**Program-1**

**Write a program to find all words that occur at least three times in the Brown Corpus.**

import nltk

from nltk.corpus import brown

from collections import Counter

def find\_common\_words(corpus, min\_occurrences):

words = [word.lower() for word in corpus.words()]

word\_counts = Counter(words)

common\_words = [word for word, count in word\_counts.items() if count >= min\_occurrences]

return common\_words

def main():

nltk.download('brown')

min\_occurrences = 3

common\_words = find\_common\_words(brown, min\_occurrences)

print(f"Words that occur at least {min\_occurrences} times in the Brown Corpus:")

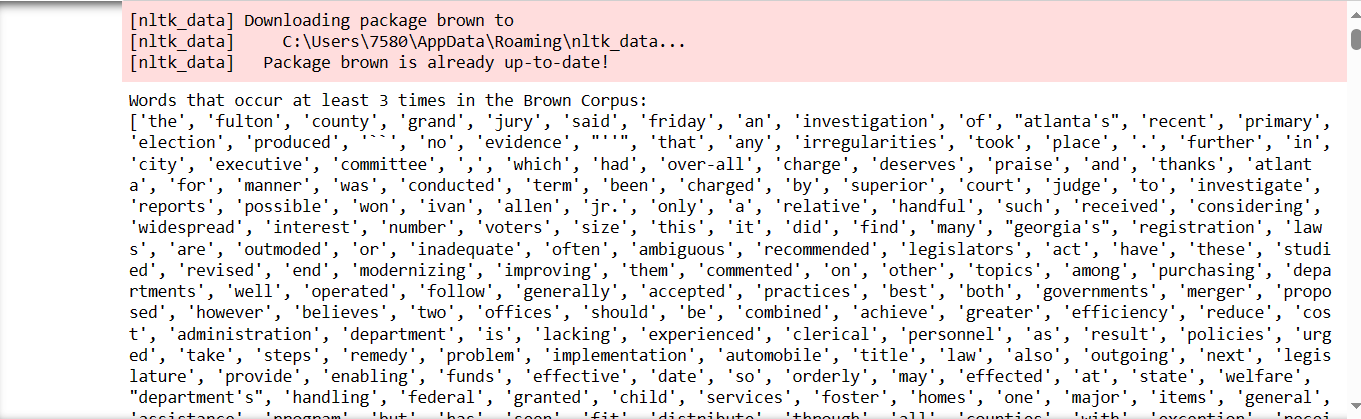
print(common\_words)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT**

**Jupyter Notebook**



**Program-2**

**Write a function that finds the 50 most frequently occurring words of a text that are not stopwords.**

import nltk

from nltk.corpus import stopwords

from nltk import FreqDist

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

nltk.download('stopwords')

def most\_frequent\_words(text):

# Tokenize the text into words

words = word\_tokenize(text)

# Filter out stopwords

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

filtered\_words = [word.lower() for word in words if word.isalpha() and word.lower() not in stop\_words]

# Calculate the frequency distribution of words

freq\_dist = FreqDist(filtered\_words)

# Get the 50 most frequent words

most\_frequent = freq\_dist.most\_common(50)

return most\_frequent

# Example usage:

sample\_text = "This is an example text. It contains some words, and some of these words may repeat. This is just an example."

result = most\_frequent\_words(sample\_text)

# Display the result

print("50 most frequent words (excluding stopwords):")

print(result)

**OUTPUT**

**Jupyter Notebook**

A close-up of a computer screen

Description automatically generated

**Program 3**

**Write a program to print the 50 most frequent bigrams (pairs of adjacent words) of a text, omitting bigrams that contain stopwords. Windows/Linux OS, IDE, Jupyter. Create and perform frequent bigram operations.**

import nltk

from nltk.corpus import stopwords

from nltk import FreqDist

from nltk import bigrams

# Download NLTK resources (stopwords)

nltk.download('stopwords')

def get\_frequent\_bigrams(text, N=50):

