practical-7

May 9, 2025

[28]:
```
import ssl
import nltk
```

[30]:
```
import ssl
import nltk
nltk.download('punkt_tab')
```

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

[30]:
```
True
```

[31]:
```
# Disable SSL certificate verification
ssl._create_default_https_context = ssl._create_unverified_context
```

[32]:
```
# Now try downloading NLTK datasets again
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data…
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger_eng.zip.
```

[32]:
```
True
```

[33]:
```
text= """Tokenization is the first step in text analytics. The process
of breaking down a text such as words or sentences is called
Tokenization."""
```

[34]:
```
#Sentence Tokenization
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[34], line 3
      1 #Sentence Tokenization
      2 from nltk.tokenize import sent_tokenize
----> 3 tokenized_text= sent_tokenize(text)
      4 print(tokenized_text)

File␣
 ↪~\AppData\Local\Programs\Python\Python313\Lib\site-packages\nltk\tokenize\__init__.
 ↪py:119, in sent_tokenize(text, language)
    109 def sent_tokenize(text, language="english"):
    110     """
    111     Return a sentence-tokenized copy of *text*,
    112     using NLTK's recommended sentence tokenizer
   (…)
    117     :param language: the model name in the Punkt corpus
    118     """
--> 119     tokenizer = _get_punkt_tokenizer(language)
    120     return tokenizer.tokenize(text)

File␣
 ↪~\AppData\Local\Programs\Python\Python313\Lib\site-packages\nltk\tokenize\__init__.
 ↪py:105, in _get_punkt_tokenizer(language)
     96 @functools.lru_cache
     97 def _get_punkt_tokenizer(language="english"):
     98     """
     99     A constructor for the PunktTokenizer that utilizes
    100     a lru cache for performance.
   (…)
    103     :type language: str
    104     """
--> 105     return PunktTokenizer(language)

File␣
 ↪~\AppData\Local\Programs\Python\Python313\Lib\site-packages\nltk\tokenize\punkt.
 ↪py:1744, in PunktTokenizer.__init__(self, lang)
   1742 def __init__(self, lang="english"):
   1743     PunktSentenceTokenizer.__init__(self)
-> 1744     self.load_lang(lang)
```

```
File␣
  ↪~\AppData\Local\Programs\Python\Python313\Lib\site-packages\nltk\tokenize\punkt.
  ↪py:1750, in PunktTokenizer.load_lang(self, lang)
   1747 from nltk.data import find
   1749 lang_dir = find(f"tokenizers/punkt_tab/{lang}/")
-> 1750 self._params = load_punkt_params(lang_dir)
   1751 self._lang = lang

File␣
  ↪~\AppData\Local\Programs\Python\Python313\Lib\site-packages\nltk\tokenize\punkt.
  ↪py:1758, in load_punkt_params(lang_dir)
   1757 def load_punkt_params(lang_dir):
-> 1758     from nltk.tabdata import PunktDecoder
   1760     pdec = PunktDecoder()
   1761     # Make a new Parameters object:

ModuleNotFoundError: No module named 'nltk.tabdata'
```

[35]:
```
pip install --upgrade nltk
```

```
Collecting nltk
  Using cached nltk-3.9.1-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: click in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
nltk) (8.1.8)
Requirement already satisfied: joblib in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
nltk) (1.5.0)
Requirement already satisfied: regex>=2021.8.3 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
nltk) (2024.11.6)
Requirement already satisfied: tqdm in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
nltk) (4.67.1)
Requirement already satisfied: colorama in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
click->nltk) (0.4.6)
Using cached nltk-3.9.1-py3-none-any.whl (1.5 MB)
Installing collected packages: nltk
Successfully installed nltk-3.9.1
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 24.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[36]:
```
import nltk
nltk.download('punkt')
```

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

[36]:
```
True
```

[37]:
```
from nltk.tokenize import sent_tokenize

text = """Tokenization is the first step in text analytics. The process of␣
 ↪breaking down a textsuch as words or sentences is called Tokenization."""

tokenized_text = sent_tokenize(text)
print(tokenized_text)
```

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

[40]:
```
#Word Tokenization
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', 'textsuch', 'as', 'words',
'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

[41]:
```
# print stop words of English
import 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)
```

```
{'nor', 'she', 'as', 'didn', 'are', 'below', 'over', "wasn't", 'up', 'herself',
'do', 'between', "she'd", 'too', 'both', 'was', "we'd", 'now', "i've", 'only',
'to', 'before', 'theirs', 'it', 'doing', 'shan', 'by', 'and', 'were', "mustn't",
'is', 'couldn', 'hadn', 'a', 'the', 'just', 'have', 'hasn', "we'll", 'any', 'o',
"aren't", 'if', 'mightn', 'been', 'myself', 'for', 'their', 'very', "we're",
'had', 'whom', 'off', 'hers', 'on', 'where', 't', 'y', 'yourself', "it's",
'because', 'during', 'into', 'same', 'some', 'with', 'his', 'against', 'all',
```

```
'itself', 'not', 'once', 'did', "hasn't", 'll', 'himself', "you'll", "that'll",
"you'd", 'through', "she'll", "he'd", 'wasn', 'yourselves', 'them', "i'll",
'under', "i'd", 'haven', "she's", 'down', "mightn't", 'why', 'am', "won't",
'no', "haven't", 'those', 'in', 'my', 'than', 'further', 'm', 'having', 'these',
'who', 'of', 'ours', 'he', 'until', 'don', 'what', 'after', 'they', 'so',
'wouldn', 'such', "should've", 'that', 'isn', 'most', 've', "it'd", "i'm",
"it'll", 'which', "wouldn't", 'at', "hadn't", 'being', 'needn', "don't", 'here',
"doesn't", "couldn't", 'then', 'will', "you've", 'be', 'doesn', 'from', "he's",
"he'll", 'yours', 'each', 'd', "they'll", "isn't", 'can', 'aren', 're', "we've",
'themselves', 'you', 'our', 'ourselves', "they've", 'again', 'mustn', "shan't",
'him', 's', "they're", "weren't", "you're", 'how', 'above', 'few', 'i',
"they'd", 'shouldn', "shouldn't", 'weren', 'we', 'your', 'this', 'an', 'when',
'ma', 'own', 'ain', 'should', 'more', 'its', 'out', 'there', 'me', 'won',
"needn't", 'about', "didn't", 'but', 'other', 'her', 'has', 'while', 'does',
'or'}
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk',
'library', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

[45]:
```python
#Perform stemming
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
```

[46]:
```python
#Perform lemmitization
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 is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

[47]:
```python
#Perform POS Tagging
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')]
```

[48]:
```
# Create representation of document by calculating TFIDF
```

[49]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

[50]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

[51]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

[54]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

[55]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
    numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

[56]:
```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
```

[57]:
```
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

[58]:
```
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
```

[59]:
```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
```

[60]:
```
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

[60]:
```
{'Sun': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'the': 0.0,
 'is': 0.0,
 'from': 0.6931471805599453,
 'fourth': 0.6931471805599453,
 'Jupiter': 0.6931471805599453,
 'largest': 0.6931471805599453}
```

[61]:
```
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
```

[62]:
```
df = pd.DataFrame([tfidfA, tfidfB])
df
```

[62]:

	Sun	Mars	Planet	planet	the	is	from	fourth	Jupiter	largest
0	0.000000	0.000000	0.138629	0.000000	0.0	0.0	0.000000	0.000000	0.138629	0.138629
1	0.086643	0.086643	0.000000	0.086643	0.0	0.0	0.086643	0.086643	0.000000	0.000000

[ ]:
```

```