RAG, et le modèle `sentence-transformers/all-mpnet-base-v2` pour les embeddings. LangChain et Chroma gèrent l'indexation et la recherche sémantique. L'interface utilisateur est développée avec Streamlit, et l'entraînement de CamemBERT a été effectué sur Google Colab avec un GPU T4.

8.3 Étapes de Développement

8.3.1 Collecte et Stockage

Un script (`update.py`) exploite l'API Gmail avec authentification OAuth 2.0, stockant les jetons dans `token.json`. Plus de 4000 emails sont collectés, avec leurs métadonnées extraites (expéditeur, sujet, corps, date). Ces données sont insérées dans une table PostgreSQL (`emails`) via `postgres.py`.

8.3.2 Classification Initiale avec GPT

Un script (`classification.py`) extrait 1000 emails non catégorisés et les classe via GPT-3.5-turbo dans neuf catégories : cours, réclamation, entreprise, spam, sondage, association, administratif, plateforme, autres. Un prompt structuré définit chaque catégorie. Ces 1000 emails catégorisés sont ensuite ajouté à la base de données PostreSQL et un fichier csv est créée contenant ces 1000 emails.

8.3.3 Classification avec CamemBERT

Le modèle CamemBERT, basé sur `CamembertForSequenceClassification`, a été entraîné sur le jeu de 1000 emails annotés. L'entraînement, utilise `CamembertTokenizer` pour tokeniser les textes (longueur maximale 256, troncature et padding). Les données sont converties en `Dataset` Hugging Face, avec un split 80/20 (800 emails pour l'entraînement, 200 pour l'évaluation). Les paramètres incluent 10 époques, une taille de batch de 8, une stratégie d'évaluation par époque, et une sauvegarde du meilleur modèle basée sur la précision (`load_best_model_at_end=True`). Le modèle, le tokenizer, et l'encodeur de labels (`label_encoder.pkl`) sont enregistrés dans `cyia_camembert_model`.
Par la suite les nouveaux emails reçus sont ajoutés à la base de données et directement catégorisés grâce au modele Camembert.

8.3.4 Système RAG

Le système RAG (`app.py`) permet d'interroger les emails et de générer des réponses. Les étapes incluent :

- **Chargement** : Extraction des 4000 emails depuis PostgreSQL, convertis en objets `Document` avec métadonnées (sujet, expéditeur, date, catégorie).
- **Indexation** : Découpage des emails en chunks avec `RecursiveCharacterTextSplitter`. Les embeddings, générés par `sentence-transformers/all-mpnet-base-v2` (dimension 768), sont stockés dans Chroma (`./chroma_db`).

- **Retrieval** : Un retriever basé sur Maximum Marginal Relevance (MMR), avec `k=8` documents retournés et `lambda_mult=0.7` pour équilibrer pertinence et diversité. MMR réduit la redondance en pénalisant les documents trop similaires, optimisant la couverture sémantique.
- **Génération** : La chaîne `RetrievalQA` exploite GPT-3.5-turbo pour synthétiser les réponses à partir des documents récupérés. Le prompt combine la requête utilisateur et les chunks pertinents.

L'interface Streamlit permet de poser des questions via un champ texte, affiche l'historique des interactions.

8.4 Résultats

Époque	Loss Entraînement	Loss Validation	Précision (Accuracy)
1	1.255600	0.987017	0.780
2	0.577900	0.636565	0.840
3	0.337300	0.628669	0.830
4	0.356400	0.652526	0.845
5	0.275800	0.744565	0.830
6	0.167300	0.792673	0.820
7	0.046900	0.777644	0.845
8	0.071000	0.743780	0.855
9	0.081700	0.725231	0.855
10	0.034000	0.743364	0.855

Table 2: Résultats de l'entraînement de CamemBERT

Le système CYIA a atteint des performances significatives pour ses fonctionnalités principales. La collecte a permis de récupérer et stocker plus de 4000 emails dans PostgreSQL, avec une gestion robuste des doublons et des champs manquants. La classification automatique via CamemBERT affiche une précision de 85,5% sur un ensemble de test de 200 emails. La perte d'entraînement a diminué de 1,256 (époque 1) à 0,034 (époque 10), tandis que la perte de validation a atteint un minimum de 0,725 à l'époque 9, indiquant une bonne convergence malgré un léger surajustement.
Le système RAG répond aux requêtes en 3 à 5 secondes en moyenne, avec des réponses pertinentes pour des questions spécifiques comme le contenu d'un email ou l'expéditeur d'un message donné. Le retriever MMR, avec 8 documents retournés et la `lambda_mult` de 0,7, améliore la diversité des résultats.

