NATURAL LANGUAGE PROCESSING (SE 313)

Practical File
(2024- 2025)

Submitted By
Sanoj (2K21/SE/159)

Under the Supervision of
Prof. Geetanjali Garg

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

INDEX

S.No	Experiment Name	Date	Teacher Sign	Remarks
1.	Import nltk and download the ‘stopwords’ and ‘punkt’ packages	13 Jan 2023		
2.	Import spacy and load the language model.	13 Jan 2023		
3.	WAP in python to tokenize a given text.	20 Jan 2023		
4.	WAP in python to get the sentences of a text document.	3 Feb 2023		
5.	WAP in python to tokenize text with stopwords as delimiters.	3 Feb 2023		
6.	WAP in python to add custom stop words in spaCy.	3 Feb 2023		
7.	WAP to remove punctuations, perform stemming, lemmatize given text and extract usernames from emails	24 Feb 2023		
8.	WAP to do spell correction, extract all nouns, pronouns and verbs in a given text	7 Mar 2023		
9.	WAP to find similarity between two words and classify a text as positive/negative sentiment	31 Mar 2023		

SANOJ
2K21/SE/159

EXPERIMENT 1 – NLTK, STOPWORDS AND PUNKT

OBJECTIVE
Import nltk and download the ‘stopwords’ and ‘punkt’ packages.

CODE:
```
import nltk

nltk.download('stopwords')

nltk.download('punkt')
```

OUTPUT:

In [1]:
```
import nltk

nltk.download('stopwords')

nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/techysanoj/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/techysanoj/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[1]: True

SANOJ
2K21/SE/159

EXPERIMENT 2 – SPACY AND LOAD THE MODEL

OBJECTIVE
Import spacy and load the language model.

CODE:
import spacy

nlp_eng = spacy.load('en_core_web_sm')

nlp_multi = spacy.load('xx_ent_wiki_sm')

OUTPUT:

In [3]:
```
pip install spacy
```

```
Collecting spacy
  Obtaining dependency information for spacy from https://files.pythonhosted.org/packages/c4/c5/1a4556a372ce1bd53f183d583126a6535cae6baa1b09b7028faf018c8a67/spacy-3.7.4-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading spacy-3.7.4-cp311-cp311-macosx_11_0_arm64.whl.metadata (27 kB)
Collecting spacy-legacy<3.1.0,>=3.0.11 (from spacy)
  Obtaining dependency information for spacy-legacy<3.1.0,>=3.0.11 from https://files.pythonhosted.org/packages/c3/55/12e842c70ff8828e34e543a2c7176dac4da006ca6901c9e8b43efab8bc6b/spacy_legacy-3.0.12-py2.py3-none-any.whl.metadata
  Downloading spacy_legacy-3.0.12-py2.py3-none-any.whl.metadata (2.8 kB)
Collecting spacy-loggers<2.0.0,>=1.0.0 (from spacy)
  Obtaining dependency information for spacy-loggers<2.0.0,>=1.0.0 from https://files.pythonhosted.org/packages/33/78/d1a1a026ef3af911159398c939b1509d5c36fe524c7b644f34a5146c4e16/spacy_loggers-1.0.5-py3-none-any.whl.metadata
  Downloading spacy_loggers-1.0.5-py3-none-any.whl.metadata (23 kB)
Collecting murmurhash<1.1.0,>=0.28.0 (from spacy)
  Obtaining dependency information for murmurhash<1.1.0,>=0.28.0 from https://files.pythonhosted.org/packages/7a/05/4a3b5c3043c6d84c00bf0f574d326660702b1c10174fe6b44cef3c3dff08/murmurhash-1.0.10-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading murmurhash-1.0.10-cp311-cp311-macosx_11_0_arm64.whl.metadata (2.0 kB)
Collecting cymem<2.1.0,>=2.0.2 (from spacy)
  Obtaining dependency information for cymem<2.1.0,>=2.0.2 from https://files.pythonhosted.org/packages/d7/f6/67bab
```

In [5]:
```
import spacy
nlp_eng = spacy.load('en_core_web_sm')
nlp_multi = spacy.load('xx_ent_wiki_sm')
```

SANOJ
2K21/SE/159

EXPERIMENT 3 – TOKENIZATION

OBJECTIVE
WAP in python to tokenize a given text.

CODE:
```
from nltk import word_tokenize

text = "Last week, the University of Cambridge shared its own research that shows if everyone wears a mask outside home,dreaded ‘second wave’ of the pandemic can be avoided."
text = word_tokenize(text)
for t in text:
    print(t)
```

OUTPUT:
```
Last
week
,
the
University
of
Cambridge
shared
its
own
research
that
shows
if
everyone
wears
a
mask
outside
home
,
dreaded
‘
second
wave
’
of
the
pandemic
can
be
avoided
.
```

SANOJ
2K21/SE/159

EXPERIMENT 4 – SENTENCE IN DOCUMENT

OBJECTIVE
WAP in python to get the sentences of a text document.

CODE:
```
file = open('file.txt')

