NLTK TOOLKIT

NLP ASSIGNMENT

2022503003

R.Prabhakara Arjun aka themysterysolver

Repo_link https://github.com/themysterysolver/ML-Learn/tree/main/NLTK

(Doc generated by py script 😊)

NLTK EXPLORE

TEXT PRE PROCESSING

```
para="My soldiers scream out. My soldiers rage!"
```

CODE:

```
!pip install nltk
```

OUTPUT:

```
Requirement already satisfied: nltk in
c:\users\dell\anaconda3\lib\site-packages (3.9.1)
Requirement already satisfied: click in
c:\users\dell\anaconda3\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in
c:\users\dell\anaconda3\lib\site-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
c:\users\dell\anaconda3\lib\site-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in
c:\users\dell\anaconda3\lib\site-packages (from nltk) (4.66.5)
Requirement already satisfied: colorama in
c:\users\dell\anaconda3\lib\site-packages (from click->nltk) (0.4.6)
```

CODE:

```
import nltk
nltk.download('punkt_tab')
```

OUTPUT:

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
```

CODE:

```
from nltk.tokenize import word_tokenize,
sent_tokenize,WhitespaceTokenizer,RegexpTokenizer,wordpunct_tokenize
word_tokenize(para)
```

CODE:

```
sent_tokenize(para)
```

CODE:

```
WhitespaceTokenizer().tokenize(para)
```

CODE:

```
wordpunct_tokenize(para)
```

CODE:

```
RegexpTokenizer(r'\.+',gaps=True).tokenize(para)
```

CLEANSING

CODE:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

OUTPUT:

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
```

```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,WordNetLemmatizer
```

CODE:

```
para.lower()
```

CODE:

```
stop_checker="a the an is was a The goated! And abd is Mr.360"
tokens=word_tokenize(stop_checker.lower())
print(tokens)
```

OUTPUT:

```
['a', 'the', 'an', 'is', 'was', 'a', 'the', 'goated', '!', 'and',
'abd', 'is', 'mr.360']
```

CODE:

```
[word for word in tokens if word not in stopwords.words('english')]
```

CODE:

```
import re
w=[word for word in tokens if word not in stopwords.words('english')]
str_digits_removed=re.sub(r'\d+','',' '.join(w))
print(str_digits_removed)
```

OUTPUT:

```
goated ! abd mr.
```

CODE:

```
[PorterStemmer().stem(word) for word in tokens if word not in
stopwords.words('english')]
```

CODE:

```
[WordNetLemmatizer().lemmatize(word) for word in tokens if word not in
stopwords.words('english')]
```

POS TAGGING

CODE:

```
import nltk
nltk.download('averaged_perceptron_tagger_eng',force=True)
```

OUTPUT:

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger_eng.zip.
```

```
from nltk import pos_tag,word_tokenize
```

CODE:

```
pos_check="I loved a girl.Beuatiful and sweet,I never knew you were the
one waiting for me!!"
pos_tag(word_tokenize(pos_check))
```

NER(Named Entity relation)

CODE:

```
nltk.download('maxent_ne_chunker_tab')
nltk.download('words')
```

OUTPUT:

```
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker_tab.zip.
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

```
from nltk import ne_chunk
from nltk.tree import Tree
```

CODE:

```
sentence = pos_tag(word_tokenize("Barack Obama was the unique and noble
president of the USA."))
tree = ne_chunk(sentence)
print(tree)
```

OUTPUT:

```
(S
  (PERSON Barack/NNP)
  (PERSON Obama/NNP)
  was/VBD
  the/DT
  unique/JJ
  and/CC
  noble/JJ
  president/NN
  of/IN
  the/DT
  (ORGANIZATION USA/NNP)
  ./.)
```

CORPUS AND WORDNET

CODE:

```
nltk.download('gutenberg')
```

OUTPUT:

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\gutenberg.zip.
```

CODE:

```
from nltk.corpus import wordnet
import nltk
nltk.download('wordnet')
```

OUTPUT:

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

CODE:

```
from nltk.corpus import gutenberg
gutenberg.words('austen-emma.txt')
```

CODE:

```
wordnet.synsets('love')
```

CODE:

```
for sys in wordnet.synsets('love'):
    print(sys.name(),sys.definition())
```

OUTPUT:

```
love.n.01 a strong positive emotion of regard and affection
love.n.02 any object of warm affection or devotion
beloved.n.01 a beloved person; used as terms of endearment
love.n.04 a deep feeling of sexual desire and attraction
love.n.05 a score of zero in tennis or squash
sexual_love.n.02 sexual activities (often including sexual intercourse)
between two people
love.v.01 have a great affection or liking for
love.v.02 get pleasure from
love.v.03 be enamored or in love with
```

```
sleep_together.v.01 have sexual intercourse with
```

FREQ

```
import nltk
from nltk import FreqDist
```

CODE:

```
FreqDist(word_tokenize(para))
```

CODE:

```
FreqDist(word_tokenize(para)).most_common(3)
```

N-GRAM MODEL

```
from nltk.util import ngrams
```

```
from nltk.corpus import gutenberg
gut=gutenberg.words('austen-emma.txt')
```

CODE:

```
list(ngrams(gut,2))[:4]
```

SCKIT LEARN

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from nltk.util import ngrams
```

CODE:

```
#generated
messages = [
    "Free entry in 2 a wkly comp to win FA Cup final tkts",      # spam
    "U dun say so early hor... U c already then say...",         # ham
    "WINNER!! As a valued network customer you have been selected to
receive a prize",  # spam
```

```
    "Hey, are we still meeting for dinner tonight?",             # ham
    "Six chances to win CASH! Just text WIN to 80086",           # spam
    "I'll call you later when I'm free",                         # ham
]
labels=[1,0,1,0,1,0]


def to_trigram(texts):
    trigram_texts = []
    for msg in texts:
        tokens = msg.lower().split()
        trigrams = list(ngrams(tokens, 3))
        trigram_texts.append(' '.join(['_'.join(t) for t in trigrams]))
    return trigram_texts

trigram_msgs = to_trigram(messages)

vectorizer=CountVectorizer()
X=vectorizer.fit_transform(trigram_msgs)

X_train, X_test, y_train, y_test = train_test_split(X, labels,
test_size=0.33, random_state=42)
model = MultinomialNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, target_names=["ham",
"spam"],zero_division=0))#zero_division error
```

OUTPUT:

```
              precision    recall  f1-score   support

         ham       0.50      1.00      0.67         1
        spam       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```