lab3_TFIDF_2

September 25, 2023

[1]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

[94]:
```
import pandas as pd
```

[31]:
```
documentA='the man went out for a walk'
documentB='the children sat around the fire'
```

[32]:
```
#create bag of words
bagofwordsA=documentA.split()
bagofwordsB=documentB.split()
```

[33]:
```
bagofwordsA
```

[33]:
```
['the', 'man', 'went', 'out', 'for', 'a', 'walk']
```

[34]:
```
bagofwordsB
```

[34]:
```
['the', 'children', 'sat', 'around', 'the', 'fire']
```

[35]:
```
#to find the unique values
uniquewords=set(bagofwordsA).union(set(bagofwordsB))
uniquewords
```

[35]:
```
{'a',
 'around',
 'children',
 'fire',
 'for',
 'man',
 'out',
 'sat',
 'the',
 'walk',
 'went'}
```

[36]:
```
#Next,we'll create a dictionary of words and their occurence for each documnet␣
 ↪in the corpus
```

[37]:
```
numofwordsA=dict.fromkeys(uniquewords,0)
numofwordsA
```

[37]:
```
{'fire': 0,
 'walk': 0,
 'for': 0,
 'man': 0,
 'went': 0,
 'children': 0,
 'a': 0,
 'sat': 0,
 'the': 0,
 'around': 0,
 'out': 0}
```

[38]:
```
#TF of each term for document 1
```

[39]:
```
for word in bagofwordsA:
    if word in numofwordsA:
        numofwordsA[word]+=1

numofwordsA
```

[39]:
```
{'fire': 0,
 'walk': 1,
 'for': 1,
 'man': 1,
 'went': 1,
 'children': 0,
 'a': 1,
 'sat': 0,
 'the': 1,
 'around': 0,
 'out': 1}
```

[40]:
```
#TF of each terms for document 2
```

[41]:
```
numofwordsB=dict.fromkeys(uniquewords,0)
numofwordsB
```

[41]:
```
{'fire': 0,
 'walk': 0,
 'for': 0,
 'man': 0,
 'went': 0,
 'children': 0,
 'a': 0,
 'sat': 0,
```

```
 'the': 0,
 'around': 0,
 'out': 0}
```

[42]:
```
for word in bagofwordsB:
    if word in numofwordsB:
        numofwordsB[word]+=1

numofwordsB
```

[42]:
```
{'fire': 1,
 'walk': 0,
 'for': 0,
 'man': 0,
 'went': 0,
 'children': 1,
 'a': 0,
 'sat': 1,
 'the': 2,
 'around': 1,
 'out': 0}
```

[48]:
```
#removing the stop words
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[48]:
```
True
```

[50]:
```
stop_words=stopwords.words('english')
```

[51]:
```
stop_words
```

[51]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
```

```
"you'll",
"you'd",
'your',
'yours',
'yourself',
'yourselves',
'he',
'him',
'his',
'himself',
'she',
"she's",
'her',
'hers',
'herself',
'it',
"it's",
'its',
'itself',
'they',
'them',
'their',
'theirs',
'themselves',
'what',
'which',
'who',
'whom',
'this',
'that',
"that'll",
'these',
'those',
'am',
'is',
'are',
'was',
'were',
'be',
'been',
'being',
'have',
'has',
'had',
'having',
'do',
'does',
```

```
'did',
'doing',
'a',
'an',
'the',
'and',
'but',
'if',
'or',
'because',
'as',
'until',
'while',
'of',
'at',
'by',
'for',
'with',
'about',
'against',
'between',
'into',
'through',
'during',
'before',
'after',
'above',
'below',
'to',
'from',
'up',
'down',
'in',
'out',
'on',
'off',
'over',
'under',
'again',
'further',
'then',
'once',
'here',
'there',
'when',
'where',
'why',
```

```
'how',
'all',
'any',
'both',
'each',
'few',
'more',
'most',
'other',
'some',
'such',
'no',
'nor',
'not',
'only',
'own',
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
'd',
'll',
'm',
'o',
're',
've',
'y',
'ain',
'aren',
"aren't",
'couldn',
"couldn't",
'didn',
"didn't",
'doesn',
"doesn't",
```

```
 'hadn',
 "hadn't",
 'hasn',
 "hasn't",
 'haven',
 "haven't",
 'isn',
 "isn't",
 'ma',
 'mightn',
 "mightn't",
 'mustn',
 "mustn't",
 'needn',
 "needn't",
 'shan',
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

[70]:
```
#Calculating the TF s (1+log(tf))
#but we are not using that equation in here
#we are using "tfDict[word]=count/float(bagofwordsCount)"
```

[65]:
```
def computeTF(wordDict,bagofwords):
    import math
    tfDict={}