# Tokenize the text into words

words = nltk.word\_tokenize(text)

# Create a list of English stopwords

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

# Filter out stopwords and create bigrams

filtered\_bigrams = [bigram for bigram in bigrams(words) if all(word.lower() not in stop\_words for word in bigram)]

# Calculate the frequency distribution of bigrams

freq\_dist = FreqDist(filtered\_bigrams)

# Get the N most common bigrams

most\_common\_bigrams = freq\_dist.most\_common(N)

return most\_common\_bigrams

# Example text

example\_text = """

*Lakshadweep, the group of 36 islands is known for its exotic and sun-kissed beaches and lush green landscape. The name Lakshadweep in Malayalam and Sanskrit means ‘a hundred thousand islands’. India’s smallest Union Territory Lakshadweep is an archipelago consisting of 36 islands with an area of 32 sq km. It is a uni-district Union Territory and comprises of 12 atolls, three reefs, five submerged banks and ten inhabited islands. The islands have a total area of 32 sq km. The capital is Kavaratti and it is also the principal town of the UT. All Islands are 220 to 440 km away from the coastal city of Kochi in Kerala, in the emerald Arabian Sea. he natural landscapes, the sandy beaches, abundance of flora and fauna and the absence of a rushed lifestyle enhance the mystique of Lakshadweep.*

"""

# Call the function and print the 50 most frequent bigrams omitting stopwords

result = get\_frequent\_bigrams(example\_text, N=50)

for bigram, frequency in result:

print(bigram, frequency)

**OUTPUT**

**Jupyter Notebook**

**A screenshot of a computer code

Description automatically generated**

**Program-4**

**Write a function word\_freq() that takes a word and the name of a section of the Brown Corpus as arguments, and computes the frequency of the word in that section of the corpus.**

import nltk

from nltk.corpus import brown

def word\_freq(word, genre):

fdist = nltk.FreqDist([w.lower() for w in nltk.corpus.brown.words(categories=genre)])

print(f"The word '{word}' appears {fdist[word]} times in the '{genre}' genre of the Brown Corpus.")

# Example usage:

word\_freq('the', 'religion')

**OUTPUT**

**Jupyter Notebook**

****

**Program-5**

**Define a string s = 'colorless'. Write a Python statement that changes this to “colourless” using only the slice and concatenation operations. Windows/Linux OS, IDE, Jupyter Create and perform slicing operation.**

# Define the original string

s = 'colorless'

modified\_s = s[:3] + 'our' + s[5:]

# Print the modified string

print(modified\_s)

**OUTPUT**

**Jupyter Notebook**

****

**Program-6**

Read in some text from a corpus, tokenize it, and print the list of all wh-word types that occur. (Wh-words in English are used in questions, relative clauses and exclamations: who, which, what, and so on.) Print them in order. Are any words duplicated in this list, because of the presence of case distinctions or punctuation.

import nltk

from nltk import word\_tokenize

f = open('corpus.txt')

raw = f.read()

# tokenizes that text

tokens = word\_tokenize(raw)

# pulls out all the words that start with wh. and prints it out.

wh\_words = [word for word in tokens if word.startswith('wh')]

# sorts the list and prints it

wh\_words.sort()

print(wh\_words)

**OUTPUT**

**Jupyter Notebook**

**A pattern of black and white text

Description automatically generated**

**Program-7**

Write code that removes whitespace at the beginning and end of a string and normalizes whitespace between words to be a single space character.

# 1. do this task using split () and join ()

# 2. do this task using regular expression substitutions.

import re

sent = ' this is my sentence that starts and ends with whitespace '

pattern = re.compile('^\s|\s$')

sent = pattern.sub('', sent)

print(sent)

sent = " This is an example text with extra whitespace. "

sent = sent.split(' ')

sent = [word for word in sent if word]

sent = ' '.join(sent)

print(sent)

**OUTPUT**

**Jupyter Notebook**

****

**Program-8**

Write a python program to remove stop words for a given passage from a text file using NLTK. Implement stop word removal operation in an NLP application.

