Mini-projet 3 : Files in a Flash

Présentation générale :

Ce module permet de trier des fichiers en fonction de leur thème en utilisant le modèle de machine learning Naive Bayes. Celui-ci est divisé en 6 fonctions qui, ensembles, permettent de trier environ un millier de fichiers en 1 minute 40, avec une fiabilité de 0,934. Deux de ces fonctions sont conçues à l'intention d'un utilisateur externe, tandis que les autres permettent de structurer ou de factoriser le code.

Choix importants :

Pour pouvoir facilement gérer les données et éviter de devoir lire plusieurs fois un même fichier, il a été choisi de stocker la fréquence de chaque mot pour chaque thème dans un grand dictionnaire de dictionnaires. Celui-ci est du format suivant :

{ thème (str) : { mot (str) : fréquence (float) } }

Ce format a l'avantage de permettre un accès direct à une fréquence lors du tri, tout en contenant la liste des mots à vérifier ainsi que la liste des thèmes. La phase d'apprentissage est pourtant très rapide car ce dictionnaire se construit au fur et à mesure, ce qui permet de lire un seul fichier à la fois, une seule fois.

Il a également été choisi de ne pas vérifier si le dossier indiqué en paramètre des fonctions existait et était valide (dans le cadre d'une programmation défensive), car les informations sur ces dossiers se trouvent dans les spécifications des fonctions, qui sont censées être lues par les utilisateurs du module. (Par 'utilisateur', il est ici sous-entendu le programmeur qui importe et utilise le module).

Par ailleurs, aucune fonction récursive n'a été utilisée car cette méthode est plus lente et demande plus de mémoire qu'une simple boucle qui est suffisante pour résoudre les problèmes de Naive Bayes.

Concernant les exemples d'utilisation, il a été choisi de tester le programme sur trois archives différentes dont la première (deux thèmes) et la quatrième (quatre thèmes). Pour tester les limitations du programme et clarifier les améliorations possibles, un test a également été effectué sur une archive composée des quatre archives combinées (onze thèmes). Il apparait que pour un grand nombre de fichiers et de thèmes, le temps nécessaire pour trier grandit fortement et la fiabilité diminue. En modifiant quelques fonctions, il serait possible de les optimiser.

Enfin, pylint a régulièrement été utilisé afin de produire un code aussi propre et conforme aux conventions que possible.

Choix spécifiques pour une fonction :

Le module est divisé en 6 fonctions plus ou moins complexes de manière à factoriser et structurer le code.

- **files_in_a_flash** : Il s'agit de la fonction principale, telle que demandée par le client. Elle trie tous les fichiers du répertoire 'unsorted' vers les sous-répertoires de 'sorted', et affiche le résultat à l'écran. Le code est divisé en deux parties : apprentissage (créer le dictionnaire des fréquences comme précisé plus haut), et tri (utiliser ce dictionnaire pour trier chaque fichier).

- **get_words :** Cette fonction lit un fichier et renvoie une liste contenant tous les mots utiles du texte. Pour déterminer quels mots sont inutiles, une liste de caractères et mots inutiles est créée afin de pouvoir supprimer chacun de ceux-ci dans le texte. C'est une méthode simple et efficace, mais qui comporte cependant un défaut : on ne peut pas prévoir tous les mots, et on ne sait pas s'adapter aux textes. Si besoin, une amélioration est possible en enlevant les mots fréquents communs entre les différents thèmes (pour un mail, un exemple est par exemple : « Subject ») et les mots extrêmement peu représentés. Cette méthode permettrait en théorie de faire monter la fiabilité de 0.93 à environ 0.97.

- **get_frequencies** : Cette fonction crée le dictionnaire des fréquences (décrit dans la section 2) et le renvoie. Elle utilise les sous-fonctions get_words et check_differences. Il a été choisi de construire le dictionnaire au fur et à mesure, en lisant chaque fichier une et une seule fois. Son exécution prend généralement entre 6 et 10 secondes et constitue l'ensemble de la partie « apprentissage ».

- **check_differences** : Bien qu'appelée une seule fois au sein de get_frequencies, il a été choisi de séparer cette fonction afin de clarifier le code et de le structurer. Elle permet d'allonger les sous-dictionnaires (fréquences par mot d'un thème) jusqu'à ce que tous les mots soient représentés dans chaque thème. Bien que son exécution puisse prendre du temps pour un nombre de mot élevé (ledit nombre pouvant baisser en utilisant les méthodes d'optimisation mentionnées dans le descriptif de get_words), ce temps a semblé plus court que celui de lire plusieurs fois chaque fichier.

- **get_theme_prob :** Cette fonction calcule et renvoie la probabilité qu'un fichier appartienne à un thème donné. Grâce au format de stockage des fréquences, il suffit de vérifier si chaque mot du dictionnaire est présent dans le fichier, et calculer le logarithme de la probabilité en fonction.

- **check_accuracy :** Cette fonction vérifie simplement si les fichiers sont triés correctement afin de calculer la précision de l'implémentation. Aucun choix particulièrement intéressant n'a dû être pris dans ce cas.

Pseudo-code de la fonction get_frequencies :

```
Input: path to the directory with the already sorted files
Output: dictionary with the appearance's probability of a list of words in a theme
Create empty general dictionary
For each theme do
    Create the empty theme's dictionary in the general dictionary
    For each file of the theme do
        For each word of the file of the theme do
            If word ∈ theme's dictionary then
                Add 1 to the number of occurrences of this word
            Else
                Add the word to the theme's dictionary
                and set its number of occurrences to 1
            End if
        End for
    End for
End for
Pool all the themes' dictionaries
And set the number of occurrences of the added words in the dictionaries
For each theme do
    For each word in the theme's dictionary do
        Number of occurrences/number of files in the theme (= probability)
    End for
End for
Return general dictionary with all the theme's dictionary
```