exp7

February 27, 2025

[1]:
```
import nltk
```

[2]:
```
import re
```

[3]:
```
import pandas as pd
```

[4]:
```
import math
```

[5]:
```
from nltk.tokenize import sent_tokenize, word_tokenize
```

[6]:
```
from nltk.corpus import stopwords
```

[7]:
```
from nltk.stem import PorterStemmer, WordNetLemmatizer
```

[8]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

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

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

[9]:
```
True
```

[10]:
```
nltk.download('stopwords')
```

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

[10]:
```
True
```

[11]:
```
nltk.download('wordnet')
```

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

[11]: True

[12]:
```
nltk.download('averaged_perceptron_tagger')
```

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

[12]: True

[13]:
```
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."
```

[14]:
```
tokenized_text = sent_tokenize(text)
```

[15]:
```
tokenized_word = word_tokenize(text)
```

[16]:
```
print("Word Tokenization:", tokenized_word)
```

```
Word Tokenization: ['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', '.']
```

[17]:
```
stop_words = set(stopwords.words("english"))
```

[18]:
```
filtered_text = [w for w in tokenized_word if w.lower() not in stop_words and w.
↪isalpha()]
```

[19]:
```
print("Filtered Text (after stop word removal):", filtered_text)
```

```
Filtered Text (after stop word removal): ['Tokenization', 'first', 'step',
'text', 'analytics', 'process', 'breaking', 'text', 'paragraph', 'smaller',
'chunks', 'words', 'sentences', 'called', 'Tokenization']
```

[20]:
```
ps = PorterStemmer()
```

[21]:
```
e_words = ["wait", "waiting", "waited", "waits"]
```

[22]:
```
print("\nStemming:")
```

```
Stemming:
```

[24]:
```
for w in e_words:
    rootWord = ps.stem(w)
    print(f"Original: {w}, Stemmed: {rootWord}")
```

```
Original: wait, Stemmed: wait
Original: waiting, Stemmed: wait
Original: waited, Stemmed: wait
Original: waits, Stemmed: wait
```

[25]:
```
wordnet_lemmatizer = WordNetLemmatizer()
```

[26]:
```
print("\nLemmatization:")
```

```
Lemmatization:
```

[27]:
```
tokenization = nltk.word_tokenize("studies studying cries cry")
```

[28]:
```
for w in tokenization:
    print(f"Lemma for {w} is {wordnet_lemmatizer.lemmatize(w)}")
```

```
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

[29]:
```
data = "The pink sweater fit her perfectly"
```

[30]:
```
words = word_tokenize(data)
```

[31]:
```
print("\nPOS Tagging:")
```

```
POS Tagging:
```

[32]:
```
for word in words:
    print(nltk.pos_tag([word]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

[33]:
```
documentA = 'Jupiter is the largest Planet'
```

[34]:
```
documentB = 'Mars is the fourth planet from the Sun'
```

[35]:
```
bagOfWordsA = documentA.split(' ')
```

[36]:
```
bagOfWordsB = documentB.split(' ')
```

[37]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

[38]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
```

[39]:
```
for word in bagOfWordsA:
    numOfWordsA[word] += 1
```

[40]:
```
numOfWordsB = dict.fromkeys(uniqueWords, 0)
```

[41]:
```
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

[42]:
```
def computeTF(wordDict, bagOfWords):
```

```
  Cell In[42], line 1
    def computeTF(wordDict, bagOfWords):
                                        ^
SyntaxError: incomplete input
```

[43]:
```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
```

[44]:
```
tfA = computeTF(numOfWordsA, bagOfWordsA)
```

[45]:
```
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

[46]:
```
def computeIDF(documents):
    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
```

[47]:
```
idfs = computeIDF([numOfWordsA, numOfWordsB])
```

[48]:
```
print("\nInverse Document Frequency (IDF):", idfs)
```

```
Inverse Document Frequency (IDF): {'Sun': 0.6931471805599453, 'Planet':
0.6931471805599453, 'largest': 0.6931471805599453, 'Mars': 0.6931471805599453,
'from': 0.6931471805599453, 'the': 0.0, 'fourth': 0.6931471805599453, 'planet':
0.6931471805599453, 'is': 0.0, 'Jupiter': 0.6931471805599453}
```

[49]:
```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
```

[50]:
```
tfidfA = computeTFIDF(tfA, idfs)
```

[51]:
```
tfidfB = computeTFIDF(tfB, idfs)
```

[52]:
```
df = pd.DataFrame([tfidfA, tfidfB])
```

[53]:
```
print("\nTF-IDF Representation of Document A and B:")
```

```
TF-IDF Representation of Document A and B:
```

[54]:
```
print(df)
```

```
        Sun    Planet   largest      Mars      from  the    fourth    planet  \
0  0.000000  0.138629  0.138629  0.000000  0.000000  0.0  0.000000  0.000000
1  0.086643  0.000000  0.000000  0.086643  0.086643  0.0  0.086643  0.086643

    is   Jupiter
0  0.0  0.138629
1  0.0  0.000000
```

[55]:
```
sample_text = "How to remove stop words with NLTK library in Python?"
```

[56]:
```
sample_text = re.sub('[^a-zA-Z]', ' ', sample_text)
```

[57]:
```
tokens = word_tokenize(sample_text.lower())
```

[58]:
```
filtered_text = [w for w in tokens if w not in stop_words]
```

[59]:
```
print("\nStop Words Removal:")
```

Stop Words Removal:

[60]:
```
print("Tokenized Sentence:", tokens)
```

```
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk',
'library', 'in', 'python']
```

[61]:
```
print("Filtered Sentence:", filtered_text)
```

```
Filtered Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

[ ]:
```

```