NATURAL LANGUAGE PROCESSING
LAB ASSIGNMENTS

Submitted by:

Christy Binu
Roll number: 97422607009

Msc. Computer Science
With Specialization in
Artificial Intelligence

Department of Computer Science

Q 1: Read a multi page PDF File and Print its First Page

Code:

```
import PyPDF2 as py

pdf = open('pdf.pdf', 'rb')
pdf_reader = py.PdfReader(pdf)

page = pdf_reader.pages[0]
text = page.extract_text()

print(text)
pdf.close()
```

Output:

```
Research  paper  on Artificial
Intelligence


Ashutosh  Kumar
Galgotias  University
(Under  the Uttar  Pradesh  Private  Universities  Act No.  12 of 2019)
Greater  Noida,  India
Email: Ashutosh.20scse1010626@galgotiasuniversity.edu.in

Rachna  Priya
Galgotias  University
(Under  the Uttar  Pradesh  Private  Universites  Act No. 12 of 2019)
Greater  Noida,India
Email:  Rachna.20scse1010564@galgotiasuniversity.edu.in


Swarna  Kumari
Galgotias  University
(Under  the Uttar  Pradesh  Private  Universites  Act No. 12 of 2019)
Greater  Noida,India
Email:  Swarna.20scse1010565@galgotiasuniversity.edu.in


Under the  guidance  of

Mr. Pradeep Bedi
Assistant Professor
(Galgotias  University)
Greater  Noida,  India
Email:  pradeepbedi@galgotiasuniversity. edu.in
```

Q 2: Read Text from an File and Print them as Tokens(Use Alice Text passage)

Code:
```python
from nltk import word_tokenize

text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
textString = ""

for i in text:
  textString += i

text_tokenized = word_tokenize(textString)

token_list = {}

for i in text_tokenized:
  if i not in token_list:
    token_list[i] = 1
  else :
    token_list[i] += 1

for word in token_list:
  print('{:<13} '.format(word))
```

Output:

```
Alice                         peeped
was                           into
beginning                     book
to                            reading
get                           but
very                          it
tired                         no
of                            pictures
sitting                       conversations
by                            in
her                           `
sister                        what
on                            is
the                           use
bank                          a
,                             '
and                           thought
having                        without
nothing                       conversation
do                            ?
:
once
or
twice
she
had
```

Q 3: Remove Stop Words FromText data

Code:

```
from nltk.corpus import stopwords
from nltk import word_tokenize

stopWords = stopwords.words('english')

text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
text = text.read()

tokenized_text = word_tokenize(text)

textString = []

for i in tokenized_text:
    if i not in stopWords:
        textString.append(i)

for text in textString:
    print(text, end = " ")
```

Output:

```
Alice beginning get tired sitting sister bank , nothing : twice peeped book sister reading , pictures
conversations , ` use book , ' thought Alice ` without
```

Q 4: Convert Text Data Into Lowercase

Code:

```
text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
text = text.read()

textString = text.lower()

print("Input text :", text, sep="\n",end="\n\n")
print("Converted Lower case text:", textString, sep="\n", end='\n\n')
```

Output:

```
Input text :
Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to
do: once or twice she had peeped into the book her sister was reading, but it had no pictures or
conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'

Converted Lower case text:
alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to
do: once or twice she had peeped into the book her sister was reading, but it had no pictures or
conversations in it, `and what is the use of a book,' thought alice `without pictures or conversation?'
```

Q 5: Remove Punctuations from text data

Code:

```
from nltk import word_tokenize
from string import punctuation

text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
text = text.read()

tokenized_text = word_tokenize(text)
textString = ""

for token in tokenized_text:
    if token not in punctuation:
        textString +=  " " + token

print(textString)
```

Output:

```
Alice was beginning to get very tired of sitting by her sister on the bank and of having nothing to do
once or twice she had peeped into the book her sister was reading but it had no pictures or
conversations in it and what is the use of a book thought Alice without pictures or conversation
```

Q 6: Print The frequency of words in a document

Code:

```
from nltk import word_tokenize

text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
textString = ""

for i in text:
  textString += i

text_tokenized = word_tokenize(textString)

token_list = {}

for i in text_tokenized:
  if i not in token_list:
    token_list[i] = 1
  else :
    token_list[i] += 1

for word in token_list:
  print('{:<13}  {:<12} '.format(word, token_list[word]))
```

Output:

```
Alice          2
was            2
beginning      1
to             2
get            1
very           1
tired          1
of             3
sitting        1
by             1
her            2
sister         2
on             1
the            3
bank           1
,              4
and            2
having         1
nothing        1
do             1
:              1
once           1
or             3
twice          1
she            1
had            2
peeped         1
into           1
book           2
reading        1
but            1
it             2
no             1
pictures       2
conversations  1
in             1
`              2
what           1
is             1
use            1
a              1
'              2
thought        1
without        1
conversation   1
?              1
```

