**Rapport Projet Python**

1. Description du projet

Le projet réalisé dans le cadre du cours de Python consiste en la création de deux outils distincts : l’un permettant l’aspiration de textes sur un site Web, l’autre permettant la classification de ces textes en fonction de leur thème. Pour la première partie du projet nous avons choisi de récolter une vingtaine d’articles de la rubrique « Culture » du journal en ligne Le Monde (en ligne : lemonde.fr). Dans la seconde partie du projet, nous avons décidé de les classer en fonction des cinq catégories : Art (sculpture, peinture, photographie etc.), Scène (théâtre, opéra etc.), Musique, Cinéma et Littérature.

1. Packages utilisés

Pour la partie de crawling, nous avons utilisé la bibliothèque BeautifulSoup. Ce module peut être téléchargé via Pip. C’est ce module qui permet de récupérer le contenu html d’une page et de manipuler les différentes balises pour récupérer les informations nécessaires, principales et utiles pour notre tâche. Nous avons également utilisé le module requests afin de faire une requête à partir de l’url de la page que l’on souhaite récupérer. Enfin nous avons utilisé le moule os pour manipuler les dossiers (création des dossiers qui contiendront le corpus de la première partie et les textes classés pour la seconde partie du projet). Ces deux modules sont déjà disponibles avec la version de base de python 3.

Pour la seconde partie, nous avions tout d’abord comme objectif d’utiliser le module Scikit-learn. Nous avons abandonné l’idée car il s’agit d’un module de Machine Learning et il nous aurait donc fallu beaucoup trop de données d’apprentissage annotées pour réaliser l’apprentissage du module ce qui demande beaucoup trop de temps pour mettre en œuvre cette partie par rapport au temps imparti pour la réalisation du projet. Nous avons donc décidé de rester sur un modèle utilisant un lexique pour faire une recherche de champ lexical dans les textes tokenisés avec le module Spacy.

1. le découpage en modules, les chaines de traitements mises en œuvres (faites des schémas)

Partie 1 :

Module crawl\_lemonde()

Module contenuHTML()

Récupérer le contenu de la page html (*Module contenuHTML()*)

Récupérer une page à partir d’une url

Trouver les balises contenant les url des articles

Récupérer son contenu

Pour chaque lien extraire : titre, auteur, date et corps du texte

Parser son contenu en fonction des balises

Créer le dossier et y enregistrer un fichier txt avec ces infos

Partie 2 :

Module classeText()

Créer le dossier du thème d’appartenance et enregistrer le fichier

Créer le dossier de classification, enregistrer le fichier de stats

Pondérer le thème du titre et trouver à nouveau le thème d’appartenance du texte

Donne le pourcentage le plus haut à partir d’une liste (fonction de python max())

Module nbMax()

Trouver le pourcentage le plus haut

Module trouveClasse()

Trouver les thèmes d’appartenance du titre et du corps (Module trouveClasse())

Compte le nombre total d’occurrence de mots appartenant à chaque thème et en fait un pourcentage

Enregistre dans un dictionnaire les mots correspondant à chaque thème

Module chargeChampsLexicaux()

Trouve les mots appartenant aux différents thèmes avec leur occurrence

Charger les champs lexicaux depuis les listes dans les fichiers txt

Module champLexTexte()

Trouver les champs lexicaux du titre et du corps (Module champLexTexte())

Transformation token en string et enregistrement dans une liste

Tokenisation avec spacy

Module tokeniser()

Crée un tableau avec le titre à l’indice 0 et le contenu à l’indice 1

Tokeniser le texte (Module tokeniser())

Module openFiletoClass()

Ouvrir le fichier et enregistrer son titre et son contenu (Module openFiletoClass())

1. Choix algorithmiques et principales structures de données

Pour la première partie, deux fonctions sont utilisées l’une utilisant requests pour récupérer une page, mettre le contenu de cette page html dans une variable et parser le contenu html à l’aide de BeautifulSoup en faisant une soup (la structure de données est donc ici un objet de type BeautifulSoup). L’autre fonction récupère les liens présents dans les balises d’une page html et enregistre dans un fichier texte le contenu (titre, auteur, date, contenu) de chaque page dans un dossier « Corpus » et dans un fichier au format txt.

