DM - Correcteur Orthographique

Rendu 2

MELLOUK Imân – MENSAH ASSIAKOLEY Ako Seer Harley

03/04/2022

Table des matières

Objectif

Nous sommes désormais capable de trouver les mots mal orthographiées (relatif au dictionnaire de référence) d'un texte.

Dans ce deuxième rendu, l'objectif étant de proposer pour chacun de ces mots une possible correction.

DOCUMENTATION UTILISATEUR

Entrée / Sortie

- Entrée : le programme prend en argument un dictionnaire de référence ainsi que le texte à corriger. Ces fichiers se trouvant dans le répertoire **ressources.**

- Sortie : Sur le terminale tous les mots considérés comme mal orthographié sont affichés suivie d'une proposition de correction

Exécution du programme

Afin de lancer le programme, il faut se situer dans le répertoire Correcteur_Part2 puis écrire la commande . /make qui créera un exécutable correcteur_1.

Puis taper la commande suivante :

./correcteur_1 ressources/[nom du texte à corriger] ressources/[nom du dictionnaire de ref]

Par exemple, pour lancer le programme avec le texte à corriger a_corriger_1.txt avec pour référence le dictionnaire dico_2.dico. On écrit la commande suivante :

./correcteur_1 ressources/a_corriger_1.txt ressources/dico_2.dico

DOCUMENTATION TECHNIQUE

Dans un premier le temps, le programme va distinguer les mots non correctes des mots corrects, puis les stockera dans une liste chainée. C'est à partir de cette liste que nous allons proposer des corrections possibles.

Implémentation de l'algorithme de correction

Distance de Levenshtein

La distance de Levenshtein est une distance donnant une mesure de la différence entre deux chaines de caractères. Elle vaut le nombre minimal d'opération à effectuer pour passer d'une chaine de caractère à une autre. Les opérations possible sont :

- Suppressions d'un caractère
- Ajout d'un caractère
- Remplacement d'un caractère par un autre

Par exemple, la distance de Levenshtein obtenue entre le mot de niche et chien est de 4.

Pour calculer cette distance, il y a 3 issues possible :

- On connait la distance entre niche et chie, puis il suffit d'ajouter +1 pour l'opération d'ajout du caractère n. L(niche,chien) = L(niche,chie)+1 pour l'opération d'ajout.

- On connait la distance entre nich et chien, L(niche,chien) = L(nich,chine)+1 pour l'opération de suppression.

- On connait la distance entre nich et chie et donc L(nich,chie) = L(nich,chie)+1 pour l'opération de modification.

Finalement pour choisir la bonne opération, on choisit celle avec le cout minimale.

Il est probable que nous ne connaissions pas les distances de Levenshtein entre nich et chien préalablement. Pour calculer ces distances il faudrait également les redissocier en 3 sous-problème et ainsi de suite…

Ainsi, nous avons décidé de nous appuyer sur l'algorithme donné par Wikipédia (https://fr.wikipedia.org/wiki/Distance_de_Levenshtein).

Ce calcule s'appuie sur un tableau, la distance calculer finale se trouvera dans la dernière case du tableau.

Pour commencer, il suffit de remplir le tableau comme tel :

	…	C	H	I	E	N
…	0	1	2	3	4	5
N	1					
I	2					
C	3					
H	4					
E	5					

Les nombres obtenues sont ceux en comparant la chaine de caractère vide et le mot .

Pour la suite du remplissage, pour remplir une case à la position i,j il y a deux possibilité :

- Soit les deux lettres comparés sont les mêmes du coup la valeur de la case i, j sera équivalente à la valeur de la case i-1,j-1.
- Sinon, on compare le cout minimale entre la case(i,j-1), case(i-1,j) et la case(i-1,j-1). On prend ce nombre et on l'incrémente de 1.

On obtient le tableau suivant :

	…	C	H	I	E	N
…	0	1	2	3	4	5
N	1	1	2	3	4	5
I	2	2	2	2	3	4
C	3	2	3	3	3	4
H	4	3	2	3	4	4
E	5	4	3	3	3	4

Correction orthographique par force brute

Pour effectuer une correction, l'algorithme à suivre été tel :

On initialise une variable dmin à une très grande valeur INFINI (on l'a mis à 1000000).

Pour chaque mot de la liste chainé erreurs, on calcule la distance de Levenshtein entre ce mot et chaque mot présent dans l'arbre (récupéré dans une variable mot dont la longueur est initialement à 26). Si cette distance est inférieur ou égale au dmin : si elle est strictement inférieur(on a trouvé une distance plus petite que précédemment), dmin prend la valeur de la distance, on vide les mots qui pouvait potentiellement être une correction du mot car une distance plus petite à été trouvé puis on insère le nouveau mot trouvé. si elle est égale, on insère le mot dans la liste chainée qui correspond à la correction. Finalement on renvoie cette liste.

Nota Bene

Nous nous sommes rendu compte que lors d'insertion d'un dictionnaire trié lexicographiquement dans un arbre ATR, nous obtenions une segfault, ce qui n'est pas le cas lors de l'insertion d'un dictionnaire non trié. Nous avons corriger la fonctions d'insertion.