NLP_Demo (1)

August 19, 2022

```
[ ]: #!pip install nltk  -----------NLP Package
```

1 Anacondo command prompt

import nltk

nltk.download()

```
[ ]: # # To check whether nltk is installed or not
     # from nltk.corpus import brown
     # brown.words()
     # brown.categories()
```

2 NLTK demo

```
[1]: import pandas as pd
     data=pd.read_csv("User_reviews (1).csv")
     data.head(5)
```

```
[1]:                                               Review  Sentiment
     0                              Wow… Loved this place.        1.0
     1   I learned that if an electric slicer is used t…        NaN
     2                        But they don't clean the chiles?        NaN
     3                                   Crust is not good.        0.0
     4           Not tasty and the texture was just nasty.        0.0
```

```
[2]: data.shape
```

```
[2]: (3729, 2)
```

```
[3]: usernew=data[0:3]
     usernew
```

```
[3]:                                               Review  Sentiment
     0                            Wow… Loved this place.        1.0
     1  I learned that if an electric slicer is used t…        NaN
     2                  But they don't clean the chiles?        NaN
```

```
[15]: # Tokenization

      from nltk.tokenize import sent_tokenize,word_tokenize


      example_text=usernew["Review"][1]

      print(example_text)
```

```
I learned that if an electric slicer is used the blade becomes hot enough to
start to cook the prosciutto.
```

```
[16]: # Sentence Tokentize   # full stop for news sentence identifier

      sent_tokens=sent_tokenize(example_text)
      print(sent_tokens)
```

```
['I learned that if an electric slicer is used the blade becomes hot enough to
start to cook the prosciutto.']
```

```
[17]: # word Tokentizer

      word_tokens=word_tokenize(example_text)
      print(word_tokens)
```

```
['I', 'learned', 'that', 'if', 'an', 'electric', 'slicer', 'is', 'used', 'the',
'blade', 'becomes', 'hot', 'enough', 'to', 'start', 'to', 'cook', 'the',
'prosciutto', '.']
```

3 stopwords

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

      stop_words=set(stopwords.words("english"))

      print(stop_words)

      print(len(stop_words))
```

```
{'couldn', 'each', 'an', "you've", 'into', 'hasn', "it's", 'me', 'he', 'under',
'my', 'o', 've', 'too', 'only', 'does', 'have', 'theirs', 'having', 'between',
'y', 'of', 'i', 'own', 'so', 'can', 'further', 'our', "hadn't", 'll', 'd',
'not', 'you', "won't", 'they', 'but', 'same', 'before', 'won', 'out', 'him',
'over', 'as', 'shouldn', 'isn', 'himself', 'down', 'hers', 'ours', 'how', 'any',
'up', 'nor', 'were', 'didn', "needn't", "aren't", 'mustn', 'when', 'has',
"she's", 'no', "wasn't", "should've", 'some', 'm', 'ourselves', 'myself',
'again', 'is', 'here', 'had', 'are', 'both', 's', 'in', 'do', "mustn't", 'am',
'mightn', 'she', 'there', 'don', "haven't", 'which', "shouldn't", 'while', 'if',
'other', 'hadn', 'being', 'below', "doesn't", 'yourselves', 'a', 'we',
'themselves', 'and', 'these', 'until', "wouldn't", 'all', 'should', 'whom',
'wouldn', 'weren', 're', 'was', 'after', 'by', 'who', 'herself', 'most',
"you'd", 'them', "hasn't", 'shan', 'just', 'yours', 'be', "you're", 'ma',
'been', "isn't", 'wasn', "couldn't", "mightn't", 'needn', 'this', 'to', 'with',
't', 'during', 'doing', 'very', 'above', 'the', 'her', 'his', "you'll", 'what',
'ain', 'off', "didn't", 'yourself', 'because', 'or', 'then', 'about', "weren't",
'at', 'through', 'where', 'aren', 'against', 'will', 'that', "shan't", 'on',
'itself', 'your', 'such', "that'll", 'than', 'now', 'for', 'once', 'it',
'haven', 'why', "don't", 'its', 'those', 'from', 'did', 'few', 'doesn', 'more',
'their'}
179
```

[19]:
```
filtered_sentence=[]
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print(filtered_sentence)
```

