Groupe 1 :

- BOUBEKEUR Maria

-ADRAR Mahrez

- BENMAMMAR Faycal

- BREKSI REGUIG Ahmed Adel

- ALIYEVA Alessia

- ABOUALI Mohamed

- BELBOUKHARI Adelkader

Rapport mini-projet n°3

-Smart Files-

Groupe n°1

INFOB131 « Introduction à la programmation »

1-Introduction :

Le mini-projet n°3 « Smart Files » a été créé dans le cadre du cours de programmation par Prof.Benoît Frenay pour le Bloc 1 de l'Université de Namur. Celui-ci consiste à réaliser en groupe en langage Python , un trieur de fichiers en utilisant une technique de Machine Learning qui est « Naïve Bayes »

2-Marche à suivre :

Pour créer notre trieur de fichiers on avait à notre disposition 3 répertoires (archive_1, archive_2, archive_3) qui contenait chacun d'eux 2 autres sous-répertoires 'sorted' pour les textes triés et 'unsorted' pour les textes non triées et un fichier labels.txt qui contient les vrais thèmes pour ces textes .
On a donc partager le travail en 2 groupes pour implémenter 2 fonctions :
smart_sort_files (path) : fonction qui trie les fichiers présents dans le sous-répertoire unsorted vers le sous-répertoire sorted pour cela on implémenter 7 fonctions :

1- replace_str (text) : une fonction qui retire la ponctuation , les symboles et les chiffres d'une chaîne de caractères donné
2- get_file_words_list (path) : qui permet de stocker le texte d'un fichier dans une liste après avoir retiré la ponctuation les symboles…
3- delete_repeated_words(list_words) : elle prend en paramètre la liste retourné par la fonction précédente et retire toutes les occurrences d'un mot
4- get_files_words_list() : fonction qui transforme tout les fichiers de tout les archives en liste en retirant les symboles et les mots inutiles
5- get_theme_frequency() : calcule les fréquences de tous les mots d'un fichier unsorted pour chaque thème dans la même archive

pseudo-code :

```
frequencies =  empty dictionary
words = empty list
archives = list of the directories in the current working directory
```

for each archive in the list of archives:
    if archive is not a directory:
        remove archive from the list archives

    else: build a dictionary for archive in the dictionary frequencies

    themes = list of the themes in the directory sorted in archive

    for each theme in the list themes:
        nb_words, the number of words in the theme = 0
        build a dictionary for theme in the nested dictionary archive in frequencies
        files = list of all the files in theme

        for each file in the list files:
            try to open the file
            read the file and store its content in the variable text
            close the file

            if decoding error occurs:
                print 'impossible d'ouvrir le fichier'

            convert the string text to list
            increment nb_words by the number of items in the list text

            for each word in the list words:
                if the word already exists in the nested dictionary theme in frequencies:
                    increment the word's frequency by 1 if the number of occurrences of the word in text = 0 else the word's frequency = number of occurrences of the word in text

                else: the word's frequency = 1 if the number of occurrences of the word in text = 0 else the word's frequency = number of occurrences of the word in text

        for each word in the nested dictionary theme in frequencies:
            divide the number of occurrences of the word by the number of the words to get the frequency

    return the dictionary frequencies

6- Index_of_max(list) :elle retourne l'index de l'élément maximal de la liste elle nous permet de trouver la plus grande fréquence d'un mot pour un thème précis
7- guess_theme_file (path,frequencies,themes): détermine le thème du fichier dans unsorted

    <u>pseudo-code:</u>

convert the string path to list

```
text = call function get_file_words_list(path) to get list of words in text
proba_list = empty list
counter of themes = 0
for each theme in the list of themes of the current archive:
        for each word in the nested dictionary theme in frequencies:
                if word already exists in frequencies:
                        if the number of items in the list of probabilities > counter:
                                increment the list of probabilities by the logarithm of the frequency
of the word
                        else:
                                append the frequency of the word to the list of probabilities
                else:
                        if the number of items in the list of probabilities > counter:
                                increment the list of probabilities by the logarithm of the subtraction
of the frequency of the word to 1
                        else:
                                append the subtraction of thefrequency of the word to 1 to the list of
probabilities

        increment counter by 1
theme_final = theme with the biggest probability in list of probabilities
return theme_final
```

check_accuracy (path) : fonction qui affiche quel pourcentage des fichiers indiquées dans labels.txt sont répartis correctement dans les sous répertoires sorted pour faire ça on a implémenter 4 fonctions :

1- Get_list_of_file_text(path,mode) : fonction qui transforme labels.txt en liste elle retourne le texte en liste
2- Lines_to_list(lines) : fonction qui transforme une liste de chaine de caractères en liste contenant des listes
3- Find_them_in_files (theme,file_name,lines_files):fonction booléenne qui vérifie l'existence d'un mot dans un fichier donnée
4- Build_path_from_list(list) :cette fonction prend en paramètre une liste qui contient la direction pour construire le path (chemin)qu'elle retourne

3-Problèmes rencontrés :

Lors de la réalisation de ce trieur de fichiers on a rencontré des petits problèmes concernant la manipulation des fichiers et des sous-répertoires avec les structures de données (les dictionnaires)

4-Conclusion :

Pour conclure , ce mini-projet n° 3 nous a permis d'apprendre à mieux manipuler les fichiers et de nous familiariser avec ces derniers et de les comprendre en profondeur .