**NLP- (NATURAL LANGUAGE PROCESSING LAB MANUAL)**

**EXPERIMENT-1**

**Implementing a regular expression to parse all the plural words**

**Aim:** Write a program to implement a regular expression to parse all the plural words

**Code:**

import re

plural = 'foods'

match = re.search(r's$', plural)

if match is None:

    print("Parsing is not successful and is not a plural word")

else:

    print("Parsing is successful and is a plural word")

**Output:**

Parsing is successful and is a plural word

**Conclusion:**

Program to implement a regular expression to parse all the plural words is successful

**EXPERIMENT – 2**

**Split the text into words and remove the special characters and punctuations**

**Aim:** Write a program to split the text into words and remove the special characters and punctuations from the text.

**Code:**

punctuations = '''!()-[]{};:'"\,<>./?@#$%^&\*\_~'''

my\_str = "Hi!!!,..I am Roochita and my gmail is 221910310047@gitam.in."

no\_punct = ""

for char in my\_str:

   if char not in punctuations:

       no\_punct = no\_punct + char

print(no\_punct)

print(no\_punct.split())

**Output**:

['Hi', 'I', 'am', 'Roochita', 'and', 'my', 'gmail', 'is', '221910310047gitamin']

**Conclusion:**

A program to split the text into words and remove the special characters and punctuations from the text is successful.

**EXPERIMENT – 3**

**Stemming and Lemmatization**

**Aim:** Write a program for stemming and lemmatization of the root word of the words present in a text.

**Code:**

**Stemming**

from nltk.stem import PorterStemmer

from nltk.tokenize import sent\_tokenize, word\_tokenize

words = ["gitam","university","sunflower","beautiful"]

ps = PorterStemmer()

for w in words:

  rootWord = ps.stem(w)

  print(rootWord)

**Output:**

gitam

univers

sunflow

Beauti

**Lemmatization**

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import nltk

ltk.download("wordnet")

nltk.download("omw-1.4")

lemmatizer.lemmatize("foods")

**Output:**

True

Food

**Conclusion:**

Performed stemming and lemmatization of the root word of the words present in text.

**EXPERIMENT – 4**

**Bi-Gram Model and Add-one Smoothing**

**Aim:** Write a program to build a bi-gram model from the words present in the text. Apply the add-one smoothing to smooth the probability distribution of the bi-gram model.

**Code:**

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]

        word2 = bigram[1]

        listOfProb[bigram] =(bigramCounts.get(bigram))/(unigramCounts.get(word1))

    return listOfProb

if \_\_name\_\_ == '\_\_main\_\_':

    data = readData()

    listOfBigrams, unigramCounts, bigramCounts = createBigram(data)

    print("\n All the possible Bigrams are ")

    print(listOfBigrams)

    print("\n Bigrams along with their frequency ")

    print(bigramCounts)

    print("\n Unigrams along with their frequency ")

    print(unigramCounts)

    bigramProb = calcBigramProb(listOfBigrams, unigramCounts, bigramCounts)

    print("\n Bigrams along with their probability ")

    print(bigramProb)

    inputList="This is my cat"

    splt=inputList.split()

    outputProb1 = 1

    bilist=[]

    bigrm=[]

    for i in range(len(splt) - 1):

        if i < len(splt) - 1:

            bilist.append((splt[i], splt[i + 1]))

    print("\n The 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('\n' + 'Probablility of sentence \"This is my cat\" = ' + str(outputProb1))

**Output:**

['This', 'is', 'a', 'dog', 'This', 'is', 'a', 'cat', 'I', 'love', 'my', 'cat', 'This', 'is', 'my', 'name']

All the possible Bigrams are

[('This', 'is'), ('is', 'a'), ('a', 'dog'), ('This', 'is'), ('is', 'a'), ('a', 'cat'), ('I', 'love'), ('love', 'my'), ('my', 'cat'), ('This', 'is'), ('is', 'my'), ('my', 'name')]

Bigrams along with their frequency

{('This', 'is'): 3, ('is', 'a'): 2, ('a', 'dog'): 1, ('a', 'cat'): 1, ('I', 'love'): 1, ('love', 'my'): 1, ('my', 'cat'): 1, ('is', 'my'): 1, ('my', 'name'): 1}

Unigrams along with their frequency

{'This': 3, 'is': 3, 'a': 2, 'dog': 1, 'cat': 2, 'I': 1, 'love': 1, 'my': 2}

Bigrams along with their probability

{('This', 'is'): 1.0, ('is', 'a'): 0.6666666666666666, ('a', 'dog'): 0.5, ('a', 'cat'): 0.5, ('I', 'love'): 1.0, ('love', 'my'): 1.0, ('my', 'cat'): 0.5, ('is', 'my'): 0.3333333333333333, ('my', 'name'): 0.5}

The bigrams in given sentence are

[('This', 'is'), ('is', 'my'), ('my', 'cat')]

Probablility of sentence "This is my cat" = 0.16666666666666666

**Conclusion:**

Built a Bi-Gram model from the words present in the text.

Applied the add-one smoothing to smooth the probability distribution of the bi-gram model.

**N-Gram Model**

**Aim:** Write a program to build a N-Gram model from the words present in the text.

**Code:**

# Method 1

from nltk import ngrams

gvn\_str = "Hello Welcome to Python Programs"

gvn\_n\_val = 2

splt\_lst = gvn\_str.split()

rslt\_n\_grms = ngrams(splt\_lst, gvn\_n\_val)

for itr in rslt\_n\_grms:

    print(itr)

**Output:**

('Hello', 'Welcome')

('Welcome', 'to')

('to', 'Python')

('Python', 'Programs')

# Method 2

from nltk import ngrams

gvn\_str = input("Enter some random string = ")

gvn\_n\_val = int(input("Enter some random number(n) = "))

splt\_lst = gvn\_str.split()

rslt\_n\_grms = ngrams(splt\_lst, gvn\_n\_val)

for itr in rslt\_n\_grms:

    print(itr)

**Output:**

Enter some random string = qwwusu xfycxgv xyufqv

Enter some random number(n) = 3

('qwwusu', 'xfycxgv', 'xyufqv')

**Conclusion:**

Built a N-Gram model from the words present in the text.

**EXPERIMENT – 5**

**HMM Parts of Speech Tagging**

**Aim:** Write to program for constructing an HMM part of speech tagger.

**Code:**

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize, sent\_tokenize

stop\_words = set(stopwords.words('english'))

txt = "The Natural Language Toolkit NLTK is a platform used for building programs for text analysis. One of the more powerful aspects of the NLTK module is the Part of Speech tagging."

tokenized = sent\_tokenize(txt)

for i in tokenized:

    wordsList = nltk.word\_tokenize(i)

    wordsList = [w for w in wordsList if not w in stop\_words]

    tagged = nltk.pos\_tag(wordsList)

    print(tagged)

**Output:**

[('The', 'DT'), ('Natural', 'NNP'), ('Language', 'NNP'), ('Toolkit', 'NNP'), ('NLTK', 'NNP'), ('platform', 'NN'), ('used', 'VBN'), ('building', 'NN'), ('programs', 'NNS'), ('text', 'JJ'), ('analysis', 'NN'), ('.', '.')]

[('One', 'CD'), ('powerful', 'JJ'), ('aspects', 'NNS'), ('NLTK', 'NNP'), ('module', 'NN'), ('Part', 'NNP'), ('Speech', 'NNP'), ('tagging', 'NN'), ('.', '.')]

**Conclusion:**

Constructed an HMM part of Speech Tagger.

**EXTRA EXPERIMENTS**

* **Stemming**

**Aim:** Perform Stemming using Various Stemmers.

**Code:**

import nltk

from nltk.stem import PorterStemmer

from nltk.stem import LancasterStemmer

from nltk.stem import RegexpStemmer

from nltk.stem import SnowballStemmer

porter = PorterStemmer()

words = ['Connects','Connecting','Connections','Connected','Connection','Connectings','Connect']

for word in words:

    print(word,"--->",porter.stem(word))

**Output:**

Connects ---> connect

Connecting ---> connect

Connections ---> connect

Connected ---> connect

Connection ---> connect

Connectings ---> connect

Connect ---> connect

snowball = SnowballStemmer(language='english')

words = ['generous','generate','generously','generation']

for word in words:

    print(word,"--->",snowball.stem(word))

**Output:**

generous ---> generous

generate ---> generat

generously ---> generous

generation ---> generat

lancaster = LancasterStemmer()

words = ['eating','eats','eaten','puts','putting']

for word in words:

    print(word,"--->",lancaster.stem(word))

**Output:**

eating ---> eat

eats ---> eat

eaten ---> eat

puts ---> put

putting ---> put

regexp = RegexpStemmer('ing$|s$|e$|able$', min=4)

words = ['mass','was','bee','computer','advisable']

for word in words:

    print(word,"--->",regexp.stem(word))

**Output:**

mass ---> mas

was ---> was

bee ---> bee

computer ---> computer

advisable ---> advis

**Conclusion:**

Performed Stemming using various Stemmers.

* **Counting Length, no of words, no of spaces in a text**

**Aim:** Write a program to count the length of text, Number of words in a text and Number of spaces in a text

**Code:**

st = input("Enter String : ")

print(st)

**Output:**

Enter String : sdfghj w yug e66tugu8

sdfghj w yug e66tugu8

#char spaces

count = 0

for i in st:

    if i == " ":

        count += 1

print(count)

**Output:**

3

#no of words

stk = []

s = ""

for i in st:

    if i==" ":

        stk.append(s)

        s=""

        continue

if s:

    stk.append(s)

print(stk)

**Output:**

['', '', '']

4

#no of chars

print("Total number of chars : ",len(st))

**Output:**

Total number of chars : 21

* **Pull a random string from a text file**

**Aim:** Write a program to pull a random string from a text file.

**Code:**

import random

f = open('a.txt','r')

txt = f.read()

r = random.randint(0,len(txt))

r1 = random.randint(0,r)

print(txt[r1:r])

f.close()

**Output:**

dcuk cwtqvxg vwuqvukxvd vxuqvwyx

**Conclusion:**

Pulled a random string from a text file.