CY TECH

CYIA : Système d'Analyse Automatisée des Emails avec RAG

Auteurs :
Yvan LOUAMBA
Bilal EL-Biyadi

Enseignante :
Souhila Arib

1 Introduction

L'essor des communications par email génère un volume de données textuelles difficile à gérer manuellement. Le projet CYIA (Classification Yvan Intelligence Artificielle) développe un système automatisé pour traiter les emails en combinant des techniques avancées de traitement du langage naturel (NLP). Ce système classe les emails en catégories prédéfinies, extrait des informations clés, et identifie les messages prioritaires. En s'appuyant sur l'API Gmail, une base PostgreSQL, un modèle CamemBERT, et une architecture RAG, le projet répond à des besoins pratiques tout en explorant des technologies modernes.

Ce rapport est structuré comme suit : la section 2 expose le contexte et les objectifs, la section 3 décrit l'architecture technique, la section 4 présente les résultats et métriques, la section 5 analyse les défis et solutions, et la section 6 conclut avec des perspectives.

2 Contexte et Objectifs

2.1 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.

2.2 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.

3 Méthodologie

3.1 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.

3.2 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.

3.3 Étapes de Développement

3.3.1 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`.

3.3.2 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.

3.3.3 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.

3.3.4 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 `RecursiveCharacterTextSplitte` 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.

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 1 – 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.

5 Discussion

5.1 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.

5.2 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.

6 Conclusion

Le projet CYIA a démontré la faisabilité d'un système automatisé pour l'analyse des emails, intégrant classification, extraction d'informations, génération de réponses, priorisation, et analyse visuelle. La pertinence des réponses du RAG valide l'efficacité de l'approche. Malgré des défis liés au déploiement et à l'échelle, les solutions mises en œuvre ont permis d'obtenir un prototype fonctionnel. Une extension future pourrait include l'indexation de l'ensemble des emails, une interface analytique plus avancée, et l'utilisation de modèles locaux pour réduire les coûts. Ce travail ouvre des perspectives pour des applications plus larges dans la gestion automatisée des communications.