13 Novembre 2025

Nom:
Prénom:

Python

/ 20

Examen pratique

Sujet

La mesure *TF-IDF* permet de détecter un mot exceptionnellement fréquent dans un texte.
C'est une méthode de NLP très utilisée pour générer des *mots clés* à partir d'un texte.

On peut comparer un texte à un autre texte, ou simplement le comparer à la *fréquence attendue* de ces mots dans la langue étudiée.
Nous utiliserons ici du contenu provenant de **Wikipédia**, en comparant les articles entre eux

Le code doit être écrit en utilisant un **bloc note** de base de votre OS
Tout IDE (éditeur spécialisé dans le code) est interdit

L'utilisation de logiciels de chat ou d'un navigateur Internet n'est pas autorisé.
En revanche vous pouvez poser des questions

Aucune librairie externe n'est requise, aucune utilisation de l'IA ne sera toléré

Vous trouverez des aides mémoire pour Python en annexe de ce document

Question 1

/ 3.75 pt

Récupérer le corpus de textes à analyser depuis CESAR

1. Créer une classe `TfIdf` vide, et y ajouter une méthode `add_corpus`.

Cette fonction lis le fichier contenant le texte d'une page à partir de son nom.

2. Dans la méthode `add_corpus`, comptez les mots du texte importé

Le comptable de chaque mot doit être réalisé dans un dictionnaire.

En fin de méthode, on ajoutera ce dictionnaire à l'attribut `self.corpus` de la classe, contenant le dictionnaire de mots comptés pour chacun des corpus analysés

3. Modifier la méthode `add_corpus` pour comptez l'ensemble des mots sur tous les corpus

À chaque mot rencontré dans un corpus, en plus de l'incrémentation du dictionnaire local, incrémenter aussi un compteur global à la classe dans l'attribut `self.words`

Example en fin d'implémentation

```
t = TfIdf()
t.add_corpus("toto") # Texte:  je suis une truite
t.add_corpus("tutu") # Texte:  je suis un saumon

print(t.words)
# { "je": 2, "suis": 2, "un": 1, "une": 1", "truite": 1", "saumon": 1 }
print(t.corpus)
# {
#   "toto": { "je": 1, "suis": 1, "une": 1, "truite": 1 },
#   "tutu": { "je": 1, "suis": 1, "un": 1, "saumon": 1 },
# }
```

4. Ajouter à l'attribut `self.tot_words` le nombre de mots contenus dans chaque corpus

Pour cela, dans la méthode `add_corpus`, compter le nombre de mots total rencontré dans le texte, et ajouter ce nombre dans l'attribut `self.tot_words` pour le corpus actuel

5. Créer un objet `TfIdf`, et ajouter tous les corpus téléchargés depuis CESAR

Question 2

/ 5 pt

1. Créer une méthode `get_idf`, prenant en argument un mot

Compter dans combien de corpus ce mot apparaît.

2. Puis, calculer l'IDF de ce mot

Pour un mot w apparaissant dans C_w corpus (sur les C_{tot} corpus existants):

$$\text{IDF} = \log\left(\frac{C_{\text{tot}}}{C_w}\right)$$

Question 3

/ 6.25 pt

1. Créer une nouvelle méthode `get_tf_idf`, prenant en argument le nom d'un corpus

Dans cette méthode, pour chaque mot du corpus:
- Caculer la fréquence du mot dans le corpus: $\text{Tf} = \frac{N_{\text{mot}}}{N_{\text{tot}}}$
- Calculer le *TF-IDF* de ce mot: $\text{TfIdf} = \text{Tf} \times \text{Idf}$

La méthode `get_tf_idf` retourne un dictionnaire contenant le TFIDF de chaque mot

2. Remplir les trous

Pour chacun des mots rencontrés dans tous les corpus:
- Si ce mot n'est pas présent dans le corpus analysé, lui assigner une valeur $\text{TfIdf} = 0$

Question 4

/ 5 pt

1. Implémentez la méthode `cos_sim`, prenant en paramètre le nom de 2 corpus de textes, et retournant la valeur numérique de leur *similarité*.

Pour calculer la similarité entre 2 corpus de texte, on calcule la *cosine similarity* des vecteurs de *TF-IDF* des corpus. (respirez, tout va bien se passer)

La formule de la similarité est la suivante, pour deux corpus A et B:

$$\frac{A \cdot B}{\|A\| \times \|B\|}$$

Où:

- $A \cdot B = \sum_w \left(t_{a_w} \times t_{b_w} \right)$ où t_{w_a} est le TF-IDF du mot **w** sur le corpus **A**
- $\|A\| = \sqrt{\sum_w (t_w^2)}$ où t_w est le TF-IDF du mot **w** sur le corpus **A**

2. Tester la similarité entre des articles similaires (`Deafness` et `Sign_language`), puis entre des articles plus éloignés

Annexes

Lire un fichier

```
with open("<chemin de mon fichier>", "r") as f:
  data = f.read()
```

Séparer un texte par caractère

```
text = "a,b,c,d,e,"
text.split(",")  # Donnera la liste ["a", "b", "c", "d", "e", ""]
```

Itérer sur les valeurs d'un dictionnaire

```
d = {"a": 3, "b": 4}

for k in d.keys():
  print(k)  # Affichera "a" puis "b"

for v in d.values():
  print(v)  # Affichera "3" puis "4"

for (k, v) in d.items():
  print(k, "=", v)  # Affichera "a = 3" puis "b = 4"
```

Opérations mathématiques

```
import math
math.log(3)   # log(3)
math.sqrt(2)  # racine carée de 2
3 ** 5        # 3 puissance 5
```