RAPPORT AIDE A LA REDACTION

Antoine CLOP - Corentin REDON - Arthur JOLY

13 Juillet 2018

Table des matières

1 Introduction

Dans le cadre du cours de Méthodes d'optimisation pour les moteurs de recherche, nous avons choisis de réaliser un système d'aide à la rédaction. Le but du programme est de prendre un ensemble de textes en entrée et d'en extraire les n-grammes les plus signifiants.

2 Notions

2.1 N-Grammes

Un N-Gramme est une sous-séquence d'éléments. L'idée est de pouvoir prédire de façon probabiliste les éléments suivants. Dans notre cas, nos éléments sont tout simplement des mots. L'objectif est ainsi de constituer des groupes de mots (ou un mot uniquement) afin de pouvoir trouver les mots ou enchaînement de mots se répétant le plus souvent et ayant par conséquence plus de chance d'apparaître dans un texte.

2.2 TF-IDF

Pour nous aider à constituer ces N-Grammes, nous utilisons une autre méthode : TF-IDF.

Le TF-IDF (signifiant *Term Frequency-Inverse Document Frequency*) est une méthode apparue à la fin du 20^{e}permettant d'évaluer l'importance d'un mot dans un texte de façon statistique. Il existe plusieurs variantes de ce calcul, permettant d'ajuster la pertinence par rapport aux besoins du projet :

- Méthode binaire
- Méthode fréquence brute
- Méthode normalisation logarithmique
- Méthode normalisation 0.5 par le max
- Méthode par le max

Dans notre cas, nous utiliserons la méthode classique du TF-IDF.

2.2.1 Méthode de calcul

Pour commencer, il faut déterminer le TF (Term Frequency) de chaque mot. Il s'agit littéralement d'obtenir la fréquence d'apparition d'un mot dans

un texte. Pour le calculer, il suffit donc de compter le nombre d'apparitions de chaque mot et de le diviser par le nombre de mots au total dans le texte.

$$\mathrm{tf}_{1,1} = \frac{n_{1,1}}{\sum_k n_{k,1}}$$

avec tf1,1 le TF du 1ermot dans le 1ertexte, n11 le nombre d'apparition du 1ermot dans le 1ertexte et k le nombre de mots du 1ertexte.

Puis il nous faut déterminer l'IDF (Inverse Document Frequency) de chaque mot. Il s'agit donc ici de calculer le logarithme du rapport nombre de textes au total divisé par nombre de textes contenant le mot dont on cherche l'IDF. Cela revient à effectuer :

$$\mathrm{idf}_1 = \log \frac{|D|}{|\{d_j : t_1 \in d_j\}|}$$

avec idf1, l'IDF du 1ermot, D le nombre de textes au total, t1 le 1ermot et dj le texte j.

Enfin, le calcule du TF-IDF revient à faire le produit du TF et de l'IDF, soit :

$$\mathrm{tfidf}_{\mathrm{i,j}} = \mathrm{tf}_{\mathrm{i,j}} \cdot \mathrm{idf}_{\mathrm{i}}$$

avec tfidfij le TF-IDF du mot i dans le texte j, tfij le TF du mot i dans le texte j et idfi l'IDF du mot i.

3 Implémentation

3.1 Usage

Notre implémentation TF-IDF est en Python, pour faciliter le développement sur plusieurs plate-formes et résoudre certaines problématiques comme la gestion de l'UTF-8.

Le programme requiert en premier paramètre le nom du dossier contenant le corpus de textes. Il est possible de préciser deux paramètres tels que le nom du fichier csv de sortie ainsi que la taille des ngrams (ayant pour valeur respectivement "output.csv" et 1 par défaut). On a ainsi :

$ main.py TEXT_FOLDER [OUPUT_FILE='output.csv'] [NB_GRAM=1]

Nous essayerons ici notre programme sur 3 courtes phrases :
- Hello World !
- Hello Word ! Hello
- Hellow Hellow

3.2 TFIDF

La première étape effectuée est une préparation du calcul de l'IDF. Cette étape va comptabiliser le nombre de fichiers présents dans le corpus et le nombre de fichiers contenant chaque mot.

