Distance de Jaccard

Maysaloon BILAL & Tristan GROULT

29 avril 2025

Sommaire

1 Introduction

Ce projet consiste à développer un programme en langage C permettant de calculer la distance de Jaccard entre plusieurs fichiers texte, afin d'analyser leur similarité lexicale. La distance de Jaccard est une valeur comprise entre 0 et 1 : plus elle est proche de 0, plus les fichiers sont différents, et plus elle est proche de 1, plus ils sont similaires.

La distance de Jaccard entre deux ensembles A et B est définie par la formule :

$$\text{Distance de Jaccard}(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Le programme permet plusieurs personnalisations. Il est possible d'afficher un graphe indiquant à quels fichiers chaque mot appartient, ou d'activer un mode détaillé qui précise pour chaque mot les fichiers dans lesquels il apparaît. Une autre option permet également de définir une longueur maximale pour les mots analysés.

Pour gérer efficacement les mots extraits des fichiers, nous avons utilisé une table de hachage. Sa gestion — ajout, suppression, affichage du graphe, etc. — est implémentée dans le module `jaccard`. Ce projet s'appuie également sur deux autres modules : le module `word`, dédié à la création et la manipulation des mots, et le module `opt`, responsable du traitement des options en ligne de commande.

L'ensemble de ces composants est intégré et testé dans le fichier principal `main.c`, qui constitue le point d'entrée du programme.

2 Méthodologie

Dans cette section, nous expliquons les différentes étapes et approches utilisées pour implémenter le programme. Nous détaillerons la conception des modules, la gestion des fichiers et l'utilisation de la table de hachage.

2.1 Module word

Le module `word.c` sert à construire dynamiquement un mot caractère par caractère. Il est composé de 3 champs :

- Un champ `s` de type `char *` qui pointe vers un tableau de caractères qui stocke le mot lui-même.
- Un champ `length` de type `size_t` qui représente la longueur du mot.
- Un champ `capacity` de type `size_t` qui représente la capacité du tableau pointé par `s`

```c
struct word {
    char *s;
    size_t length;
    size_t capacity;
};
```

FIGURE 1 – Structure word

— **Initialisation d'un mot** : La fonction `word_init()` permet de créer un nouveau mot avec une capacité initiale minimale définie par la constante `CAPACITY_MIN`. Elle alloue dynamiquement de la mémoire pour la chaîne de caractères et initialise sa longueur à zéro.
— **Ajout d'un caractère au mot** : La fonction `word_add()` ajoute un caractère à un mot. Si le mot atteint sa capacité maximale, la fonction double l'espace mémoire disponible avec `realloc()`, pour permettre l'ajout de nouveaux caractères. Cela permet au programme de gérer des mots de tailles variables sans perdre de données.
— **Réinitialisation du mot** : La fonction `word_reinit()` permet de réinitialiser un mot, c'est-à-dire de remettre sa longueur à zéro et de vider la chaîne. Cela est utile lorsqu'un mot doit être réutilisé sans allouer de mémoire supplémentaire.
— **Accès à la chaîne de caractères** : La fonction `word_get()` renvoie un pointeur vers la chaîne, tandis que `word_get_clean()` copie le mot dans dest, permettant ainsi de ne récupérer que le mot réel, sans les parties inutilisées de la mémoire allouée. Cela garantit que le mot est prêt à être utilisé sans risque de contenir des espaces "vides" dans la mémoire.

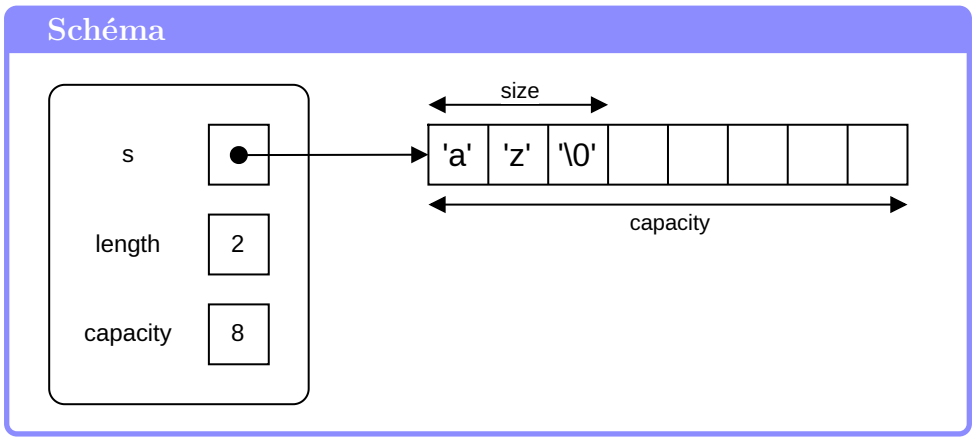

FIGURE 2 – Schéma de la structure word

2.2 Module opt

Le module `opt` prend en charge le traitement des options fournies en ligne de commande.