Input_text = file.read()

ans = Input_text.split('.')

for an in ans:
    print(an,'\n')
```

OUTPUT:
```
In [10]: file = open('file.txt')

         Input_text = file.read()

         ans = Input_text.split('.')

         for an in ans:
             print(an,'\n')

The 'Ramayana' is an ancient Hindu epic about Rama and Sita

 It is one of the two most important ancient epics of [India], the first one being the ancient Mahabharata


 The epic was originally written by sage (rishi) Valmiki of Ancient India

 The book has about 24,000 verses and is divided into six parts
```

SANOJ
2K21/SE/159

EXPERIMENT 5 – TOKENIZATION WITH STOP WORDS

OBJECTIVE
WAP in python to tokenize text with stop words as delimiters.

CODE:
```
text = "Walter was feeling anxious. He was diagnosed today. He probably is the best person I know."

stop_words_and_delims = ['was', 'is', 'the', '.', ',', '-', '!', '?']

for r in stop_words_and_delims:
    text = text.replace(r, 'DELIM')

words = [t.strip() for t in text.split('DELIM')]
words_filtered = list(filter(lambda a: a not in [''], words))

for word in words_filtered:
    print(word)
```

OUTPUT:
```
In [12]: text = "Walter was feeling anxious. He was diagnosed today. He probably is the best person I know."
         stop_words_and_delims = ['was', 'is', 'the', '.', ',', '-', '!', '?']
         for r in stop_words_and_delims:
             text = text.replace(r, 'DELIM')
         words = [t.strip() for t in text.split('DELIM')]
         words_filtered = list(filter(lambda a: a not in [''], words))
         for word in words_filtered:
             print(word)

         Walter
         feeling anxious
         He
         diagnosed today
         He probably
         best person I know
```

SANOJ
2K21/SE/159

EXPERIMENT 6 – CUSTOM WORDS IN spaCy

OBJECTIVE
WAP in python to add custom stop words in spaCy.

CODE:
```
import spacy
nlp = spacy.load('en_core_web_sm')
custom_stop_words = ['was', 'is','the','JUNK','NIL','of','more' ,'.', ',', '-', '!', '?','a']

for word in custom_stop_words:
   nlp.vocab[word].is_stop = True
doc = nlp("Jonas was a JUNK great guy NIL Adam was evil NIL Martha JUNKwas more of a fool")

for token in doc:
   if not token.is_stop:
      print(token.text, end=" ")
```

OUTPUT:

In [1]:
```
import spacy
nlp = spacy.load('en_core_web_sm')
custom_stop_words = ['was', 'is','the','JUNK','NIL','of','more' ,'.', ',', '-', '!', '?','a']
for word in custom_stop_words:
    nlp.vocab[word].is_stop = True
doc = nlp("Jonas was a JUNK great guy NIL Adam was evil NIL Martha JUNKwas more of a fool")
for token in doc:
    if not token.is_stop:
        print(token.text, end=" ")
```

```
Jonas great guy Adam evil Martha JUNKwas fool
```

SANOJ
2K21/SE/159

EXPERIMENT 7 – STEMMING

OBJECTIVE
WAP to remove punctuations, perform stemming, lemmatize given text and extract usernames from emails.

CODE:
```
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
string = "Jonas!!! great \\guy <> Adam --evil [Martha] ;;fool() ."
ans = ""
for char in string:
   if char not in punctuations:
      ans+=char
      print(ans)

from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
text= "Dancing is an art. Students should be taught dance as a subject in schools . I danced in many of my school function. Some people are always hesitating to dance."
ans = ""
stemmer = PorterStemmer()
tokens = word_tokenize(text)
for token in tokens:
   ans+=stemmer.stem(token)
   ans+=" "
print(ans)

from nltk.corpus import wordnet
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
text= "Dancing is an art. Students should be taught dance as a subject in schools . I danced in many of my school function. Some people are always hesitating to dance."
ans = ""
tokens = word_tokenize(text)
for token in tokens:
   ans+=lemmatizer.lemmatize(token, wordnet.VERB)
   ans+=" "
print(ans)