    bagofwordsCount=len(bagofwords)

    for word,count in wordDict.items():
        tfDict[word]=count
        tfDict[word]=count/float(bagofwordsCount)

    return tfDict
```

[66]:
```
#the following lines compute the term frequency for each of our documents
tfA=computeTF(numofwordsA,bagofwordsA)
tfB=computeTF(numofwordsB,bagofwordsB)
```

[67]:
```
tfA
```

[67]:
```
{'fire': 0.0,
 'walk': 0.14285714285714285,
 'for': 0.14285714285714285,
 'man': 0.14285714285714285,
 'went': 0.14285714285714285,
 'children': 0.0,
 'a': 0.14285714285714285,
 'sat': 0.0,
 'the': 0.14285714285714285,
 'around': 0.0,
 'out': 0.14285714285714285}
```

[68]:
```
tfB
```

[68]:
```
{'fire': 0.16666666666666666,
 'walk': 0.0,
 'for': 0.0,
 'man': 0.0,
 'went': 0.0,
 'children': 0.16666666666666666,
 'a': 0.0,
 'sat': 0.16666666666666666,
 'the': 0.3333333333333333,
 'around': 0.16666666666666666,
 'out': 0.0}
```

[71]:
```
#Calculate the IDF values
```

[82]:
```
def computeIDF(documents):

    import math
    N=len(documents)

    idfDict=dict.fromkeys(documents[0].keys(),0)

    #calculating the df
    for document in documents:
        for word,val in document.items():
            if val>0:
                idfDict[word]+=1

    #calculating the idf
    for word,val in idfDict.items():

        idfDict[word]=math.log(N/float(val))
```

```
    return idfDict
```

[83]:
```
idfs=computeIDF([numofwordsA,numofwordsB])
idfs
```

[83]:
```
{'fire': 0.6931471805599453,
 'walk': 0.6931471805599453,
 'for': 0.6931471805599453,
 'man': 0.6931471805599453,
 'went': 0.6931471805599453,
 'children': 0.6931471805599453,
 'a': 0.6931471805599453,
 'sat': 0.6931471805599453,
 'the': 0.0,
 'around': 0.6931471805599453,
 'out': 0.6931471805599453}
```

[84]:
```
#calculatin the TfIdf values of each term
```

[87]:
```
def computeTFIDF(tfbagofwords,idfs):
    tfidf={}
    for word,val in tfbagofwords.items():
        tfidf[word]=val*idfs[word]
    return tfidf
```

[90]:
```
tfidfA=computeTFIDF(tfA,idfs)
tfidfB=computeTFIDF(tfB,idfs)
```

[91]:
```
tfidfA
```

[91]:
```
{'fire': 0.0,
 'walk': 0.09902102579427789,
 'for': 0.09902102579427789,
 'man': 0.09902102579427789,
 'went': 0.09902102579427789,
 'children': 0.0,
 'a': 0.09902102579427789,
 'sat': 0.0,
 'the': 0.0,
 'around': 0.0,
 'out': 0.09902102579427789}
```

[92]:
```
tfidfB
```

[92]:
```
{'fire': 0.11552453009332421,
 'walk': 0.0,
 'for': 0.0,
```

```
 'man': 0.0,
 'went': 0.0,
 'children': 0.11552453009332421,
 'a': 0.0,
 'sat': 0.11552453009332421,
 'the': 0.0,
 'around': 0.11552453009332421,
 'out': 0.0}
```

[97]:
```
df=pd.DataFrame([tfidfA,tfidfB])
df
#0 - doc1
#1 - doc2
```

[97]:

	fire	walk	for	man	went	children	a	\
0	0.000000	0.099021	0.099021	0.099021	0.099021	0.000000	0.099021	
1	0.115525	0.000000	0.000000	0.000000	0.000000	0.115525	0.000000	

	sat	the	around	out
0	0.000000	0.0	0.000000	0.099021
1	0.115525	0.0	0.115525	0.000000

[ ]:

[ ]: