Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL504 **DATE: 4/11/22**

COURSE NAME: Artificial Intelligence Laboratory **CLASS: TYBTech-IT**

EXPERIMENT NO. 6

CO/LO: Apply NLP algorithms and methods to solve domain-specific problems

AIM: To perform Text Processing on a particular dataset using NLTK.

DESCRIPTIONOF EXPERIMENT:

a. Steps in Text Processing
   1. Sentence Tokenization
   2. Word Tokenization
   3. Text Lemmatization and Stemming
   4. Removing Stop Words
   5. Regular Expressions
   6. Bag of Words
   7. Term Frequency – Inverse Document Frequency (TF-IDF)

b. Bag of Words
   - Bag of words is a Natural Language Processing technique of text modelling. In technical terms, we can say that it is a method of feature extraction with text data. This approach is a simple and flexible way of extracting features from documents.
   - A bag of words is a representation of text that describes the occurrence of words within a document. We just keep track of word counts and disregard the grammatical details and the word order. It is called a “bag” of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

c. TF-IDF
   - One approach of scoring words is to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words like “the” that are also frequent across all documents are penalized. This approach is called term frequency-inverse document frequency or shortly known as TF-IDF approach of scoring.

TECHNOLOGY STACK USED: python

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

SOURCE CODE and OUTPUT:

1) **Tokenization by word, sentence, paragraph :**

```
doc=open("this.txt" , "r")

def tokenizationWords(para):
    print("tokenization by words : ")
    para.replace(".","")
    print(para.split())

def tokenizationSentence(para):
    print("tokenization by sentence: ")
    l=para.split(".")
    l.remove("")
    print(l)

def tokenizationPara(para):
    print("tokenization by para : ")
    l=para.split("\n")
    l.remove("")
    print(l)

#tokenizationWords(doc.read())
#tokenizationSentence(doc.read())
tokenizationPara(doc.read())
```

output:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\tokenization.py"
tokenization by words :
['The', 'task', 'of', 'face', 'recognition', 'has', 'been', 'actively', 'researched', 'in', 'recent', 'years.',
'This', 'paper', 'provides', 'an', 'up-to-date', 'review', 'of', 'major', 'human', 'face', 'recognition', 'resea
rch.', 'We', 'first', 'present', 'an', 'overview', 'of', 'face', 'recognition', 'and', 'its', 'applications.', '
Then,', 'a', 'literature', 'review', 'of', 'the', 'most', 'recent', 'face', 'recognition', 'techniques', 'is', '
presented.', 'Description', 'and', 'limitations', 'of', 'face', 'databases', 'which', 'are', 'used', 'to', 'test
', 'the', 'performance', 'of', 'these', 'face', 'recognition', 'algorithms', 'are', 'given.', 'A', 'brief', 'sum
mary', 'of', 'the', 'face', 'recognition', 'vendor', 'test', '(FRVT)', '2002,', 'a', 'large', 'scale', 'evaluati
on', 'of', 'automatic', 'face', 'recognition', 'technology,', 'and', 'its', 'conclusions', 'are', 'also', 'given
.', 'Finally,', 'we', 'give', 'a', 'summary', 'of', 'the', 'research', 'results.', 'Scholarcy', 'Synopsis', 'In'
, 'the', 'world', 'of', 'machine', 'learning,', 'deep', 'learning', 'has', 'made', 'its', 'significant', 'impact
', 'in', 'certain', 'tasks', 'which', 'seemed', 'impossible', 'a', 'few', 'years', 'ago', 'such', 'as', 'lip', '
reading', 'and', 'speech', 'recognition.', 'Kartik', 'Datar', 'and', 'colleagues', '(2020)', 'report', 'that', '
this', 'page', 'was', 'uploaded', 'by', 'Meet', 'Gandhi', 'on', '19', 'February', '2022.', 'The', 'user', 'has',
 'requested', 'enhancement', 'of', 'the', 'downloaded', 'file.', 'Programmed', 'lip-perusing', 'innovation', 'is
', 'one', 'of', 'the', 'significant', 'segments', 'of', 'humanâ€“computer', 'cooperation', 'innovation.', 'It',
'assumes', 'a', 'fundamental', 'job', 'in', 'human', 'language', 'correspondence', 'and', 'visual', 'recognition
.', 'Three', 'different', 'approaches', 'were', 'discussed', 'for', 'lip', 'reading,', 'and', 'it', 'comes', 'wi
th', 'its', 'own', 'advantages', 'and', 'disadvantages.', 'The', 'proposed', 'solution', 'for', 'the', 'problem'
, 'was', 'to', 'adopt', 'bottom-up', 'approach', 'which', 'overcomes', 'the', 'weaknesses', 'of', 'image-based',
 'feature', 'extraction.', '6', 'speakers', 'were', 'included', 'in', 'the', 'research.', 'The', 'researchers',
'recommend', 'that', 'a', 'CNN', 'has', 'the', 'potential', 'to', 'acquire', 'a', 'speaker', 'independent', 'mod
el', 'for', 'the', 'VSR', 'task.', 'Future', 'work', 'is', 'to', 'investigate', 'the', 'possibility', 'of', 'bui
lding', 'a', 'speaker-independent', 'phoneme', 'recognition', 'model', 'by', 'preparing', 'a', 'larger', 'datase
t.']
PS C:\Users\SHREE RAM\Desktop\ai>
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\tokenization.py"
tokenization by sentence:
['\nThe task of face recognition has been actively researched in recent years', ' This paper provides an up-to-d
ate review of major human face recognition research', ' We first present an overview of face recognition and its
 applications', ' Then, a literature review of the most recent face recognition techniques is presented', ' Desc
ription and limitations of face databases which are used to test the performance of these face recognition algor
ithms are given', ' A brief summary of the face recognition vendor test (FRVT) 2002, a large scale evaluation of
 automatic face recognition technology, and its conclusions are also given', ' Finally, we give a summary of the
 research results', '\nScholarcy Synopsis\nIn the world of machine learning, deep learning has made its signific
ant impact in certain tasks which seemed impossible a few years ago such as lip reading and speech recognition',
 '\nKartik Datar and colleagues (2020) report that this page was uploaded by Meet Gandhi on 19 February 2022', '
\nThe user has requested enhancement of the downloaded file', '\nProgrammed lip-perusing innovation is one of th
e significant segments of humanâ€“computer cooperation innovation', '\nIt assumes a fundamental job in human lan
guage correspondence and visual recognition', '\nThree different approaches were discussed for lip reading, and
it comes with its own advantages and disadvantages', '\nThe proposed solution for the problem was to adopt botto
m-up approach which overcomes the weaknesses of image-based feature extraction', '\n\n6 speakers were included i
n the research', ' \n\nThe researchers recommend that a CNN has the potential to acquire a speaker independent m
odel for the VSR task', '\nFuture work is to investigate the possibility of building a speaker-independent phone
me recognition model by preparing a larger dataset', '\n\n\n']
PS C:\Users\SHREE RAM\Desktop\ai>
```

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\tokenization.py"
tokenization by para :
['', 'Kartik Datar and colleagues (2020) report that this page was uploaded by Meet Gandhi on 19 February 2022.'
, 'Three different approaches were discussed for lip reading, and it comes with its own advantages and disadvant
ages.', 'The user has requested enhancement of the downloaded file.', 'Future work is to investigate the possibi
lity of building a speaker-independent phoneme recognition model by preparing a larger dataset.', 'It assumes a
fundamental job in human language correspondence and visual recognition.', 'The researchers recommend that a CNN
 has the potential to acquire a speaker independent model for the VSR task.', '6 speakers were included in the r
esearch. ', 'The proposed solution for the problem was to adopt bottom-up approach which overcomes the weaknesse
s of image-based feature extraction.', 'Programmed lip-perusing innovation is one of the significant segments of
 humanâ€“computer cooperation innovation.', 'The task of face recognition has been actively researched in recent
 years. This paper provides an up-to-date review of major human face recognition research. We first present an o
verview of face recognition and its applications. Then, a literature review of the most recent face recognition
techniques is presented. Description and limitations of face databases which are used to test the performance of
 these face recognition algorithms are given. A brief summary of the face recognition vendor test (FRVT) 2002, a
 large scale evaluation of automatic face recognition technology, and its conclusions are also given. Finally, w
e give a summary of the research results.', 'In the world of machine learning, deep learning has made its signif
icant impact in certain tasks which seemed impossible a few years ago such as lip reading and speech recognition
.', 'Scholarcy Synopsis']
PS C:\Users\SHREE RAM\Desktop\ai>
```