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

example\_sent = """This is a sample sentence,

showing off the stop words filtration."""

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

print(stopwords.words('english'))

word\_tokens = word\_tokenize(example\_sent)

filtered\_sentence = [w for w in word\_tokens if not w.lower() in stop\_words]

#with no lower case conversion

filtered\_sentence = []

for w in word\_tokens:

if w not in stop\_words:

filtered\_sentence.append(w)

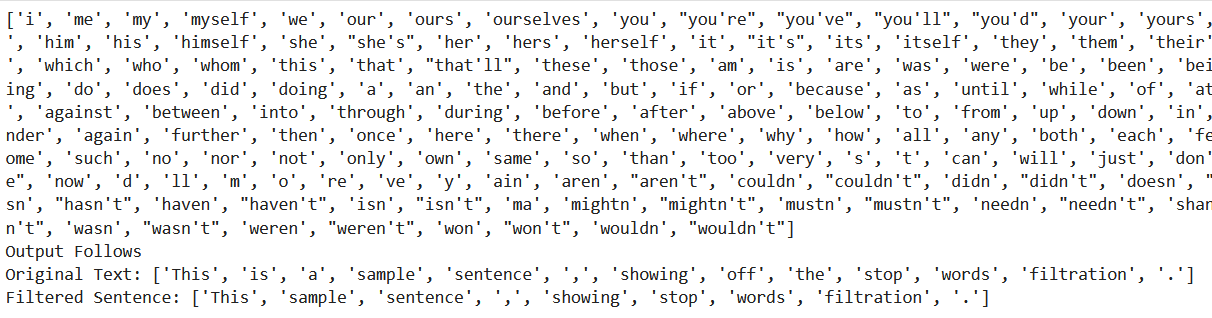
print("Output Follows")

print("Original Text:",word\_tokens)

print("Filtered Sentence:",filtered\_sentence)

**OUTPUT**

**Jupyter Notebook**



**Program-9**

Write a python program to implement stemming for a given sentence using NLTK? Windows/Linux OS, IDE, Jupyter Implement stemming operation in an NLP application.

import nltk

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

def apply\_stemming(sentence):

# Tokenize the sentence into words

words = word\_tokenize(sentence)

# Initialize the Porter Stemmer

porter\_stemmer = PorterStemmer()

# Apply stemming to each word

stemmed\_words = [porter\_stemmer.stem(word) for word in words]

# Join the stemmed words back into a sentence

stemmed\_sentence = ' '.join(stemmed\_words)

return stemmed\_sentence

def main():

# Example sentence

input\_sentence = "Stemming is an important step in natural language processing."

# Apply stemming to the sentence

stemmed\_result = apply\_stemming(input\_sentence)

# Display the results

print("Original Sentence:")

print(input\_sentence)

print("\nStemmed Sentence:")

print(stemmed\_result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT**

**Jupyter Notebook**

A white background with black text

Description automatically generated

**Program-10**

**Write a python program to POS (Parts of Speech) tagging for the give sentence using NLTK. Implement POS tagging operation in an NLP application.**

import nltk

from nltk import pos\_tag

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

def pos\_tagging(sentence):

# Tokenize the sentence into words

words = word\_tokenize(sentence)

# Remove stopwords (optional, depending on your use case)

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

words = [word for word in words if word.lower() not in stop\_words]

# Perform POS tagging

pos\_tags = pos\_tag(words)

return pos\_tags

def main():

# Example sentence

input\_sentence = "NLTK is a powerful library for natural language processing."

