Distance de Jaccard

Maysaloon BILAL & Tristan GROULT

30 avril 2025

Sommaire

1 Introduction **3**

2 Méthodologie **3**

2.1 Module word 3

2.2 Module opt 4

2.2.1 Spécification 4

2.2.2 Implantation 5

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`

```
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.

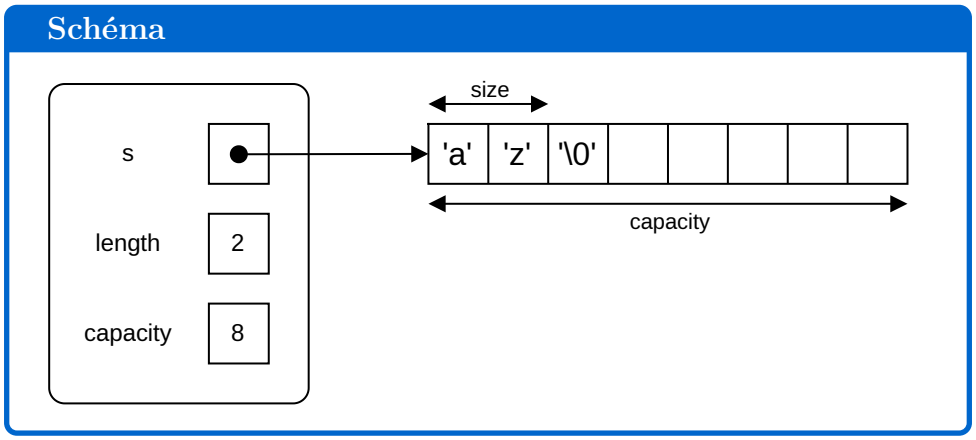

FIGURE 2 – Schéma de la structure word

2.2 Module opt

Le module `opt` a pour objectif de gérer facilement les options lue sur la ligne de commande afin de les annlyser et de le retranscrire dans une forme utilisable. Pour ce projet, nous avons beson des options suivante :
— Une option permettant la demande d'affichage du graphe de Jaccard.
— Une option permettant d'ignoré les caractères de ponctuations.
— Une option permettant de majoré la longueur des mots.
— Une option permettant d'utiliser l'entré standard comme source.
— Une option permettant de définir le prochain élément comme le nom d'un fichier.
— Une option permettant d'afficher comment utiliser l'exécutable.
— Une option permettant d'afficher la version de l'exécutable.
— Une option permettant d'afficher l'aide de l'exécutable.

2.2.1 Spécification

Pour cella, nous utilisont une structure de donné définie par le shéma de la figure 3. Nous avons choisit de limiter le nombre de fichier à une constantante nomé ici `max files`

afin de bénéficiait d'un temps et espace constant. Cepandant, ce choix est aussi utile pour les choix d'implémentations que nous nous sommes fixé pour le module `jaccard`.

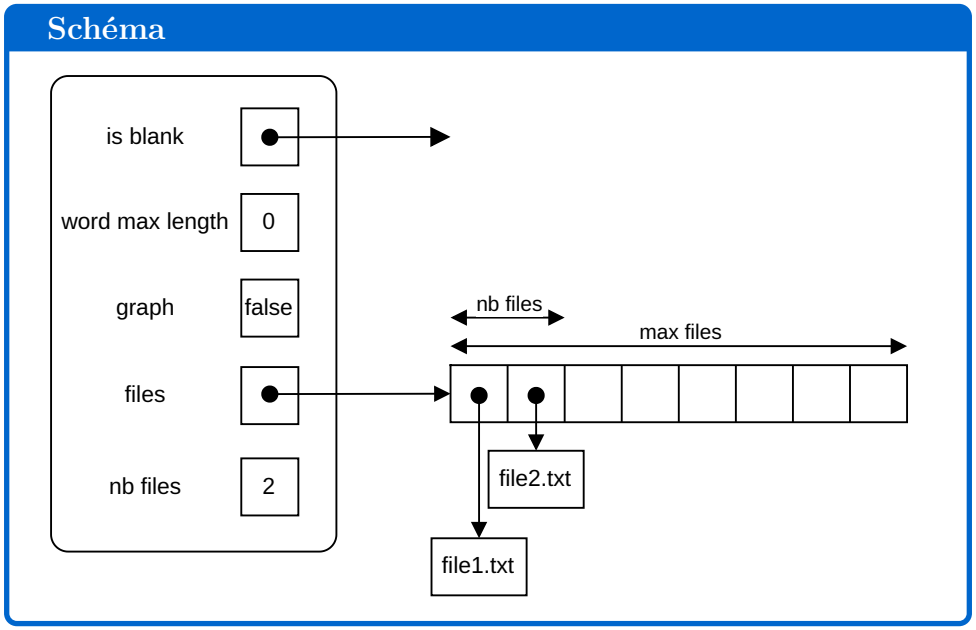

FIGURE 3 – Schéma de la structure word

2.2.2 Implantation

Lors de l'implémantation, nous avons choisit d'utiliser un `struct` (figure 4) pour représenter notre structure de donnés. Nous avons aussi choisi de sauvgarder l'utilisation de l'entré standard de la même manière que pour les fichiers en définisant grâce à une macro constante nomé `STDIN` la valeur spécifique `""` qui est la chaine vide car aucun systheme majeure authorsise cette valeur comme nom de fichier. Cette structure est composé de 5 champs :

- Un champ `isBlank` de type `int (*)(int)` qui définie la fonction déterminant les caractère séparant les mots.
- Un champ `word_max_length` de type `int` qui définie le nombre de caratères maximum composant un mot. Si cette valeur vaut 0, alors aucune limitations sur la taille du mot n'est appliqué, ça valeur par défaut est 0.
- Un champ `graph` de type `bool` qui définie si l'utilisateur demande l'affichage du graph de Jaccard ou non.
- Un champ `files` de type `const char **` qui définie les nom des fichiers à traiter.
- Un champ `nb_files` de type `int` qui définie le nombre de fichiers spécifiait.

Nous avons fait le choix de dévelopé uniqument les options courtes pour les options sont énéoncé si dessous. Les options sont noté respectivement :

- L'option est noté `-g`.
- L'option est noté `-p`.
- L'option est noté `-iVALUE`.
- L'option est noté `-`.
- L'option est noté `--`.
- L'option est noté `-u`.
- L'option est noté `-v`.

```c
struct opt {
    int (*isBlank)(int);
    int word_max_lenght;
    bool graph;
    char const **files;
    int nb_files;
};
```

FIGURE 4 – Structure opt

— L'option est noté `-?`

Nous pouvons remarquer que pour l'option `-iVALUE`, `VALUE` est collé au `-i`. Celle signifie que cette options fonctionne uniquement dans ce cas. Par exemple `-i3` est correct à l'inverse de `-i 3`.

Finalement, nous avons aussi fait le choix de rendre le codage de nos options facilemnt adaptatif en définisant toutes nos options grâce à des macro constante mais aussi en définisant séparément le préfix signifiant une option courte pour que n'importe qu'elle valeurs puisse être utilisé dans nos macros.