**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