8.5 Discussion

8.6 Défis Techniques

La mise en œuvre du système a révélé plusieurs obstacles techniques. L'authentification via OAuth pour l'API Gmail a posé problème lors des tentatives de déploiement sur Google App Engine, notamment en raison d'une erreur liée à l'absence du fichier de jeton, probablement exclu par la configuration du projet.
Une autre difficulté majeure concernait l'automatisation de l'ajout des emails à la base de données. L'objectif initial était que les nouveaux messages soient insérés automatiquement, sans intervention manuelle. Cependant, cette automatisation complète n'a pas pu être réalisée. À la place, le système actuel nécessite le déclenchement manuel d'un script, qui reste actif en tâche continue (sous forme de boucle d'écoute) pour surveiller et insérer les nouveaux emails au fur et à mesure.
Enfin, la gestion d'une base de données contenant plus de 4000 emails a nécessité des optimisations pour garantir des performances acceptables lors des requêtes fréquentes. Le système RAG, bien que performant pour répondre à des questions précises, souffre encore d'une couverture limitée dans certains cas.

8.7 Solutions

Pour résoudre les problèmes d'authentification, une reconfiguration du fichier de déploiement a été envisagée, en s'assurant que le fichier de jeton soit inclus explicitement dans l'environnement de production.
Concernant l'ajout des emails, bien que l'automatisation complète n'ait pas pu être mise en place, une solution intermédiaire a été adoptée : un script, une fois lancé, fonctionne en boucle d'attente active pour détecter les nouveaux messages et les ajouter automatiquement à la base de données.
Des optimisations supplémentaires ont été mises en œuvre pour améliorer les performances de la base de données, notamment en affinant les index et en limitant la quantité de données récupérées par requête. Pour renforcer la couverture du système RAG, l'intégration de sources supplémentaires ou l'utilisation de modèles plus robustes est envisagée.

9 Conclusion

En conclusion, ce projet constitue une étape prometteuse vers la simplification et l'enrichissement de la vie étudiante. En s'appuyant sur les principes de la Retrieval-Augmented Generation, deux pipelines RAG exploitant respectivement les pages web de CY Tech et les e-mails institutionnels ont pu être implémenté. Ces architectures permettent déjà aux utilisateurs de poser des questions en langage naturel et d'obtenir des réponses contextualisées, tirées de sources spécifiques et pertinentes. Cette capacité à combiner la puissance des Large Language Models avec une recherche sémantique dynamique illustre le potentiel des approches hybrides pour surmonter les limites des LLM traditionnels, telles que les connaissances statiques ou les risques d'hallucinations.

Néanmoins, comme tout système en développement, ces pipelines ne sont pas encore parfaits. Des améliorations sont nécessaires pour garantir une fiabilité optimale, notamment à travers un pré-traitement plus rigoureux des données, un renforcement des techniques de RAG avancées ou encore l'ajout de sources externes pour enrichir le corpus de connaissances. Ces pistes d'optimisation, détaillées dans le rapport, constituent une feuille de route claire pour faire évoluer le système.

Malgré ces défis, les résultats actuels posent des fondations solides et valident l'orientation choisie. En poursuivant ces efforts, ce projet a le potentiel de transformer l'expérience étudiante à CY Tech, en offrant une interface intuitive, transparente et personnalisée, capable de répondre avec précision aux besoins des utilisateurs et ainsi optimiser l'accès à l'information au sein de l'établissement.

10 Liens Utiles

- Site CY Tech : https://cytech.cyu.fr
- Documentation Crawl4AI : https://docs.crawl4ai.com/advanced/multi-url-crawling/
- GitHub Crawl4AI : https://github.com/unclecode/crawl4ai
- Classement des modèles OpenRouter : https://openrouter.ai/rankings
- Guide Prompt Engineering (Lilian Weng) : https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/
- Guide du Prompting : https://www.promptingguide.ai
- Rerankers dans les systèmes RAG (Pinecone) : https://www.pinecone.io/learn/series/rag/rerankers/
- RAG multi-requête (Langchain + LlamaIndex) : https://teetracker.medium.com/...
- Documentation Next.js : https://nextjs.org/docs
- Documentation TypeScript : https://www.typescriptlang.org/docs/
- Documentation Tailwind CSS : https://tailwindcss.com/docs
- Documentation Neon DB : https://neon.tech/docs
- Documentation Pinecone DB : https://docs.pinecone.io/
- Documentation Langchain : https://docs.langchain.com/
- Documentation OpenRouter : https://docs.openrouter.ai/