Matrices terme-document

April 2, 2024

Pondération tf-idf

Ce poids dépend de la fréquence du terme t dans le document d (tf, pour *term frequency* et du nombre des documents D qui contiennent le mot (idf, pour 'inverse doc frequency')) :

$$tf - idf_{t,d} = tf_{t,d} \cdot idf_{t,D} \tag{1}$$

où **tf** est la fréquence de t dans d, et

$$idf_{t,D} = log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{2}$$

Sélectionner des cellules d'une matrice selon des critères

```
>>>
>>>
>>>
>>> x
array([[3, 4, 5],
       [6, 7, 8]])
>>> x[x==4]
array([4])
>>>
>>> x[x>4]
array([5, 6, 7, 8])
>>>
>>>
>>> _
```

- x[x> 4] retourne **les éléments de x** qui sont > 4
- np.argwhere(x>4) retourne **les indices** des éléments de x qui sont > 4
- np.amax(x) retourne la valeur maximale dans x
- np.argwhere(x==np.amax(x)) retourne l'indice de l'élément avec la valeur maximale

Exercices.

1. Nous allons construire une matrice terme-docuemnt à partir d'un extrait du corpus Reuters dans NLTK (400 documents numérotés de 0 à 399). Téléchargez les fichiers 'reuters', term-doc.py, terms.lst.

   - Les lignes de la matrice correspondent aux documents, les colonnes aux termes, les cellules au nombre d'occurrences du terme dans le document. Seul les 5000 termes figurant dans la liste terms.lst sont pris en compte. Complétez term-doc.py selon les instructions dans le fichier pour créer la matrice.

```
matrix = np.zeros([400,5000])

f = open("terms.lst", "r")
for line in f:
  word.append(line.rstrip())
f.close

for n in range (0,399):
  filename = str("reuters/" + str(n) + ".txt")
  f = open(filename,"r")
  print("Processing file " + filename)
  for line in f:
    for w in word_tokenize(line):
      if w in word:
        matrix[n,word.index(w)] += 1
```

2. Complétez term-doc.py pour retourner :

   - le nombre maximum d'occurrences d'un mot dans un document observé dans le corpus

     print(str(np.amax(matrix)))

   - le nombre maximum d'occurrences d'un mot précis dans un document du corpus

     x = input("enter a word:")
     print("max nb of occurrences of the word in a doc: " + str(np.amax(matrix[:,word.index(x)])))

   - pour un mot précis, le document dans lequel il a le plus d'occurrences

     x = input("enter a word:")
     print("doc with the max of occurrences: " + str(np.argwhere(matrix[:,word.index(x)]==np.amax(matrix[:,word.index(x)]))))

   - pour un mot précis, le nombre de documents dans lesquels il apparaît

     x = input("enter a word:")
     print("nb of docs that contain this word: " + str(np.shape(matrix[:,word.index(x)][matrix[:,word.index(x)]>0])[0]))

3. Créez une deuxième matrice dans laquelle le nombre d'occurrences est remplacé par 1 si le mot apparaît dans le document, et par 0 s'il n'apparaît pas.

   solution 1 :
   matrix2 = (matrix>0).astype(int)

   solution 2:
   matrix2 = np.zeros(np.shape(matrix)).astype(int)
   matrix2[matrix>0] = 1

4. Créez une troisième matrice dans laquelle le nombre d'occurrences est remplacé par le poids tf-idf du terme dans le document.