Q 7: Extract Entity from a text

Code:

```
import spacy

# nlp = spacy.load('en_core_web_sm')
# nlp = spacy.load('en_core_web_md')

nlp = spacy.load('en_core_web_lg')
text = "England won the 2019 world cup vs The 2019 world cup"

# text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
# text = text.read()

doc = nlp(text)

for ent in doc.ents:
    print(ent.text, ent.label_)
```

Input: England won the 2019 world cup vs The 2019 world cup

Output:

```
England GPE
2019 DATE
2019 DATE
```

Q 8: Create a custom Lookup Dictionary and Create a Custom Function For text Standardization

Code:

```
from nltk import word_tokenize

customDictionary = {
    'LOL': 'Laughing out loud',
    'ASAP': 'As soon as possible',
    'FYI': 'For your information',
    'G2G': 'Got to go',
    'FB': 'Facebook',
    'MSG': 'Message',
    'TTYL': 'Talk to you later',
    'IMO': 'In my opinion'
}

text = input("Enter any text: ")
tokenized_text = word_tokenize(text)

textString = " "
for word in tokenized_text:
    if word in customDictionary:
        textString += customDictionary[word] + " "
    else:
        textString += word +" "

print("The standardized form of given text: ", textString, sep="\n")
```

Input:

```
Enter any text: Come here ASAP!
```

Output:

```
The standardized form of given text:
 Come here As soon as possible !
```

Q 9: Correct spelling mistakes of Given Words

Code:

```
from textblob import TextBlob


text = open(r"2nd Sem\NLP\alicespellingMistake.txt")
textString = ""

for i in text:
    textString += i + " "

print('Text with error: ')
print(textString)

textString = TextBlob(textString)
correctedText = textString.correct()

print('\nCorrected Text:')
print(correctedText)
```

Input:

```
Text with error:
Alicpe was begwinning to get vtery tiregd of sitting by her sisteur on trhe bank, and of having
nothing to do: once or twicqe she had peeped into the blook her sister whas reading, butf it hadp
ngo picturmes or conversationgs in it, `and whaat is the use ofc a book,' thoughtf Alicej `without
pictures or conversation?'
```

Output:

```
Corrected Text:
```

```
Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to
do: once or twice she had peeped into the look her sister was reading, but it had no pictures or
conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'
```

Q 10: Perform steaming and lemmatization on Text

Code:

```
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

text = input("enter any text:\n")

word_stemmer = PorterStemmer()
stemmedText = word_stemmer.stem(text)
print("\nStem of text: ", stemmedText)

word_Lemmatizer = WordNetLemmatizer()
lemmatizedText = word_Lemmatizer.lemmatize(text)
print("\nlemma of text: ", lemmatizedText)
```

Input:

```
Writing
```

Output:

```
Stem of text:  write

lemma of text:  Writing
```

Q 11: Generate N-Grams For a Given Sentence

Code:

```
from digitToWord import digitToWord

def generate_ngram(text, ngram):
    words = text.split()
    if len(words) < ngram:
        ngram = len(words) - 1
    output = []

    for i in range(len(words) - ngram):
        output.append(words[i:i+ngram])
    return output

text = 'this is a very good book to study'
print("Given text:",text,sep="\n", end="\n\n")

n = int(input("Enter n value : "))
```

```
if n == 0:
    print("\nN must be greater than 0\n")
    exit()
else :
    word_for_n = digitToWord(n)
    print(f"\n{word_for_n}-gram Model for given text :\n")



generated_ngram = generate_ngram(text, n)


for i in generated_ngram:
    print(f"{i[len(i)-1]} | ", end="")
    for j in range(0, len(i)-1):
      print(i[j], end=" ")
    print("\n")
```

Input:

```
this is a very good book to study

Enter n value : 2
```

Output:

```
Bi-gram Model for given text :

is | this

a | is

very | a

good | very

book | good

to | book
```

Q 12: Convert Text Features using one hot Encoding

Code:

```
from sklearn.feature_extraction.text import CountVectorizer

text = ['this is a very good book to study',
        'This is second sentence',
        'This is third sentence']

# Initialize CountVectorizer to perform one-hot encoding
vectorizer = CountVectorizer(binary=True)
```

```
# Fit and transform the text data
one_hot_encoded = vectorizer.fit_transform(text)

# Get the feature names (unique words)
feature_names = vectorizer.get_feature_names_out()

# Display the one-hot encoded vectors and feature names
print("One-hot encoded vectors:")
print(one_hot_encoded.toarray())
print("\nFeature names:")
print(feature_names)
```

Input:

```
'this is a very good book to study',
'This is second sentence',
'This is third sentence'
```

Output:

```
One-hot encoded vectors:
[[1 1 1 0 0 1 0 1 1 1]
 [0 0 1 1 1 0 0 1 0 0]
 [0 0 1 0 1 0 1 1 0 0]]

Feature names:
['book' 'good' 'is' 'second' 'sentence' 'study' 'third' 'this' 'to' 'very']
```

