lab-07

April 13, 2024

[23]:
```
import nltk
from nltk import word_tokenize, sent_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

[45]:
```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
```

[45]:
```
True
```

Tokenization example

[25]:
```
text = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce commodo␣
↪mauris id justo condimentum dignissim. Nullam placerat semper dapibus.␣
↪Pellentesque ac risus nulla. Phasellus ut dapibus nunc, id aliquam dolor."
```

[26]:
```
print(word_tokenize(text))
```

```
['Lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipiscing',
'elit', '.', 'Fusce', 'commodo', 'mauris', 'id', 'justo', 'condimentum',
'dignissim', '.', 'Nullam', 'placerat', 'semper', 'dapibus', '.',
'Pellentesque', 'ac', 'risus', 'nulla', '.', 'Phasellus', 'ut', 'dapibus',
'nunc', ',', 'id', 'aliquam', 'dolor', '.']
```

[27]:
```
print(sent_tokenize(text))
```

```
['Lorem ipsum dolor sit amet, consectetur adipiscing elit.', 'Fusce commodo
mauris id justo condimentum dignissim.', 'Nullam placerat semper dapibus.',
'Pellentesque ac risus nulla.', 'Phasellus ut dapibus nunc, id aliquam dolor.']
```

POS Tagging

[28]:
```
to_tag = word_tokenize(text)
```

[29]:
```
print(pos_tag(to_tag))
```

```
[('Lorem', 'NNP'), ('ipsum', 'NN'), ('dolor', 'NN'), ('sit', 'NN'), ('amet',
'NN'), (',', ','), ('consectetur', 'NN'), ('adipiscing', 'VBG'), ('elit', 'NN'),
('.', '.'), ('Fusce', 'NNP'), ('commodo', 'JJ'), ('mauris', 'NN'), ('id', 'NN'),
('justo', 'NN'), ('condimentum', 'NN'), ('dignissim', 'NN'), ('.', '.'),
('Nullam', 'NNP'), ('placerat', 'VBZ'), ('semper', 'JJR'), ('dapibus', 'NN'),
('.', '.'), ('Pellentesque', 'NNP'), ('ac', 'JJ'), ('risus', 'NN'), ('nulla',
'NN'), ('.', '.'), ('Phasellus', 'CC'), ('ut', 'JJ'), ('dapibus', 'NN'),
('nunc', 'NN'), (',', ','), ('id', 'JJ'), ('aliquam', 'NN'), ('dolor', 'NN'),
('.', '.')]
```

Stopwords

[30]:
```
stop_words = set(stopwords.words("english"))
print(stop_words)
```

```
{'should', "didn't", 'by', 'because', 'whom', 'those', 'theirs', 'up', 'its',
'my', 's', 'what', "couldn't", "you've", 'not', 'during', 'herself', 'ma',
'where', 'on', "you'd", 'does', 'doesn', 'am', 'under', 'do', 'our', 'itself',
'if', 'isn', 'just', 'further', 'between', 'very', 'y', 'once', 'can', 'until',
'here', "it's", "wasn't", 'm', 'out', 'having', "you're", 'than', 'him', 'down',
'he', 'ourselves', 'above', 'into', 'it', 'through', 'before', 'been', "aren't",
"needn't", 'nor', 'yourselves', 'about', 'some', 'will', "weren't", 'more',
'most', 'but', 'then', 'too', 'needn', 'did', 'so', 'no', 'ain', 'below',
"doesn't", 'how', "isn't", 'are', 'any', 'yourself', 'or', "mightn't", 'we',
"you'll", 'me', 'you', 'own', 'each', 'such', 'being', 'd', 'wasn', 'them', 't',
'haven', 've', 'in', 'a', 'won', 'were', 'as', 'off', 'at', "hadn't", "she's",
'your', 'be', 'other', 'o', 'hadn', 'while', 'for', 'they', 'shouldn', 'yours',
'now', "hasn't", "don't", 'both', 're', 'which', 'from', 'is', 'ours', 'myself',
'after', "haven't", 'this', 'doing', 'hasn', 'same', 'weren', 'has', 'and',
```

