7

April 15, 2024

[1]:
```
import pandas as pd
import sklearn as sk
import math
```

[2]:
```
import docx
```

[3]:
```
pwd
```

[3]:
```
'C:\\Users\\Tej'
```

[4]:
```
cd E:\
```

```
E:\
```

[5]:
```
pip install python-docx
```

```
Requirement already satisfied: python-docx in c:\users\tej\anaconda3\lib\site-
packages (0.8.11)Note: you may need to restart the kernel to use updated
packages.

Requirement already satisfied: lxml>=2.3.2 in c:\users\tej\anaconda3\lib\site-
packages (from python-docx) (4.5.2)
```

[7]:
```
document = docx.Document('Sample.docx')
```

[8]:
```
print(document.paragraphs[0].text)
```

```
Hello this is class of TE student Div1
```

[9]:
```
import nltk
nltk.download('punkt')
```

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

[9]:
```
True
```

[10]:
```
word = "It originated from the idea that there are readers who prefer learning␣
  ↪new skills from the comforts of their drawing rooms"
nltk_tokens = nltk.word_tokenize(word)
print(nltk_tokens)
```

```
['It', 'originated', 'from', 'the', 'idea', 'that', 'there', 'are', 'readers',
'who', 'prefer', 'learning', 'new', 'skills', 'from', 'the', 'comforts', 'of',
'their', 'drawing', 'rooms']
```

[11]:
```
word.split()
```

[11]:
```
['It',
 'originated',
 'from',
 'the',
 'idea',
 'that',
 'there',
 'are',
 'readers',
 'who',
 'prefer',
 'learning',
 'new',
 'skills',
 'from',
 'the',
 'comforts',
 'of',
 'their',
 'drawing',
 'rooms']
```

[12]:
```
from nltk import pos_tag
from nltk import RegexpParser
nltk.download('averaged_perceptron_tagger')
```

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

[12]:
```
True
```

[13]:
```
word1 = "Learn from IIT and make Life easy".split()
print("After Split:- ", word1)
```

```
After Split:-  ['Learn', 'from', 'IIT', 'and', 'make', 'Life', 'easy']
```

[14]:
```
token_tags = pos_tag(word1)
print("After Tokenization:- ", token_tags)
```

```
After Tokenization:-  [('Learn', 'NNP'), ('from', 'IN'), ('IIT', 'NNP'), ('and',
'CC'), ('make', 'VB'), ('Life', 'NNP'), ('easy', 'JJ')]
```

[15]:
```
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
```

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

[16]:
```
text = "Nick likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.
 ↪words()]

print(tokens_without_sw)
```

```
['Nick', 'likes', 'play', 'football', ',', 'however', 'fond', 'tennis', '.']
```

[17]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
```

[18]:
```
ps = PorterStemmer()

sentence = "Programmers program with programming languages"

words = word_tokenize(sentence)

for w in words:
    print(w, " : ", ps.stem(w))
```

```
Programmers  :  programm
program  :  program
with  :  with
programming  :  program
languages  :  languag
```

[20]:
```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
```

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

[20]: True

[21]:
```
word3 = WordNetLemmatizer()

print("rocks :", word3.lemmatize("rocks"))
print("corpora :", word3.lemmatize("corpora"))

print("better :", word3.lemmatize("better", pos ="a"))
```

```
rocks : rock
corpora : corpus
better : good
```

[32]:
```
import pandas as pd
import sklearn as sk
import math
```

[37]:
```
first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"

first_sentence = first_sentence.split(" ")

second_sentence = second_sentence.split(" ")

total= set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'Data', 'the', 'machine', 'for', 'science', 'key', 'sexiest', 'century', 'of',
'job', '21st', 'is', 'learning', 'data', 'Science'}
```

[38]:
```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)

for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1
```

[54]:
```
pd.DataFrame([wordDictA, wordDictB])
```

[54]:
```
   Data  the  machine  for  science  key  sexiest  century  of  job  21st  is  \
0     1    2        0    0        0    0        1        1   1    1     1   1
1     0    1        1    1        1    1        0        0   0    0     0   1

   learning  data  Science
0         0     0        1
```

```
1          1     1        0
```

[67]:
```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)

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

        tfFirst = computeTF(wordDictA, first_sentence)
        tfSecond = computeTF(wordDictB, second_sentence)

pd.DataFrame([tfFirst, tfSecond])
```

[67]:
```
   Data    the  machine    for  science    key  sexiest  century   of  job  \
0   0.1  0.200    0.000  0.000    0.000  0.000      0.1      0.1  0.1  0.1
1   0.0  0.125    0.125  0.125    0.125  0.125      0.0      0.0  0.0  0.0

   21st     is  learning   data  Science
0   0.1  0.100     0.000  0.000      0.1
1   0.0  0.125     0.125  0.125      0.0
```

[69]:
```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

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

    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)


idfs = computeIDF([wordDictA, wordDictB])
```

[78]:
```
def computeTFIDF(tfBow, idfs):
    tfidf = {}

    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)

    idfFirst = computeTFIDF(tfFirst, idfs)
    idfSecond = computeTFIDF(tfSecond, idfs)
    idf= pd.DataFrame([idfFirst, idfSecond])

print(idf)
```

```
       Data       the   machine       for   science       key   sexiest  \
0  0.030103  0.060206  0.000000  0.000000  0.000000  0.000000  0.030103
1  0.000000  0.037629  0.037629  0.037629  0.037629  0.037629  0.000000

    century        of       job      21st        is  learning      data  \
0  0.030103  0.030103  0.030103  0.030103  0.030103  0.000000  0.000000
1  0.000000  0.000000  0.000000  0.000000  0.037629  0.037629  0.037629

    Science
0  0.030103
1  0.000000
```

[ ]:

```

```