Q 13: Convert Text Features using a count vectoriser

Code:

```
from nltk import word_tokenize
import pandas as pd

text_array = ['Hello my name is james', 'this is my python notebook']
tokenized_text = []

for i in range(len(text_array)):
    tokenized_text.append(word_tokenize(text_array[i]))

uniqueWords = {}

for text in tokenized_text:
    for word in text:
        if word not in uniqueWords:
            uniqueWords[word] = []


for word in uniqueWords:
    for text in tokenized_text:
        if word in text:
            uniqueWords[word].append(1)
        else:
            uniqueWords[word].append(0)

df = pd.DataFrame(uniqueWords)
print(df)
```

Input:

'Hello my name is james', 'this is my python notebook'

Output:

```
   Hello  my  name  is  james  this  python  notebook
0     1   1      1     1     1     0      0         0
1     0   1      0     1     0     1      1         1
```

Q 14: Tag The Parts Of Speech (POS Tagging) in a Sentence

Code:

```
from nltk import word_tokenize
from nltk.tag import pos_tag

pos_dict = {
'CC': 'coordinating conjunction',
'CD': 'cardinal digit',
'DT': 'determiner',
'EX': 'existential',
'FW': 'foreign word',
'IN': 'preposition/subordinating conjunction',
'JJ': 'adjective',
'JJR': 'adjective, comparative',
'JJS': 'adjective, superlative' ,
'LS': 'list marker' ,
'MD': 'modal ',
'NN': 'noun, singular' ,
'NNS': 'noun plural' ,
'NNP': 'proper noun, singular  ',
'NNPS': 'proper noun, plural ',
'PDT': 'predeterminer ',
'POS': 'possessive ending parent’s ',
'PRP': 'personal pronoun ',
'PRP$': 'possessive pronoun',
'RB': 'adverb ',
'RBR': 'adverb, comparative ',
'RBS': 'adverb, superlative',
'RP': 'particle ',
'TO':  'to go ',
'UH': 'interjection ',
'VB': 'verb, base form'  ,
'VBD': 'verb, past tense' ,
'VBG': 'verb, gerund/present participle',
'VBN': 'verb, past participle  ',
'VBP': 'verb, sing. present, non-3d' ,
'VBZ': 'verb, 3rd person sing. present' ,
'WDT': 'wh-determiner ',
'WP': 'wh-pronoun ',
'WP$': 'possessive wh-pronoun',
'WRB': 'wh-adverb'
}

text = open(r"2nd Sem\NLP\aliceInWonderLand.txt")
text = text.read()

tokenized_text = word_tokenize(text)
parts_of_speech = pos_tag(tokenized_text)

for token in parts_of_speech:
    if token[1] in pos_dict:
        print('{:<13}  {:<12} '.format(token[0], pos_dict[token[1]]))
```

Output:

```
Alice          proper noun, singular
was            verb, past tense
beginning      verb, gerund/present participle
to             to go
get            verb, base form
very           adverb
tired          adjective
of             preposition/subordinating conjunction
sitting        verb, gerund/present participle
by             preposition/subordinating conjunction
her            possessive pronoun
sister         noun, singular
on             preposition/subordinating conjunction
the            determiner
bank           noun, singular
and            coordinating conjunction
of             preposition/subordinating conjunction
having         verb, gerund/present participle
nothing        noun, singular
to             to go
do             verb, base form
once           adverb
or             coordinating conjunction
twice          verb, base form
she            personal pronoun
had            verb, past tense
peeped         verb, past participle
into           preposition/subordinating conjunction
the            determiner
book           noun, singular
```

```
her            possessive pronoun
sister         noun, singular
was            verb, past tense
reading        verb, gerund/present participle
but            coordinating conjunction
it             personal pronoun
had            verb, past tense
no             determiner
pictures       noun plural
or             coordinating conjunction
conversations  noun plural
in             preposition/subordinating conjunction
it             personal pronoun
and            coordinating conjunction
what           wh-pronoun
is             verb, 3rd person sing. present
the            determiner
use            noun, singular
of             preposition/subordinating conjunction
a              determiner
book           noun, singular
thought        adjective
Alice          proper noun, singular
without        preposition/subordinating conjunction
pictures       noun plural
or             coordinating conjunction
conversation   noun, singular
```

Q 15: Find The Word Error Rate (WER) of a Sentence

Code:

```
from nltk import word_tokenize

reference = "What a bright day"
print("Reference Word : ", reference)

hypothesis = "What a light day"
print("hypothesis Word : ", hypothesis)

reference_tokenized =  word_tokenize(reference)
hypothesis_tokenized = word_tokenize(hypothesis)

sameWordCount = 0

for word in hypothesis_tokenized:
    if word in reference_tokenized:
        sameWordCount += 1

totalCount = len(reference_tokenized)
errorCount = len(reference_tokenized) - sameWordCount
```

```
print(totalCount, errorCount)

wordErrorRate = errorCount/totalCount

print("Word Error rate of given case : ", wordErrorRate)
```

Output:

```
Reference Word :  What a bright day
hypothesis Word :  What a light day
4 1
Word Error rate of given case :  0.25
```