Nom : Saou

Prénom : Ayman

Matricule : 000593526

Rapport d'analyse

1) Choix des Types :

Pour stocker les entiers j'ai la plupart du temps utiliser le type *unsigned int* étant donné qu'on ne travaille pas avec des chiffres négatifs dans ce contexte, a l'exception de la taille du vecteur contenant dans les mots dans la fonction *computeTF* pour lequel j'ai utilisé *unsigned long int* car elle peut être amené à être supérieure à 65 535, dans la structure *AnalysisResult* j'ai aussi jugé bon de transformer le type *int* en *unsigned int* qui nous permet de stocker 2 fois plus de nombres étant donnés que *positiveCount* et *negativeCount* ne seront jamais < 0. Concernant les nombre décimaux, j'ai gardé *double* pour plus de précision.

2) Utilisation des boucles

J'utilise des boucles selon moi inévitables dans mes fonctions crées *inVector* et *countFrequency*, réalisant les mêmes opérations que respectivement *std::find* et *std::count* mais sans utiliser la librairie *algorithm*. Dans la fonction *computeIDF* j'utilise deux boucles imbriquées pour regrouper tous les mots de tout les documents dans un seul vecteur pour ensuite utiliser une seconde fois deux boucles imbriquées mais à partir de la liste de mots sur les documents, je trouve ça pas très élégant ni optimisé et ça aurait surement pu être évité mais je n'ai pas trouvé comment. Dans ma fonction *searchWordInDictionary* qui permet de savoir si un mot se trouve dans le vecteur positif ou négatif d'un dictionnaire, j'utilise une boucle pour chaque vecteur, avec un return dans chaque boucle pour ne pas boucler inutilement lorsqu'on a trouvé si le mot était positif ou négatif.

3) Passage de paramètre

J'utilise principalement des passages en paramètre par référence afin d'éviter d'utiliser trop d'espace inutilement dans le cas d'une copie volumineuse, j'utilise le mot clé *const* lorsque je veux m'assurer qu'un paramètre qui n'a pas vocation à être modifié ne soit modifié involontairement, exemple : dans ma fonction *addWordsToDictionnary* je passe en paramètre un *SentimenDictionary* par référence sans le mot clé *const* car la fonction a pour but de modifier les attributs *positiveWords* et *negativeWords* du dictionnaire. Cependant, dans la fonction *calculateTFIDF* j'aurais aimé que le paramètre *idf* ne soit pas constant, en effet pour calculer le TFIDF je copie la *map* contenant le TF, itère paire par paire sur cette copie et remplace la valeur actuelle (TF) de la clé (mot) par la valeur actuelle multipliée par l'IDF de la clé (mot)[1], cependant vu que la *map idf* est passé en paramètre constant, je suis obligé de faire une copie ce qui est couteux en temps et en espace mémoire.

4) Comparaison avec Python

[1] Exemple : map TFIDF (copie map TF) => {{« de », 2}, {« la », 3}}, map IDF => {{« de », 0,5}, {« la », 1}}
TFIDF[« de »] = IDF[« de »] * 2

N’étant pas encore completement familier avec le C++, j’ai l’impression de « réfléchir » comme en python et de simplement transposer ce que j’aurais écrit en python mais avec les outils a disposition en C++, cependant en C++ il faut être beaucoup plus rigoureux quant au choix des types, des paramètres et des structures de données, par exemple le C++ permet de ne pas utiliser d’espace mémoire inutilement avec la caractérisation spécifique des types comme le unsigned qui permet d’optimiser le nombre de bytes utilisé lorsque l’on sait qu’on utilisera pas de nombre négatif, le long double qui permet une plus grand précision quant aux nombres décimaux etc…