Rapport d'analyse sur l'implémentation des programmes TF-IDF et Analyse de Sentiments

Ethan Van Ruyskensvelde

22 avril 2024

1 Introduction

Dans le cadre du cours de Langages de Programmation, il nous a été demandé de réaliser un projet sur la simulation d'une Intelligence Artificielle Générative. Pour ce faire, nous devions écrire deux programmes : TF-IDF et analyse de sentiments.

Ce rapport d'analyse présentera l'implémentation des programmes TF-IDF et analyse de sentiments en C++. Dans celui-ci seront expliqués les choix des structures de données utilisées, l'utilisation des boucles, des méthodes de passage de paramètres, et des différences que nous aurions pu rencontrer entre Python et C++ pour l'implémentation de ce projet.

2 Méthode

2.1 Choix des Types

Voici les différents types de données utilisés dans les deux programmes :

- `std::vector` pour stocker une liste de mots dans un document.
- `std::map` pour stocker les comptages de mots. Il a permis d'associer une valeur à un mot. Notamment pour stocker les valeurs TF, IDF et TF-IDF associé à un mot.
- `std::string` pour stocker et modifier un mot.
- `struct` pour définir la structure du dictionnaire de sentiments et y stocker les mots positifs et négatifs.

Dans le cas où je ne savais pas quel type de données choisir pour que ce soit le plus efficace, j'ai laissé le compilateur choisir en utilisant auto.

2.2 Utilisation des Boucles

J'ai utlisé dans mon programme différentes structures de boucles en fonction des besoins :

- Des boucles `for` pour parcourir les conteneurs comme les vecteurs et les maps, où pour parcourir tous les caractères d'un mot pour vérifier si c'était un signe de ponctuation ou un alphanumérique.
- Des boucles `while` pour lire tous les mots d'un fichier.

2.3 Passage de Paramètres

J'ai utilisé plusieurs méthodes pour passer des paramètres en C++, notamment :

- Le passage par valeur lorsque je ne voulais pas modifier les paramètres.
- Le passage par référence lorsque je devais modifier les paramètres.
- Le passage par const référence pour éviter la copie inutile de données.

3 Résultats

3.1 TF-IDF

Si je n'ai pas fait d'erreur, je pense avoir réussi à implémenter correctement le programme TF-IDF, qui permet de calculer les scores TF-IDF pour chaque mot dans un ensemble de documents. TF étant le nombre de fois qu'un mot apparaît dans un document et IDF étant le nombre de documents contenant un mot en particulier.

3.2 Analyse de Sentiments

Je pense aussi avoir réussi à implémenter correctement le programme analyse de sentiments qui fonctionne en déterminant si un texte est positif, négatif ou neutre en fonction du nombre de mots positifs et négatifs présents à la fois dans le document à traiter, et dans le dictionnaire qui permet de le déterminer. Le dictionnaire est lui-même rempli par des mots présents dans un texte source.

4 Analyse

4.1 Comparaison avec Python

Si le projet avait été réalisé en Python, nous aurions bénéficié d'une syntaxe plus lisible. Les boucles et les conditions sont plus intuitifs, plus simples à implémenter. Python propose aussi une gamme de structures de données intégrées plus importante qu'en C++, comme les listes qui sont intégrées en Python, alors qu'en C++ il faudrait importer le module vector par exemple. Cependant, en C++, nous avons plus de contrôle sur la gestion de la mémoire et des ressources. C++ sera globalement plus rapide en temps d'exécution que Python.

5 Conclusion

Pour conclure, les programmes TF-IDF et analyse de sentiments sont fonctionnels (si je n'ai pas fait d'erreur). J'ai essayé de les implémenter de manière la plus efficace que j'ai pu faire. Ce projet m'a permis d'apprendre encore d'autres choses utiles en C++, comme l'utilisation des vecteurs et maps que nous n'avions pas fait en TP. Ce projet m'a également permis de me familiariser avec le choix des types et le passage de paramètres.

6 Sources

W3Schools : https://www.w3schools.com/cpp/
freeCodeCamp : https://www.freecodecamp.org/news/c-plus-plus-map-explained-with-examples/