ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL

DEPARTMENT OF CSE-AIML.

LAB MANUAL

Programme :B.Tech

Semester : V

Course Code : AL-504(B)

Subject Name : NATURAL LANGUAGE PROCESSING

Submitted To:

Prof. Ruchi Jain

Asst. Prof. (AIML)

Submitted by:

IT-504(B) NATURAL LANGUAGE PROCESSING

Lab Objectives:

1. Be able to discuss the current and likely future performance of several NLP applications.
2. Be able to describe briefly a fundamental technique for processing language for several subtasks, such as morphological processing.
3. Implement parsing, wordsense disambiguation and etc.
4. Understand how these techniques draw on and relate to other areas of computer science.
5. Understand the basic principles of designing and running an NLP experiment.

Lab Outcomes:

Upon successful completion of this course, the students will be able to:

1. Student will be able to implement LSI, NER.
2. Student will be able to implement TD-IDF method and N gram models.
3. Develop a Part of speech tagger.
4. Student can able classify the text based on part of speech tagger.
5. Student can able to implements ever all NLP applications.

List of Experiment

S.No	Name of Experiment
1	Write a Python program to perform tokenization by word and sentence using NLTK.
2	Write a Python program to perform Parts of Speech tagging using NLTK.
3	Write a Python program to perform lemmatization using NLTK.
4	Write a Python program for chunking using NLTK.
5	Write a Python program for CYK Parsing (Cocke–Younger–Kasami Parsing) or Chart Parsing.
6	Write a Python program to find all unigrams, bigrams, and trigrams present in the given corpus.
7	Write a Python program to find the probability of the given statement **"This is my cat"** by taking an example corpus into consideration.
8	Write a Python program to eliminate stopwords using NLTK.
9	Write a Python program to perform stemming using NLTK.
10	Write the Python code to detect Fake News using NLP.

PROGRAM 1

Write a Python program to perform tokenization by word and sentence using NLTK.

- **Program for Sentence Tokenization**

```
import nltk
nltk.download('punkt')  # Download tokenizer models
from nltk.tokenize import sent_tokenize

def tokenize_sentences(text):
    sentences = sent_tokenize(text)
    return sentences

# Example text
text = ("NLTK is a leading platform for building Python programs to work "
        "with human language data. It provides easy-to-use interfaces to "
        "over 50 corpora and lexical resources such as WordNet, along with
"
        "a suite of text processing libraries for classification,
tokenization, "
        "stemming, tagging, parsing, and semantic reasoning, wrappers for "
        "industrial-strength NLP libraries, and an active discussion
forum.")

# Tokenize sentences
sentences = tokenize_sentences(text)

# Print tokenized sentences
for i, sentence in enumerate(sentences):
    print(f"Sentence {i+1}: {sentence}")
```

- **Program for Word Tokenization**

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize

def tokenize_words(text):
    words = word_tokenize(text)
    return words

# Example text
text = "NLTK is a leading platform for building Python programs to work
with human language data."

# Tokenize words
words = tokenize_words(text)

# Print tokenized words
print(words)
```

Output:

PROGRAM 2

Write a Python program to perform Parts of Speech (POS) tagging using NLTK.

```
import nltk
from nltk.tokenize import word_tokenize

# Download models
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

def pos_tagging(text):
    words = word_tokenize(text)
    tagged_words = nltk.pos_tag(words)
    return tagged_words

# Example text
text = "NLTK is a leading platform for building Python programs to work
with human language data."

# Perform POS tagging
tagged_text = pos_tagging(text)

print(tagged_text)
```

Output:

PROGRAM 3

Write a Python program to perform lemmatization using NLTK.

```
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import nltk

nltk.download('punkt')
nltk.download('wordnet')

def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(text)
    lemmatized_text = ' '.join([lemmatizer.lemmatize(word) for word in
tokens])
    return lemmatized_text

text = "The cats are chasing mice and playing in the garden"
lemmatized_text = lemmatize_text(text)

print("Original Text:", text)
print("Lemmatized Text:", lemmatized_text)
```

Output:

PROGRAM 4

Write a Python program for Chunking using NLTK.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, RegexpParser

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

def chunk_sentence(sentence):
    words = word_tokenize(sentence)
    tagged_words = pos_tag(words)

    grammar = r"""
        NP: {<DT|JJ|NN.*>+}          # Noun Phrase
        PP: {<IN><NP>}               # Prepositional Phrase
        VP: {<VB.*><NP|PP|CLAUSE>+$} # Verb Phrase
        CLAUSE: {<NP><VP>}           # Clause
    """

    parser = RegexpParser(grammar)
    chunked_sentence = parser.parse(tagged_words)

    return chunked_sentence

sentence = "The quick brown fox jumps over the lazy dog"
chunked_sentence = chunk_sentence(sentence)
print(chunked_sentence)
```

Output:

PROGRAM 5

Write a Python program for CYK Parsing / Chart Parsing.

```
import nltk

grammar = nltk.CFG.fromstring("""
S -> V NP
V -> 'describe' | 'present'
NP -> PRP N
PRP -> 'your'
N -> 'work'
""")

parser = nltk.ChartParser(grammar)
sent = 'describe your work'.split()

print(list(parser.parse(sent)))
```

Output:

PROGRAM 6

Write a Python program to find all unigrams, bigrams, and trigrams.

```
import nltk
nltk.download('punkt')
from nltk.util import ngrams

sampleText = 'this is a very good book to study'

for i in range(1, 4):
    NGRAMS = ngrams(sequence=nltk.word_tokenize(sampleText), n=i)
    for grams in NGRAMS:
        print(grams)
```

Output:

PROGRAM 7

Write a Python program to find the probability of the statement "This is my cat".

```
# Corpus
def readData():
    data = ['This is a dog', 'This is a cat', 'I love my cat', 'This is my
name']
    dat = []
    for i in range(len(data)):
        for word in data[i].split():
            dat.append(word)
    print(dat)
    return dat

def createBigram(data):
    listOfBigrams = []
    bigramCounts = {}
    unigramCounts = {}

    for i in range(len(data)-1):
        if i < len(data) - 1 and data[i+1].islower():
            listOfBigrams.append((data[i], data[i+1]))

        if (data[i], data[i+1]) in bigramCounts:
            bigramCounts[(data[i], data[i+1])] += 1
        else:
            bigramCounts[(data[i], data[i+1])] = 1

        if data[i] in unigramCounts:
            unigramCounts[data[i]] += 1
        else:
            unigramCounts[data[i]] = 1

    return listOfBigrams, unigramCounts, bigramCounts

def calcBigramProb(listOfBigrams, unigramCounts, bigramCounts):
    listOfProb = {}
    for bigram in listOfBigrams:
        word1 = bigram[0]
        listOfProb[bigram] = (bigramCounts.get(bigram)) /
(unigramCounts.get(word1))
    return listOfProb

if __name__ == '__main__':
    data = readData()
    listOfBigrams, unigramCounts, bigramCounts = createBigram(data)

    print("\nAll the possible Bigrams are")
    print(listOfBigrams)
```

```
    print("\nBigrams along with their frequency")
    print(bigramCounts)

    print("\nUnigrams along with their frequency")
    print(unigramCounts)

    bigramProb = calcBigramProb(listOfBigrams, unigramCounts, bigramCounts)

    print("\nBigrams along with their probability")
    print(bigramProb)

    inputList = "This is my cat"
    splt = inputList.split()

    outputProb1 = 1
    bilist = []

    for i in range(len(splt)-1):
        bilist.append((splt[i], splt[i+1]))

    print("\nThe bigrams in given sentence are")
    print(bilist)

    for i in range(len(bilist)):
        if bilist[i] in bigramProb:
            outputProb1 *= bigramProb[bilist[i]]
        else:
            outputProb1 *= 0

    print('\nProbability of sentence "This is my cat" = ' +
str(outputProb1))
```

Output:

PROGRAM 8

Write a Python program to eliminate stopwords using NLTK.

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

nltk.download('stopwords')
nltk.download('punkt')

def remove_stopwords(text):
    words = word_tokenize(text)
    english_stopwords = set(stopwords.words('english'))
    filtered_words = [word for word in words if word.lower() not in
english_stopwords]
    filtered_text = ' '.join(filtered_words)
    return filtered_text

text = "NLTK is a leading platform for building Python programs to work
with human language data."

filtered_text = remove_stopwords(text)
print(filtered_text)
```

Output:

PROGRAM 9

Write a Python program to perform stemming using NLTK.

```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

nltk.download('punkt')

def stem_text(text):
    porter_stemmer = PorterStemmer()
    words = word_tokenize(text)
    stemmed_words = [porter_stemmer.stem(word) for word in words]
    stemmed_text = ' '.join(stemmed_words)
    return stemmed_text

text = "NLTK is a leading platform for building Python programs to work
with human language data."

stemmed_text = stem_text(text)
print(stemmed_text)
```

Output:

PROGRAM 10

Write a Python program to perform Named Entity Recognition (NER) using NLTK.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, ne_chunk

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

def ner(text):
    words = word_tokenize(text)
    tagged_words = pos_tag(words)
    named_entities = ne_chunk(tagged_words)
    return named_entities

text = "Apple is a company based in California, United States. Steve Jobs
was one of its founders."

named_entities = ner(text)
print(named_entities)
```

Output: