Projets en Programmation Python

Préambule

Deux sujets sont proposés cette année. La description ci-dessous donne des indications qu'il faut tâcher de suivre au maximum, mais l'important est de démontrer votre maîtrise des éléments fondamentaux de la programmation (notions de classes et d'héritage, polymorphisme, gestion des collections, modularisation, etc.). Aussi, ne perdez pas de vue qu'il vaut mieux aboutir à un programme robuste qui assure le minimum de fonctionnalité plutôt qu'un logiciel qui propose de nombreuses fonctionnalités mais mal implémentées. Il est également encouragé de faire vos propres propositions afin d'améliorer la qualité de votre programme et le rendre plus intéressant.

Des consignes plus précises concernant le rendu attendu sont données dans un document à part.

1 Premier sujet : analyse comparative de corpus

Les documents contiennent énormément d'informations pertinentes pour les chercheurs en SHS (sociologues, linguistes, etc.). Néanmoins, la masse de données rend compliquée leur exploration. Pour cela, vous créerez un outil permettant d'explorer les documents en adoptant une approche comparative.

La première piste pour créer ce type d'outil est de permettre une analyse comparative de deux corpus (par ex. comparer les articles issus de Reddit de ceux d'Arxiv). La seconde piste consiste à permettre d'observer l'évolution temporelle d'un mot (ou d'un groupe de mots) donné. Il faut alors être capable de traiter le champ date pour découper la frise temporelle en périodes. Dans les deux cas, il s'agit d'observer l'importance relative d'un ou plusieurs mots dans un corpus vs. un autre corpus : quels sont les mots communs ? les mots spécifiques ?

Au-delà de la simple fréquence d'un mot (TF), des mesures de l'importance d'un terme dans un corpus sont faciles à coder. La mesure comme TFxIDF (`https://fr.wikipedia.org/wiki/TF-IDF`) et OKAPI-BM25 (`https://fr.wikipedia.org/wiki/Okapi_BM25`) constituent une bonne base de départ. Vous pourrez proposer une analyse du vocabulaire en utilisant ces scores. Il semble cependant nécessaire de les adapter un peu, par ex. en considérant un corpus comme un "gros" document.

Ce projet pose également la question de l'affichage du résultat afin de le rendre digeste. Il existe plusieurs solutions, comme par exemple utiliser une interface via tkinter (`https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/234859-creez-des-interfaces-graphiques-avec-tkinter`).

2 Deuxième sujet : extraction de collocations

Un graphe est un objet mathématique défini par un ensemble de noeuds et un ensemble de liens entre ces noeuds (arcs dans le cas d'un graphe dirigé, arêtes sinon). Dans ce projet, on souhaite construire un graphe des mots du corpus : deux mots apparaissant dans le même document seront liés (on parle de co-occurrence). Le graphe est dit pondéré lorsqu'un poids est associé au lien : dans notre cas, il est égale au nombre de fois où les mots ont co-occurré. Développez un outils de visualisation de ce graphe de co-occurrences, pour pouvoir l'explorer (avec filtre, recherche etc...). Vous pouvez utiliser n'importe quelle libraire Python (`https://towardsdatascience.com/python-interactive-network-visualization-using-networkx-plotly-and-dash-e44749161ed7`). Prévoyez la gestion de l'import et la sauvegarde du graphe. Vous pouvez aller plus loin en affichant des mesures de centralité, des communautés... que vous calculerez au moyen d'algorithmes sur les graphes. Pour aller encore plus loin, vous pouvez extraire les expression courantes, ou collocats, c'est à dire les suites de mots qui occurrent souvent (par ex. : "c'est-à dire"). Pour cela, vous utiliserez la méthode de la PMI (`http://mlwiki.org/index.php/Collocation_Extraction` et `https://en.wikipedia.org/wiki/Pointwise_mutual_information`).

Toutes les idées originales sont bien sûr les bienvenues !

Master 1ère année – Université Lumière (Lyon 2) © 2020
Enseignants : Julien Velcin, Antoine Gourru