Pour la deuxième partie, une fonction permet d’ouvrir un fichier, d’en extraire le titre et le reste du fichier, ces deux éléments sont contenus dans un tableau avec en premier indice le titre et en deuxième indice le reste du contenu du fichier. Afin de relever les mots qui seront important pour la classification nous avons créé une autre fonction permettant de tokeniser le texte à l’aide de l’outil Spacy et qui retourne la liste des tokens (liste de strings). Pour classer les textes en fonction de nos cinq catégories (art, cinéma, littérature, musique, scène) nous avons décidé d’utiliser un dictionnaire des champs lexicaux de chaque catégorie (des fichiers textes qui contiennent un mot par ligne appartenant au domaine en question). Chaque mot de chaque texte est alors comparé au dictionnaire (celui-ci est chargé à l’aide d’une fonction qui retourne une table de hachage de la forme {art : {mot1 :"", mot2 : "" }, musique : {mot1 : ""}…}) à l’aide d’une fonction qui retourne une table de hachage contenant tout le vocabulaire appartenant à l’un des cinq thèmes pour ce texte-là. Quand l’un des mots est identique, il est ajouté à un dictionnaire avec le thème en question et le nombre de fois où ce mot est présent dans le texte (par exemple le mot *caméra* sera référencé de la manière suivante s’il apparait trois fois dans le texte : {caméra : {cinéma : 3}}). On va également utiliser ces fonctions pour trouver le thème d’appartenance du titre, une fois qu’on a trouvé ce thème on lui donne un poids plus important dans les résultats (ajoute par exemple 15% à la proportion de mots appartenant à cette catégorie car en général il est possible de connaitre le thème d’un texte en ne lisant que le titre, cela permet de trancher en cas d’égalité des scores : par exemple si le titre est classé dans cinéma et que la proportion de mots appartenant à cinéma est 43%, ce pourcentage passe à 58%). La fonction trouveClasse() permet de donner ces résultats : elle donne le thème d’appartenance en calculant les pourcentages d’appartenance à chaque thème pour un texte donné (ou un titre), elle renvoie également un dictionnaire contenant les résultats : {"Cinéma": nbcinema / nbMots , "Art": nbart / nbMots ,…}. Une fois qu’on a calculer ces pourcentage pour le texte et le titre, et qu’on a pondéré le thème dont le titre fait partie, on sélectionne le thème avec le pourcentage le plus élevé : tout cela se fait dans la fonction principale classeText(). Ensuite cette fonction crée le dossier qui contiendra les textes classés, les sous-dossiers correspondants aux différents thèmes et elle classe les textes dans les dossiers correspondant à leur thème en fonction du plus haut pourcentage d’appartenance (si le texte 1 a une majorité de mots appartenant au thème cinéma il sera classé dans le dossier cinéma). Cette fonction enregistre enfin un fichier txt stats.txt contenant tous les pourcentages pour chaque texte :

*Pourcentages d'appartenance aux thèmes pour le fichier text1.txt:*

*Cinéma : 33.25%*

*Art : 14.25%*

*Scènes : 7.13%*

*Musique : 21.38%*

*Littérature : 15.00%*

1. Evaluation des sorties

La première partie du projet donne le corpus sur lequel nous nous sommes appuyées dans la seconde partie. Nous avons donc sélectionné vingt textes du journal LeMonde. Nous avons contrôlé les sorties de ce premier script et toute ont la forme voulue : Le titre, l’auteur, la date et le contenu séparé par des sauts de ligne.

Pour la seconde partie, nous avons évalué manuellement les résultats des sorties : sur les vingt fichiers extraits du journal à l’aide du crawler de la partie 1, seulement deux textes sont mal catégorisés : text11 appartient à la catégorie arts mais est catégorisé comme musique et text17 catégorisé comme musique appartient à littérature. Le texte 11 est en effet difficile à classer, c’est un texte qui parle de jardins et de plantations d’arbres, et utilise très peu de vocabulaire pour chaque thème (probablement un ou deux mot pour chacun), ce qui est confirmé par les statistiques suivantes :

*Pourcentages d'appartenance aux thèmes pour le fichier text11.txt:*

*Cinéma : 24.39%*

*Art : 24.39%*

*Scènes : 24.39%*

*Musique : 24.39%*

*Littérature : 15.00%*

Le texte 17 est plus énigmatique, en le lisant on voit mal quels sont les mots qui ont fait pencher la balance pour la musique.

Cependant, les textes restent très courts ce qui rend la tâche plutôt difficile mais le script semble toutefois donner des résultats plutôt satisfaisant compte tenu du temps de réalisation et de la simplicité des règles qui ont permis la classification des textes. On obtient en effet 90% de bonnes réponses en moyenne.

1. Bogues constatés

Nous n’avons pas constaté de bogues lors de l’exécution du projet. Toutes les fonctionnalités marchent comme nous le souhaitions.

1. Les améliorations à apporter et extensions prévues.

Pour améliorer ce projet, il faudrait plus de catégories avec plus de mots dans chacune d’elles. Le choix des mots dans les catégories doit en effet être plus fin pour couvrir le maximum de vocabulaire et affiner le choix final des thèmes. Le problème reste également la taille des textes, plus les textes sont courts, moins on trouve de mots appartenant à un thème, et plus difficile il est de classer ce texte car on augmente les chances d’avoir des égalités de score.

Nous pourrions donc faire comme nous avions initialement prévu : en utilisant le module Scikit-learn. Avec Scikit-learn, il y a beaucoup de données d’apprentissages annotées et donc les données de test seront beaucoup plus précises. L’évaluation serait donc meilleure, la taille du texte entrerait aussi en compte mais on peut espérer que cela aura moins d’incidence sur le résultat.