# Perform POS tagging on the sentence

pos\_tags\_result = pos\_tagging(input\_sentence)

# Display the results

print("Original Sentence:")

print(input\_sentence)

print("\nPOS Tags:")

print(pos\_tags\_result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT**

**Jupyter Notebook**

A close-up of a computer screen

Description automatically generated

**Program-11**

**Write a python program to POS (Parts of Speech) tagging for the give sentence using NLTK. Implement POS tagging operation in an NLP application**

import nltk

nltk.download('wordnet')

from nltk.corpus import wordnet

from nltk.stem import WordNetLemmatizer

def get\_wordnet\_pos(word):

"""Map POS tag to first character lemmatize() accepts"""

tag = nltk.pos\_tag([word])[0][1][0].upper()

tag\_dict = {"J": wordnet.ADJ,

"N": wordnet.NOUN,

"V": wordnet.VERB,

"R": wordnet.ADV}

return tag\_dict.get(tag, wordnet.NOUN)

# 1. Initialize the Lemmatizer

lemmatizer = WordNetLemmatizer()

# 2. Lemmatize Single Word with the appropriate POS tag

word = 'feet'

print(lemmatizer.lemmatize(word, get\_wordnet\_pos(word)))

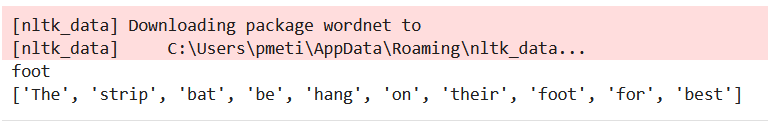
# 3. Lemmatize a Sentence with the appropriate POS tag

sentence = "The striped bats are hanging on their feet for best"

print([lemmatizer.lemmatize(w, get\_wordnet\_pos(w)) for w in nltk.word\_tokenize(sentence)])

**OUTPUT**

**Jupyter Notebook**



**Program-12**

**Write a python program to for Text Classification for the give sentence using NLTK. Implement Text Classification operation in an NLP application**

import nltk

from nltk.corpus import movie\_reviews

from nltk import FreqDist

from nltk import classify

from nltk import NaiveBayesClassifier

# Download the movie\_reviews dataset if not already downloaded

nltk.download('movie\_reviews')

# Feature extraction function

def extract\_features(words):

return dict(FreqDist(words))

# Prepare the dataset

documents = [(list(movie\_reviews.words(fileid)), category)

for category in movie\_reviews.categories()

for fileid in movie\_reviews.fileids(category)]

# Shuffle the documents

import random

random.shuffle(documents)

# Split the dataset into training and testing sets

split\_ratio = int(len(documents) \* 0.8)

train\_set, test\_set = documents[:split\_ratio], documents[split\_ratio:]

# Extract features using the defined function

training\_features = [(extract\_features(words), category) for (words, category) in train\_set]

testing\_features = [(extract\_features(words), category) for (words, category) in test\_set]

# Train the Naive Bayes classifier

classifier = NaiveBayesClassifier.train(training\_features)

# Evaluate the classifier on the testing set

accuracy = classify.accuracy(classifier, testing\_features)

print("Accuracy:", accuracy)

# Example sentences to classify

new\_sentences = [

"This movie was fantastic!",

"I didn't like the plot of this film.",

"The acting was superb in this movie.",

"The screenplay was terrible."

]

# Classify the new sentences

for sentence in new\_sentences:

words = nltk.word\_tokenize(sentence)

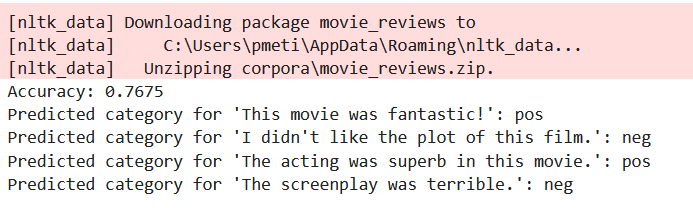
features = extract\_features(words)

category = classifier.classify(features)

print(f"Predicted category for '{sentence}': {category}")

**OUTPUT**

**Jupyter Notebook**

****

Rubrics

Program or Lab internals – 13

CIA – 8

* P – 3
* R – 2
* V – 2
* A - 1

Lab Internal - 5

Mini project – 12

* Execution - 6
* Publication/Project Competition - 6