Les textes sont 1 à 1 coupés en tokens selon les étapes suivantes : passage en lettres minuscules, suppression des fins de lignes, suppression des mots contenant 1 lettre et enfin découpage en mot en retirant la ponctuation qui pourrait être les toucher.

On obtient ainsi les résultats suivant :

```
Processing IDF 1gram on short_sentence/sentence1.txt file
Processing IDF 1gram on short_sentence/sentence2.txt file
Processing IDF 1gram on short_sentence/sentence3.txt file
Counter({('hello',): 2, ('world',): 1, ('hellow',): 1, ('word',): 1})
```

La seconde étape est le calcul du TF pour chaque texte du corpus. On compte donc le nombre d'occurrence de chaque mot dans ce texte et on divise ce nombre par le nombre total de mots présents.

On obtient ainsi les résultats suivant :

```
Processing TF 1gram on short_sentence/sentence1.txt file
{('hello',): 0.5, ('world',): 0.5}
Processing TF 1gram on short_sentence/sentence2.txt file
{('word',): 0.3333333333333333, ('hello',): 0.6666666666666666}
Processing TF 1gram on short_sentence/sentence3.txt file
{('hellow',): 1.0}
```

La dernière étape est finalement le calcul du TF-IDF. Pour le mot *hello*, on effectue ainsi :

```
tfidf["hello"] = tf["hello"] * math.log(nb_file, idf["hello"])
```

Le résultat final est écrit dans le fichier de sortie. On obtient ainsi un fichier csv contenant les valeurs de TF-IDF pour chaque mot dans chaque fichier :

```
# Fichier sentence1.txt
('world'),0.23856062735983122
('hello'),0.08804562952784062

# Fichier sentence2.txt
('word'),0.15904041823988746
('hello'),0.11739417270378749

# Fichier sentence3.txt
('hellow'),0.47712125471966244
```

Dans notre exemple, on peut donc en conclure que le mot *'word'* aurait plus de chance d'être suggéré dans une aide à la saisie car il obtient le plus grand TF-IDF dans deux textes sur trois.

4 Résultats

Pouvoir suggérer les mots les plus couramment utilisés est déjà un bon point. Mais il peut être plus pertinent de pouvoir prédire les prochains mots qui seront saisis. Pour y parvenir, on peut modifier le ngram. En le mettant à 2, on obtient ainsi les couples de mots revenant le plus souvent. Ainsi, si l'utilisateur écrit le mot *'hello'*, on peut estimer que le mot *'world'* a le plus chance d'être le mot suivant.

```
# Fichier sentence1.txt
('hello', 'world'),0.23856062735983122

# Fichier sentence2.txt
('hello', 'word'),0.15904041823988746
('word', 'hello'),0.15904041823988746

# Fichier sentence3.txt
('hellow', 'hellow'),0.23856062735983122
```

On pourrait être plus critique avec nous-même sur les performances. En effet notre programme passe deux fois sur chaque fichier du corpus. La première fois pour voir quels mots sont présent dans le fichier et une deuxième fois pour déterminer la fréquence d'apparition de ce mot dans le texte. Une façon de pallier à ce problème serait de compter le nombre d'occurrence de chaque mot par fichier au tout début, mais malheureusement la façon dont nous avons organisé le code ne prévoyait pas ce défaut. Néanmoins pour la taille de nos fichiers, les performances sont suffisamment bonnes pour obtenir les résultats presque instantanément.

5 Conclusion

Pour notre aide à la saisie, la méthode du TF-IDF nous permet de retrouver les mots ou enchaînements de mots les plus fréquents dans un texte. Les résultats étant ensuite exportés sous forme de fichier csv, ils peuvent être intégrés dans n'importe quel logiciel.

En revanche si l'on voulait ajouter des textes au corpus analysé par le programme, il faudrait re-calculer tous les TF-IDF et l'on ne pourrait pas simplement repartir des valeurs du csv.