```
'don', 'wouldn', 'himself', 'have', 'll', 'shan', 'their', 'of', 'again',
'aren', 'with', 'couldn', 'why', 'i', 'an', 'to', 'mightn', "that'll", 'the',
"mustn't", "shan't", "wouldn't", 'she', 'who', 'there', 'only', 'was', 'his',
'all', "should've", "shouldn't", 'against', "won't", 'few', 'didn', 'mustn',
'hers', 'themselves', 'these', 'had', 'her', 'over', 'that', 'when'}
```

[31]:
```
to_clean = word_tokenize(text)
to_clean
```

[31]:
```
['Lorem',
 'ipsum',
 'dolor',
 'sit',
 'amet',
 ',',
 'consectetur',
 'adipiscing',
 'elit',
 '.',
 'Fusce',
 'commodo',
 'mauris',
 'id',
 'justo',
 'condimentum',
 'dignissim',
 '.',
 'Nullam',
 'placerat',
 'semper',
 'dapibus',
 '.',
 'Pellentesque',
 'ac',
 'risus',
 'nulla',
 '.',
 'Phasellus',
 'ut',
 'dapibus',
 'nunc',
 ',',
 'id',
 'aliquam',
 'dolor',
 '.']
```

[32]:
```
no_stopwords_text = []
for token in to_clean:
    if(token not in stop_words):
        no_stopwords_text.append(token)

print(no_stopwords_text)
```

```
['Lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipiscing',
'elit', '.', 'Fusce', 'commodo', 'mauris', 'id', 'justo', 'condimentum',
'dignissim', '.', 'Nullam', 'placerat', 'semper', 'dapibus', '.',
'Pellentesque', 'ac', 'risus', 'nulla', '.', 'Phasellus', 'ut', 'dapibus',
'nunc', ',', 'id', 'aliquam', 'dolor', '.']
```

Stemming

[33]:
```
stemmer = PorterStemmer()
```

[34]:
```
stemmed_words = []
for token in no_stopwords_text:
    stemmed_word = stemmer.stem(token)
    stemmed_words.append(stemmed_word)
```

[35]:
```
print(stemmed_words)
```

```
['lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipisc',
'elit', '.', 'fusc', 'commodo', 'mauri', 'id', 'justo', 'condimentum',
'dignissim', '.', 'nullam', 'placerat', 'semper', 'dapibu', '.', 'pellentesqu',
'ac', 'risu', 'nulla', '.', 'phasellu', 'ut', 'dapibu', 'nunc', ',', 'id',
'aliquam', 'dolor', '.']
```

Lemmatization

[36]:
```
lemmatizer = WordNetLemmatizer()
```

[37]:
```
lemmatized_words = []
for token in no_stopwords_text:
    lemmatized = lemmatizer.lemmatize(token)  # Assuming you want to lemmatize␣
 ↪verbs (you can change the 'pos' argument as needed)
    lemmatized_words.append(lemmatized)
```

[38]:
```
print(lemmatized_words)
```

```
['Lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipiscing',
'elit', '.', 'Fusce', 'commodo', 'mauris', 'id', 'justo', 'condimentum',
'dignissim', '.', 'Nullam', 'placerat', 'semper', 'dapibus', '.',
'Pellentesque', 'ac', 'risus', 'nulla', '.', 'Phasellus', 'ut', 'dapibus',
'nunc', ',', 'id', 'aliquam', 'dolor', '.']
```

TF-IDF Vectorization

[39]:
```
vectorizer = TfidfVectorizer()
```

[40]:
```
corpus = [
    "I love to eat pizza",
    "Pizza is my favorite food",
    "I enjoy eating pizza with friends",
    "I like to have pizza for dinner",
    "Pizza toppings include cheese, pepperoni, and mushrooms"
]
```

[41]:
```
vectorizer = TfidfVectorizer()
vectorizer
```

[41]:
```
TfidfVectorizer()
```

[42]:
```
tfidf_matrix = vectorizer.fit_transform(corpus)

feature_names = vectorizer.get_feature_names_out()
```

[43]:
```
print(tfidf_matrix.toarray())

print(feature_names)
```

```
[[0.         0.         0.         0.58946308 0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.58946308 0.         0.         0.
  0.28088232 0.4755751  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.48638585 0.48638585 0.         0.         0.         0.
  0.48638585 0.         0.         0.         0.48638585 0.
  0.23176546 0.         0.         0.        ]
 [0.         0.         0.         0.         0.48638585 0.48638585
  0.         0.         0.         0.48638585 0.         0.
  0.         0.         0.         0.         0.         0.
  0.23176546 0.         0.         0.48638585]
 [0.         0.         0.45277275 0.         0.         0.
  0.         0.         0.45277275 0.         0.45277275 0.
  0.         0.45277275 0.         0.         0.         0.
  0.21574864 0.36529421 0.         0.        ]
 [0.40073619 0.40073619 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.40073619
  0.         0.         0.         0.40073619 0.         0.40073619
  0.19095294 0.         0.40073619 0.        ]]
['and' 'cheese' 'dinner' 'eat' 'eating' 'enjoy' 'favorite' 'food' 'for'
 'friends' 'have' 'include' 'is' 'like' 'love' 'mushrooms' 'my'
 'pepperoni' 'pizza' 'to' 'toppings' 'with']
```