2) Stemming and Lemmatization:

```python
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer

def stemming(words):
    print("#Stemming is a process of normalization, in which words are reduced
to their root word (or) stem : ")
    print("stemming words : ",words)
    snow_stem = SnowballStemmer(language='english')
    for word in words:
        print(word + '--->' + snow_stem.stem(word))
```

```
def lemminization(tokenized_word):
    print("lemmatization is also used to reduce the word to their root word.
Lemmatizing gives the complete meaning of the word which makes sense.It uses
vocabulary and morphological analysis to transform a word into a root word. ")
    print("lemminization words : ",tokenized_word)
    lemmatizer = WordNetLemmatizer()
    lemmatized_words_list = []
    for each_word in tokenized_word:
        lem_word = lemmatizer.lemmatize(each_word)
        lemmatized_words_list.append(lem_word)
    print('Text with Stop Words: {}'.format(tokenized_word))
    print('Lemmatized Words list {}'.format(lemmatized_words_list))


words = ['happy', 'happier', 'happiest', 'happiness',
'breathing','fairly','eating']
stemming(words)
txt="Life will always have problems and pressures."
#lemminization(txt.split())
```

Output:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\exp6_nltk.py"
lemmatization is also used to reduce the word to their root word. Lemmatizing gives the complete meaning of the w
ord which makes sense.It uses vocabulary and morphological analysis to transform a word into a root word.
lemminization words :  ['Life', 'will', 'always', 'have', 'problems', 'and', 'pressures.']
Text with Stop Words: ['Life', 'will', 'always', 'have', 'problems', 'and', 'pressures.']
Lemmatized Words list ['Life', 'will', 'always', 'have', 'problem', 'and', 'pressures.']
PS C:\Users\SHREE RAM\Desktop\ai>
```

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\exp6_nltk.py"
#Stemming is a process of normalization, in which words are reduced to their root word (or) stem :
stemming words :  ['happy', 'happier', 'happiest', 'happiness', 'breathing', 'fairly', 'eating']
happy--->happi
happier--->happier
happiest--->happiest
happiness--->happi
breathing--->breath
fairly--->fair
eating--->eat
PS C:\Users\SHREE RAM\Desktop\ai>
```

Activate
Go to Settir

3) Stop words:

```
from nltk import word_tokenize
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
text = 'Learn to lose your destiny to find where it leads you'
filtered_text = []
tokenized_word = word_tokenize(text)
for each_word in tokenized_word:
```

```
    if each_word not in stop_words:
        filtered_text.append(each_word)
print('Toxenized list with stop words: {}'.format(tokenized_word))
print('Toxenized list with out stop words: {}'.format(filtered_text))
```

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\stopwords.py"
Toxenized list with stop words: ['Learn', 'to', 'lose', 'your', 'destiny', 'to', 'find', 'where', 'it', 'leads',
'you']
Toxenized list with out stop words: ['Learn', 'lose', 'destiny', 'find', 'leads']
PS C:\Users\SHREE RAM\Desktop\ai>
```

4) Pos:

```
import nltk

text = "I am going to meet M.S. Dhoni."
tokenized_word = text.split()
print(nltk.pos_tag(tokenized_word, tagset='universal'))
```

output:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\tempCodeRunnerFile.py"
[('I', 'PRON'), ('am', 'VERB'), ('going', 'VERB'), ('to', 'PRT'), ('meet', 'VERB'), ('M.S.', 'NOUN'), ('Dhoni.',
'NOUN')]
PS C:\Users\SHREE RAM\Desktop\ai>
```

5) Bag of words:

```
n=int(input("Enter number of sentence to build vocabulary : "))
sentences = []
for i in range(0,n):
    l=input()
    sentences.append(l)

vocab = set()
for i in sentences:
    for j in i.split():
        vocab.add(j)

print("vocab: ",vocab)
print("sentence: ",sentences)
voclist = list(vocab)
bow=[]
```

```
for i in sentences:
    vi=[]
    for k in range(0,len(vocab)):
        vi.append(0)

    l=i.split()
    for j in range(0,len(vi)):
        if voclist[j] in l:
            vi[j] = vi[j]+1

    bow.append(vi)
print("bag of words: ",bow)
```

output:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\bow_general.py"
Enter number of sentence to build vocabulary : 2
hello world
hell u world
vocab:  {'u', 'hello', 'world', 'hell'}
sentence:  ['hello world', 'hell u world']
bag of words:  [[0, 1, 1, 0], [1, 0, 1, 1]]
PS C:\Users\SHREE RAM\Desktop\ai>
```

6) Bag of words for nouns only:

```
import nltk

def get(sentence):
    t=[]
    tokenized_word = sentence.split()
    need=nltk.pos_tag(tokenized_word, tagset='universal')
    for i in range(0,len(need)):
        if(need[i][1]=="NOUN"):
            t.append(need[i][0])

    return t

n=int(input("enter number of sentences to build vocabulary : "))
sentences=[]
for i in range(0,n):
    l=input()
    sentences.append(l)

vocab=set()
for i in sentences:
    t=get(i)
```

```
    for k in t:
        vocab.add(k)

print("vocab : ",vocab)
print("sentences : ",sentences)
voclist=list(vocab)
bow=[]
for i in sentences:
    vi=[]
    for m in range(0,len(vocab)):
        vi.append(0)

    l=i.split()
    for i in range(0,len(vi)):
        if voclist[i] in l:
            vi[i]=vi[i]+1

    bow.append(vi)

print("bag of words : ")
print(bow)
```

output:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\tempCodeRunnerFile.py"
enter number of sentences to build vocabulary : 2
cat is drinking milk
jack and jim went up the hill
vocab :  {'cat', 'jim', 'milk', 'jack', 'hill'}
sentences :  ['cat is drinking milk', 'jack and jim went up the hill']
bag of words :
[[1, 0, 1, 0, 0], [0, 1, 0, 1, 1]]
PS C:\Users\SHREE RAM\Desktop\ai>
```

7) TF-IDF:

```
from sklearn.feature_extraction.text import TfidfVectorizer

d0 = 'hello world'
d1 = 'hell you world'
d2 = 'hello from the otherside of world'

corpus = [d0, d1, d2]

tfidf = TfidfVectorizer()
result = tfidf.fit_transform(corpus)


print('\nWord indexes:')
```

```
print(tfidf.vocabulary_)
print('\ntf-idf value:')
print(result)
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

output:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\tfids.py"

Word indexes:
{'hello': 2, 'world': 6, 'hell': 1, 'you': 7, 'from': 0, 'the': 5, 'otherside': 4, 'of': 3}

tf-idf value:
  (0, 6)        0.6133555370249717
  (0, 2)        0.7898069290660905
  (1, 7)        0.652490884512534
  (1, 1)        0.652490884512534
  (1, 6)        0.3853716274664007
  (2, 3)        0.450504072643198
  (2, 4)        0.450504072643198
  (2, 5)        0.450504072643198
  (2, 0)        0.450504072643198
  (2, 6)        0.2660749625405929
  (2, 2)        0.3426199591918006

tf-idf values in matrix form:
[[0.         0.         0.78980693 0.         0.         0.
  0.61335554 0.        ]
 [0.         0.65249088 0.         0.         0.         0.
  0.38537163 0.65249088]
 [0.45050407 0.         0.34261996 0.45050407 0.45050407 0.45050407
  0.26607496 0.        ]]
PS C:\Users\SHREE RAM\Desktop\ai>
```

CONCLUSION:

bag-of-words major downside is that because it emphasizes words only on the basis of counts. To overcome this, a simple twist to bag-of-words introduces tf-idf approach.

Unlike, bag-of-words, tf-idf creates a normalized count where each word count is divided by the number of documents this word appears in.

Thus, Tf-Idf makes rare words more prominent and effectively ignores common words. It is closely related to frequency based filter, but much more mathematically elegant than placing hard cutoff thresholds.

While modelling an algorithm on tf-idf feature space, it is important to make use of other feature reduction technique as discussed earlier like stemming, part-of-speech etc, to get an effective results.

Hence, implemented Natural Language Processing using NLTK library, Bag of words and TF-IDF.

REFERENCES:

https://www.analyticsvidhya.com/blog/2021/07/getting-started-with-nlp-using-nltk-library/

https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_stemming_lemmatization.html

https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/