exp-7

April 26, 2024

[ ]:
```
#exp_7
#Name:Mahesh Jagtap
#Roll No: A-28
```

[1]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/kj-comp/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[1]:
```
True
```

[3]:
```
from nltk import word_tokenize, sent_tokenize
sent = "Sachin is considered to be one of the greatest cricket players. Virat␣
  ↪is the captain of the Indian cricket team"
print(word_tokenize(sent))
print(sent_tokenize(sent))
```

```
['Sachin', 'is', 'considered', 'to', 'be', 'one', 'of', 'the', 'greatest',
'cricket', 'players', '.', 'Virat', 'is', 'the', 'captain', 'of', 'the',
'Indian', 'cricket', 'team']
['Sachin is considered to be one of the greatest cricket players.', 'Virat is
the captain of the Indian cricket team']
```

[4]:
```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stop_words = stopwords.words('english')
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to /home/kj-
[nltk_data]     comp/nltk_data…
```

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

```
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[6]:
```python
token = word_tokenize(sent)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
print("This is the unclean version : ",token)
print("This is the cleaned version : ",cleaned_token)
```

```
This is the unclean version :  ['Sachin', 'is', 'considered', 'to', 'be', 'one',
'of', 'the', 'greatest', 'cricket', 'players', '.', 'Virat', 'is', 'the',
'captain', 'of', 'the', 'Indian', 'cricket', 'team']
This is the cleaned version :  ['Sachin', 'considered', 'one', 'greatest',
'cricket', 'players', '.', 'Virat', 'captain', 'Indian', 'cricket', 'team']
```

[7]:
```python
words = [cleaned_token.lower() for cleaned_token in cleaned_token if␣
 ↪cleaned_token.isalpha()]
```

[8]:
```python
print(words)
```

```
['sachin', 'considered', 'one', 'greatest', 'cricket', 'players', 'virat',
'captain', 'indian', 'cricket', 'team']
```

[9]:
```python
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
port_stemmer_output = [stemmer.stem(words) for words in words]
print(port_stemmer_output)
```

```
['sachin', 'consid', 'one', 'greatest', 'cricket', 'player', 'virat', 'captain',
'indian', 'cricket', 'team']
```

[30]:
```
import nltk
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
lemmatizer_output = [lemmatizer.lemmatize(words) for words in words]
print(lemmatizer_output)
```

```
[nltk_data] Downloading package omw-1.4 to /home/kj-comp/nltk_data…
[nltk_data] Downloading package wordnet to /home/kj-comp/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

```
['sachin', 'considered', 'one', 'greatest', 'cricket', 'player', 'virat',
'captain', 'indian', 'cricket', 'team']
```

[18]:
```
from nltk import pos_tag
import nltk
nltk.download('averaged_perceptron_tagger')
token = word_tokenize(sent)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
tagged = pos_tag(cleaned_token)
print(tagged)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/kj-comp/nltk_data…
```

```
[('Sachin', 'NNP'), ('considered', 'VBD'), ('one', 'CD'), ('greatest', 'JJS'),
('cricket', 'NN'), ('players', 'NNS'), ('.', '.'), ('Virat', 'NNP'), ('captain',
'NN'), ('Indian', 'JJ'), ('cricket', 'NN'), ('team', 'NN')]
```

```
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

[19]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
```

[20]:
```
docs = [
"Sachin is considered to be one of the greatest cricket players",
"Federer is considered one of the greatest tennis players",
"Nadal is considered one of the greatest tennis players",
"Virat is the captain of the Indian cricket team"]
```

[23]:
```
vectorizer = TfidfVectorizer(analyzer = "word", norm = None , use_idf = True ,␣
 ↪smooth_idf=True)
Mat = vectorizer.fit(docs)
print(Mat.vocabulary_)
```

```
{'sachin': 12, 'is': 7, 'considered': 2, 'to': 16, 'be': 0, 'one': 10, 'of': 9,
'the': 15, 'greatest': 5, 'cricket': 3, 'players': 11, 'federer': 4, 'tennis':
14, 'nadal': 8, 'virat': 17, 'captain': 1, 'indian': 6, 'team': 13}
```

[24]:
```
tfidfMat = vectorizer.fit_transform(docs)
```

[25]:
```
print(tfidfMat)
```

```
  (0, 11)	1.2231435513142097
  (0, 3)	1.5108256237659907
  (0, 5)	1.2231435513142097
  (0, 15)	1.0
  (0, 9)	1.0
  (0, 10)	1.2231435513142097
  (0, 0)	1.916290731874155
  (0, 16)	1.916290731874155
  (0, 2)	1.2231435513142097
  (0, 7)	1.0
  (0, 12)	1.916290731874155
  (1, 14)	1.5108256237659907
  (1, 4)	1.916290731874155
  (1, 11)	1.2231435513142097
  (1, 5)	1.2231435513142097
  (1, 15)	1.0
  (1, 9)	1.0
  (1, 10)	1.2231435513142097
  (1, 2)	1.2231435513142097
  (1, 7)	1.0
  (2, 8)	1.916290731874155
  (2, 14)	1.5108256237659907
  (2, 11)	1.2231435513142097
  (2, 5)	1.2231435513142097
  (2, 15)	1.0
  (2, 9)	1.0
  (2, 10)	1.2231435513142097
  (2, 2)	1.2231435513142097
  (2, 7)	1.0
  (3, 13)	1.916290731874155
  (3, 6)	1.916290731874155
  (3, 1)	1.916290731874155
  (3, 17)	1.916290731874155
  (3, 3)	1.5108256237659907
  (3, 15)	2.0
  (3, 9)	1.0
  (3, 7)	1.0
```

[26]:
```
features_names = vectorizer.get_feature_names_out()
print(features_names)
```

```
['be' 'captain' 'considered' 'cricket' 'federer' 'greatest' 'indian' 'is'
 'nadal' 'of' 'one' 'players' 'sachin' 'team' 'tennis' 'the' 'to' 'virat']
```

[27]:
```
dense = tfidfMat.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist , columns = features_names)
```

[28]:
```
df
```

[28]:
```
         be   captain  considered   cricket   federer  greatest    indian  \
0  1.916291  0.000000    1.223144  1.510826  0.000000  1.223144  0.000000
1  0.000000  0.000000    1.223144  0.000000  1.916291  1.223144  0.000000
2  0.000000  0.000000    1.223144  0.000000  0.000000  1.223144  0.000000
3  0.000000  1.916291    0.000000  1.510826  0.000000  0.000000  1.916291

    is     nadal   of       one   players    sachin      team    tennis  the  \
0  1.0  0.000000  1.0  1.223144  1.223144  1.916291  0.000000  0.000000  1.0
1  1.0  0.000000  1.0  1.223144  1.223144  0.000000  0.000000  1.510826  1.0
2  1.0  1.916291  1.0  1.223144  1.223144  0.000000  0.000000  1.510826  1.0
3  1.0  0.000000  1.0  0.000000  0.000000  0.000000  1.916291  0.000000  2.0

         to     virat
0  1.916291  0.000000
1  0.000000  0.000000
2  0.000000  0.000000
3  0.000000  1.916291
```

[29]:
```
features_names = sorted(vectorizer.get_feature_names())
```

```
/home/kj-comp/anaconda3/lib/python3.9/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function
get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will
be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

[ ]: