anurag-dsabda-pr7-2

April 15, 2024

[1]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\system21\anaconda3\lib\site-
packages (3.8.1)
Requirement already satisfied: click in c:\users\system21\anaconda3\lib\site-
packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\system21\anaconda3\lib\site-
packages (from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in
c:\users\system21\anaconda3\lib\site-packages (from nltk) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\system21\anaconda3\lib\site-
packages (from nltk) (4.65.0)
Requirement already satisfied: colorama in c:\users\system21\anaconda3\lib\site-
packages (from click->nltk) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import nltk as nltk
nltk.download("punkt")
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[2]:
```
True
```

[3]:
```python
text= "Tokenization is the first step in text analytics. The process of␣
↪breaking down a text paragraph into smaller chunks such as words or␣
↪sentences is called Tokenization."
```

[4]:
```python
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.', 'The process of breaking
down a text paragraph into smaller chunks such as words or sentences is called
Tokenization.']
```

[5]:
```python
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.',
'The', 'process', 'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into',
'smaller', 'chunks', 'such', 'as', 'words', 'or', 'sentences', 'is', 'called',
'Tokenization', '.']
```

[6]:
```python
import regex as re
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
text= "How to remove stop words with NLTK library in Python?"
text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
    if w not in stop_words:
        filtered_text.append(w)
print("Tokenized Sentence:",tokens)
print("Filterd Sentence:",filtered_text)
```

```
{'from', "won't", 'most', 'and', 'wasn', 'very', 'those', 'now', 'doing', 'won',
'ourselves', 'yourselves', 'm', "mustn't", 'into', 'through', "haven't",
'about', "you'll", 'will', 'in', 'isn', 'hers', 'had', 'when', "it's", 'be',
'where', 'than', "shan't", 'them', 'whom', 'of', 'because', 'other', 'out',
'our', 'all', 'below', 'under', "mightn't", 'up', 'nor', 'over', 'until', 't',
'these', 'needn', 'having', 'so', 'hasn', 'the', 'with', 'above', "didn't",
'both', 'doesn', 'shouldn', 'do', 'is', "isn't", 'his', "don't", 'myself',
"needn't", 'a', "hasn't", 'such', 'she', 'before', 'him', 'i', 'at', 'mustn',
's', 'been', 'himself', 'that', 'didn', 'there', 'you', 'wouldn', 'did', 'her',
'down', 'once', 'd', "you're", 'how', 'again', 'some', 'herself', 'are', 'just',
'have', 'ain', 'my', 'mightn', 'own', 'which', 'more', 'here', 'were', "she's",
'does', 'while', 'what', "hadn't", 'can', 'o', 'he', "you've", 'couldn', 'any',
'each', 'few', "you'd", 'their', 'theirs', 'has', 'they', 'am', 'haven', 'to',
'yourself', 'as', 'between', "weren't", 'during', 'was', "wasn't", 'being', 'y',
'why', 'hadn', "wouldn't", 'by', 'your', 'should', 'ma', 'it', 'off', 'after',
'yours', "aren't", 'we', 'then', 'this', 'me', 'an', 'but', 'weren', 'for',
'against', 're', 'ours', 'same', 'its', 'only', 'shan', "doesn't", 'or',
'themselves', 'don', "couldn't", "should've", 'aren', 'itself', "shouldn't",
'no', 'too', 'll', 'on', 've', 'further', 'if', 'who', "that'll", 'not'}
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk',
'library', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

[90]:
```
pip install regex
```

```
Requirement already satisfied: regex in c:\users\system21\anaconda3\lib\site-
packages (2023.10.3)
Note: you may need to restart the kernel to use updated packages.
```

[76]:
```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited","waits"]
ps =PorterStemmer()
for w in e_words:
   rootWord=ps.stem(w)
print(rootWord)
```

```
wait
```

[77]:
```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer =WordNetLemmatizer()
text = "studies studying cries cry"
tokenization =nltk.word_tokenize(text)
for w in tokenization:
   print("Lemma for {} is{}".format(w,
wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for studies isstudy
Lemma for studying isstudying
Lemma for cries iscry
Lemma for cry iscry
```

[78]:
```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
```

```
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

[112]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

[113]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

[114]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

[115]:
```
uniqueWords =set (bagOfWordsA).union(set(bagOfWordsB))
```

[116]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
    numOfWordsB = dict.fromkeys(uniqueWords,0)
    for word in bagOfWordsB:
        numOfWordsB[word] += 1
```

[122]:
```
def computeTF(wordDict, bagOfWords):
   tfDict = {}
   bagOfWordsCount =len(bagOfWords)
   for word, count in wordDict.items():
      tfDict[word] = count / float (bagOfWordsCount)
   return tfDict
tfA = computeTF(numOfWordsA,bagOfWordsA)
tfB =computeTF(numOfWordsB, bagOfWordsB)
tfA
tfB
```

[122]:
```
{'is': 0.125,
 'Planet': 0.0,
 'Jupiter': 0.0,
 'the': 0.25,
 'planet': 0.125,
 'Mars': 0.125,
 'fourth': 0.125,
 'Sun': 0.125,
 'from': 0.125,
 'largest': 0.0}
```

[180]:
```
def computeIDF(documents):
     import math
     N = len(documents)
     idfDict = dict.fromkeys(documents[0].keys(),0)
     for document in documents:
       for word, val in document.items():
          if val > 0 :
            idfDict[word] += 1
     for word, val in idfDict.items():
      idfDict[word] = math.log(N / float(val))
     return idfDict
idfs = computeIDF([numOfWordsA,numOfWordsB])
idfs
```

[180]:
```
{'is': 0.0,
 'Planet': 0.6931471805599453,
 'Jupiter': 0.6931471805599453,
 'the': 0.0,
 'planet': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'fourth': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'from': 0.6931471805599453,
 'largest': 0.6931471805599453}
```

[ ]:
```
Name-Anurag Jadhav
Roll No-13171
PR-7
```