UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et génie informatique

RAPPORT APP5

Structure de données et complexité
GIF 270

Présenté à
Équipe de formateurs de la session S2

Présenté par
Raphael Bouchard – bour0703
Alexis Guérard – guea0902

Sherbrooke – 15 mars 2023

1. STRUCTURES DE DONNÉES UTILISÉES

Pour réaliser la problématique de cet APP, nous avons dû réaliser des structures de données soient des dictionnaires et des listes. Premièrement, nous avons utilisé des dictionnaires afin de stocker des mots utilisés dans les œuvres de différents auteurs. Pour ce faire nous avons créé un premier dictionnaire ayant comme clé le nom de l'auteur et comme valeur un autre dictionnaire de mots. Ce deuxième dictionnaire était composé de n-gramme comme clé et de la fréquence comme valeur. Puisque la clé était de n-gramme, celle-ci pouvait changer de forme selon ce qui était demandée en configuration (un mot, deux mots, trois mots, etc.). Ces structures de données ont été créé pour la fonction $analyze()$ et la fonction $find\_author()$. Ensuite, pour effectuer la fonction $get\_nth\_element()$ et $gen\_text()$ nous avons utilisé les dictionnaires déjà créé dans les fonctions plus haut. Les listes sont les deuxièmes structures de données utilisées. En effet, ceux-ci nous ont été utile pour les fonctions $analyse()$ et $find\_author()$. Elles ont permis de stocker les mots afin d'effectuer les n-grammes pour ensuite les entreposer dans le dictionnaire correspondant à son auteur. Pour la fonction $get\_nth\_element()$, le dictionnaire est inversé afin d'avoir la fréquence comme clé. L'utilisation d'une liste de n-gramme, étant la valeur dans ce dictionnaire, nous a permis de retourner le $n^{ième}$ élément de la liste du plus fréquent au moins fréquent ainsi que les cas où il y en a plusieurs. Pour effectuer la fonction $gen\_text()$, on utilise une liste de probabilité pour stocker la fréquence des n-grammes et une liste de n-gramme.

2. COMPLEXITÉ DES ALGORITHMES

L'algorithme des fonctions $analyse()$ et $find\_author()$ ont une complexité de $Big\ O(n)$. Puisque le nombre d'auteur et d'œuvre est faible, nous pouvons négliger ces boucles for. En effet, pour ces deux algorithmes, la complexité dépend seulement du nombre de mots se retrouvant dans les œuvres soient n. Cependant, s'il avait été question de faire ces algorithmes pour une immense bibliothèque de plusieurs auteurs et d'œuvres, le $Big\ O$ aurait été de n^3 puisqu'il y a trois boucles for.

3. MOTS LES PLUS UTILISÉS

Tableau 1: Mots les plus utilisés par chacun des auteurs

Auteurs	Mot le plus utilisé
Balzac	les
Hugo	les
Ségur	que
Verne	les
Voltaire	les
Zola	les

4. BI-GRAMMES LES PLUS UTILISÉS

Tableau 2 : Digramme les plus utilisés par chacun des auteurs

Auteurs	Digramme le plus utilisé
Balzac	dans les
Hugo	jean valjean
Ségur	des ormes
Verne	dans les
Voltaire	tous les
Zola	elle avait