from nltk.tokenize import word_tokenize
text= "The new registrations are potter709@gmail.com , elixir101@gmail.com. If you find any disruptions, kindly contact granger111@gamil.com or severus77@gamil.com "
text_list = word_tokenize(text)
usernames = []
for i in range(len(text_list)):
   if text_list[i] == "@":
      usernames.append(text_list[i-1])
print(usernames)
```

SANOJ
2K21/SE/159

OUTPUT:

In [7]:
```
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
string = "Jonas!!! great \\guy <> Adam --evil [Martha] ;;fool() ."
ans = ""
for char in string:
    if char not in punctuations:
        ans+=char
print(ans)
```

```
Jonas great guy  Adam evil Martha fool
```

In [4]:
```
om nltk.stem import PorterStemmer
om nltk.tokenize import word_tokenize
xt= "Dancing is an art. Students should be taught dance as a subject in schools . I danced in many of my school func
s = ""
emmer = PorterStemmer()
kens = word_tokenize(text)
r token in tokens:
  ans+=stemmer.stem(token)
  ans+=" "
int(ans)
```

```
danc is an art . student should be taught danc as a subject in school . i danc in mani of my school function . some
peopl are alway hesit to danc .
```

In [6]:
```
from nltk.tokenize import word_tokenize
text= "The new registrations are potter709@gmail.com , elixir101@gmail.com. If you find any disruptions, kindly cont
text_list = word_tokenize(text)
usernames = []
for i in range(len(text_list)):
    if text_list[i] == "@":
        usernames.append(text_list[i-1])
print(usernames)
```

```
['potter709', 'elixir101', 'granger111', 'severus77']
```

SANOJ
2K21/SE/159

EXPERIMENT 8 – SPELL CORRECTION

OBJECTIVE
WAP to do spell correction, extract all nouns, pronouns and verbs in a given text.

CODE:
```
import textblob
from textblob import TextBlob
text="He is a gret person. He beleives in bod"
textb = TextBlob(text)
correct_text = textb.correct()
print(correct_text)

import nltk
from nltk import word_tokenize, pos_tag
text="James works at Microsoft. She lives in manchester and likes to play the flute"
tokens = word_tokenize(text)
parts_of_speech = nltk.pos_tag(tokens)
nouns = list(filter(lambda x: x[1] == "NN" or x[1] == "NNP",
parts_of_speech))
for noun in nouns:
   print(noun[0])

from nltk import pos_tag, word_tokenize
text = "I may bake a cake for my birthday. The talk will introduce reader about Use of baking"
words = word_tokenize(text)
verb_phrases = []
for i in range(len(words)):
   if i > 0 and pos_tag(words)[i][1] == 'VB':
      verb_phrase = words[i-1] + ' ' + words[i]
      verb_phrases.append(verb_phrase)
for i in verb_phrases:
   print (i)
```

OUTPUT:
```
He is a great person. He believes in god
```

```
James
Microsoft
manchester
flute
```

```
may bake
will introduce
```

SANOJ
2K21/SE/159

EXPERIMENT 9 – SIMILARITY BETWEEN WORDS

OBJECTIVE
WAP to find similarity between two words and classify a text as positive/negative sentiment.

CODE:
```
import spacy
nlp = spacy.load('en_core_web_md')
words = "amazing terrible excellent"
tokens = nlp(words)
token1, token2, token3 = tokens[0], tokens[1], tokens[2]
print(f"Similarity between {token1} and {token2} : ",
token1.similarity(token2))
print(f"Similarity between {token1} and {token3} : ",
token1.similarity(token3))

from textblob import TextBlob
text = "It was a very pleasant day"
print(TextBlob(text).sentiment)
```

OUTPUT:
```
In [19]: import spacy
         nlp = spacy.load('en_core_web_md')
         words = "amazing terrible excellent"
         tokens = nlp(words)
         token1, token2, token3 = tokens[0], tokens[1], tokens[2]
         print(f"Similarity between {token1} and {token2} : ",
         token1.similarity(token2))
         print(f"Similarity between {token1} and {token3} : ",
         token1.similarity(token3))

         Similarity between amazing and terrible :  0.4538300335407257
         Similarity between amazing and excellent :  0.7237433791160583

In [20]: from textblob import TextBlob
         text = "It was a very pleasant day"
         print(TextBlob(text).sentiment)

         Sentiment(polarity=0.9533333333333333, subjectivity=1.0)
```

SANOJ
2K21/SE/159