Détection de langue

On souhaite réaliser un programme permettant de déterminer la langue utilisée dans une phrase tapée par l'utilisateur. On utilisera pou cela le principe de l'apprentissage supervisé.

On procèdera en 3 étapes :

- On utilise un texte écrit en langue française (*Les Misérables*, Victor Hugo) et un texte écrit en langue anglaise (*Une Étude en Rouge*, Conan Doyle) pour extraire un ensemble de phrases étiquetées par leur langue (français ou anglais).
- On étudie les bigrammes présents dans ces phrases, ce qui permet d'associer à chaque phrase un vecteur de l'espace vectoriel $\mathcal{M}_{26}(\mathbb{R})$.
- On utilise l'algorithme des k plus proches voisins pour déterminer l'étiquette d'une phrase tapée par l'utilisateur.

1 Extraction du corpus

Le corpus utilisé est donné sous forme de deux fichiers textes `miserables.txt` et scarlet.txt qui ont été prétraités pour enlever certains caractères (accents, ponctuation, ...), mettre tout en minuscules et posséder exactement une phrase par ligne.

La première étape consiste à charger en mémoire chacun de ces deux fichiers, dans une matrice de caractères où les lignes correspondent aux phrases du texte.

```
//On se limite à 10 000 phrases et 10 000 caractères par phrase

// lire fichier, lit le fichier et enregistre les phrases dans la ma-
trice corpus, elle retourne le nombre de phrases lues

int lire_fichier(char* nom_fichier, char corpus[10000][10000]) {
    FILE* fichier;
    fichier = fopen(fichier, "r");
    if (fichier == NULL) {
        fprintf(stderr, "Impossible d'ouvrir le fichier %s\n", nom_fichier);
        exit(1);
    }
    // lecture caractère par caractère
    while ((c = fgetc(fichier)) != EOF) {
        // à completer
    }
```

```
    fclose(fichier);
}
```

Question 1

Calculer l'espace mémoire utilisé par la matrice `corpus`.

Question 2

Compléter la fonction `lire_fichier`. Puis écrire un programme qui charge en mémoire les deux fichiers `pmiserables.txt` et `pscarlett.txt` et qui affiche à l'écran le nombre de phrases.

2 Bigrammes d'une phrase

Un bigramme dans un texte est une suite de 2 lettres consécutives. Par exemple : `to be or not to be` contient les bigrammes `to be or no ot`. Un bigramme peut apparaître plusieurs fois : il y a 2 occurrences du bigramme `to`.

Si p est une phrase, on appelle *matrice de bigrammes* de p une matrice $M \in \mathcal{M}_{26}(\mathbb{R})$, où les lignes et colonnes sont indexées par les lettres de l'alphabet et où $M_{u,v}$ contient le nombre d'occurrences du bigramme uv dans p. Cette matrice récapitule donc le nombre d'occurrences dans la phrase donnée de chacun des diagrammes possibles.

Question 3

Écrire une fonction

```
int bigramme(char *phrase, int M[26][26])
```

prenant en entrée une phrase et construisant sa matrice de bigrammes M. Toutes les cases de M devront être initialisées. La fonction retourne le nombre de bigrammes détectés.

INDICATIONS

Parcourir les suites de deux caractères consécutifs de `phrase` et lorsque ces caractères sont tous les deux des lettres incrémenter la valeur correspondante dans la matrice de bigrammes.

Question 4

Écrire une fonction `double dist(int A[26][26], int B[26][26])` calculant la distance euclidienne entre deux matrices de bigrammes `A` et `B`. Cette distance est donnée par

$$d(A,B) = \sqrt{\sum_{u\in\text{alpha}}\sum_{v\in\text{alpha}}\left(A_{u,v}-B_{u,v}\right)^2}$$

3 Apprentissage supervisé

Question 5

Écrire une fonction

```
Resultat plusprochevoisin(char* phrase,
                          char c1[10000][10000],
                          char c2[10000][10000],
                          int n1,
                          int n2)
```

retournant un triplet `(d, ppv, langue)` où `ppv` est le numéro de phrase la plus proche de `phrase`, `d` est la distance correspondante et `langue` vaut 1 ou 2.

Question 6

En déduire un programme demandant à l'utilisateur de taper une phrase, puis qui affiche la phrase du corpus la plus proche de la phrase tapée. On affichera également la langue détectée.

On souhaite évaluer notre méthode de détection. Pour cela on extrait un petit nombre de phrases de chaque texte pour s'en servir de phrases de test. On souhaite alors construire la matrice de confusion qui en résulte.

Question 7

À la main, extraire une centaine de lignes environ dans chacun des fichiers tests. Les phrases extraites seront enregistrées dans deux nouveaux fichiers `tmiserables.txt` et `tscarlett.txt`. (faire des sauvegardes des fichiers initiaux)

Question 8

Ecrire un programme chargeant ligne par ligne les fichiers de tests et détectant les langues des phrases. On pourra utiliser `fgets` pour lire les fichiers ligne par ligne. Le programme affiche ensuite la matrice de confusion à l'écran.