```
['I', 'learned', 'electric', 'slicer', 'used', 'blade', 'becomes', 'hot',
'enough', 'start', 'cook', 'prosciutto', '.']
```

[20]:
```
stop_words.update([".","...","?","{","}","(",")"]) #update the stop words
print(len(stop_words))
print(stop_words)
```

```
186
{'couldn', 'each', 'an', "you've", 'into', 'hasn', "it's", 'me', 'he', 'under',
'my', 'o', 've', '}', 'too', 'only', 'does', 'have', 'theirs', 'having',
'between', '{', 'y', 'of', 'i', 'own', 'so', 'can', 'further', 'our', "hadn't",
'll', 'd', 'not', 'you', "won't", 'they', 'but', 'same', 'before', 'won', 'out',
'him', 'over', 'as', 'shouldn', 'isn', 'himself', 'down', 'hers', 'ours', 'how',
'any', 'up', 'nor', '(', 'were', 'didn', "needn't", "aren't", 'mustn', 'when',
'has', "she's", 'no', "wasn't", "should've", 'some', 'm', 'ourselves', 'myself',
'again', 'is', 'here', 'had', '?', 'are', 'both', 's', 'in', 'do', "mustn't",
'am', 'mightn', 'she', 'there', 'don', "haven't", 'which', "shouldn't", 'while',
'if', 'other', 'hadn', 'being', 'below', "doesn't", 'yourselves', 'a', 'we',
'themselves', 'and', 'these', 'until', "wouldn't", 'all', 'should', 'whom',
'wouldn', 'weren', 're', 'was', 'after', 'by', 'who', 'herself', 'most',
"you'd", 'them', "hasn't", 'shan', 'just', 'yours', 'be', "you're", 'ma',
'been', "isn't", 'wasn', "couldn't", "mightn't", '…', 'needn', 'this', 'to',
'with', 't', 'during', 'doing', 'very', 'above', 'the', 'her', 'his', "you'll",
'what', 'ain', 'off', "didn't", 'yourself', 'because', 'or', 'then', 'about',
"weren't", 'at', 'through', 'where', 'aren', 'against', 'will', 'that',
"shan't", 'on', 'itself', 'your', 'such', '.', "that'll", 'than', 'now', 'for',
'once', 'it', 'haven', 'why', "don't", 'its', 'those', 'from', 'did', 'few',
'doesn', ')', 'more', 'their'}
```

[21]:
```python
filtered_sentence=[]
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print(filtered_sentence)
```

```
['I', 'learned', 'electric', 'slicer', 'used', 'blade', 'becomes', 'hot',
'enough', 'start', 'cook', 'prosciutto']
```

4 stemming

[22]:
```python
from nltk.stem import PorterStemmer

stemmer=PorterStemmer()

stem_token=[stemmer.stem(word) for word in word_tokens]

print(stem_token)
```

```
['I', 'learn', 'that', 'if', 'an', 'electr', 'slicer', 'is', 'use', 'the',
'blade', 'becom', 'hot', 'enough', 'to', 'start', 'to', 'cook', 'the',
'prosciutto', '.']
```

[23]:
```python
# Lemmatization
from nltk.stem import WordNetLemmatizer

lemmatizer=WordNetLemmatizer()

lemm_token=[lemmatizer.lemmatize(word) for word in word_tokens]

print(lemm_token)
```

```
['I', 'learned', 'that', 'if', 'an', 'electric', 'slicer', 'is', 'used', 'the',
'blade', 'becomes', 'hot', 'enough', 'to', 'start', 'to', 'cook', 'the',
'prosciutto', '.']
```

5 POS tagging

[24]:
```
import nltk
txt= "Text mining is also refeered as text data Mining,rough equivalent text␣
 ↪analytics  is the process of derivating"

wordli=nltk.word_tokenize(txt)

tag=nltk.pos_tag(wordli)

print(tag)
```

```
[('Text', 'NN'), ('mining', 'NN'), ('is', 'VBZ'), ('also', 'RB'), ('refeered',
'VBN'), ('as', 'IN'), ('text', 'NN'), ('data', 'NNS'), ('Mining', 'NNP'), (',',
','), ('rough', 'JJ'), ('equivalent', 'JJ'), ('text', 'NN'), ('analytics',
'NNS'), ('is', 'VBZ'), ('the', 'DT'), ('process', 'NN'), ('of', 'IN'),
('derivating', 'VBG')]
```

[26]:
```
# Name Entity Recognisation

doc='''
      Apple bought car from Apple store from Stanford University '''



#tokenize doc

tokenizedoc=nltk.word_tokenize(doc)
tagged_sentenc=nltk.pos_tag(tokenizedoc)

nechunck=nltk.ne_chunk(tagged_sentenc)

print(nechunck)
named_entity=[]

for tagged_tree in nechunck:

    if hasattr(tagged_tree,"label"):
        entity_name=' '.join(c[0] for c in tagged_tree.leaves())
        entity_type=tagged_tree.label()
        named_entity.append((entity_name,entity_type))
print(named_entity)
```

```
(S
  (PERSON Apple/NNP)
  bought/VBD
  car/NN
  from/IN
  (GPE Apple/NNP)
  store/NN
  from/IN
  (ORGANIZATION Stanford/NNP University/NNP))
[('Apple', 'PERSON'), ('Apple', 'GPE'), ('Stanford University', 'ORGANIZATION')]
```

[27]:
```
# Vectorisation

from sklearn.feature_extraction.text import CountVectorizer

countvect1=CountVectorizer()

dtm1=pd.DataFrame(countvect1.fit_transform(usernew["Review"]).
 ↪toarray(),columns=countvect1.get_feature_names())

dtm1
```

[27]:
```
   an  becomes  blade  but  chiles  clean  cook  don  electric  enough  …  \
0   0        0      0    0       0      0     0    0         0       0  …
1   1        1      1    0       0      0     1    0         1       1  …
2   0        0      0    1       1      1     0    1         0       0  …

   prosciutto  slicer  start  that  the  they  this  to  used  wow
0           0       0      0     0    0     0     1   0     0    1
1           1       1      1     1    2     0     0   2     1    0
2           0       0      0     0    1     1     0   0     0    0

[3 rows x 26 columns]
```

[28]:
```
# TF -IDF vectorizer

from sklearn.feature_extraction.text import TfidfVectorizer
countvect2=TfidfVectorizer()
dtm2=pd.DataFrame(countvect2.fit_transform(usernew["Review"]).
 ↪toarray(),columns=countvect2.get_feature_names())
dtm2
```

[28]:
```
         an   becomes     blade       but    chiles     clean      cook  \
0  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1  0.216607  0.216607  0.216607  0.000000  0.000000  0.000000  0.216607
2  0.000000  0.000000  0.000000  0.423394  0.423394  0.423394  0.000000

        don  electric    enough  …  prosciutto    slicer     start  \
0  0.000000  0.000000  0.000000  …    0.000000  0.000000  0.000000
1  0.000000  0.216607  0.216607  …    0.216607  0.216607  0.216607
2  0.423394  0.000000  0.000000  …    0.000000  0.000000  0.000000
```

```
       that       the      they  this        to      used  wow
0  0.000000  0.000000  0.000000   0.5  0.000000  0.000000  0.5
1  0.216607  0.329470  0.000000   0.0  0.433213  0.216607  0.0
2  0.000000  0.322002  0.423394   0.0  0.000000  0.000000  0.0

[3 rows x 26 columns]
```

[ ]:

```

```