EX.NO.1

07.01.26

ADDITIONAL PROGRAMS

BASIC PROGRAMS OF NLP

(A) AIM:

To write a python program to perform tokenisation by word and sentence using NLTK

Algorithm:

Step1 : Start

Step2 ; Import the required libraries

Step3 ; Define a function tokensize_sentence (text)

Step 4 ; Define a Multi-line text Input

Step 5 ; Call the function and store the returned Sentences in a variable.

Step 6 : Iterate through the list of sentences using a loop.

Step 7 : Print each Sentences with its sentence number.

Step 8 : Stop.

Program:

```
import nltk
nltk.download("punkt_tab")
from nltk.tokenise import sent_tokenize
def tokenize_sentences(text):
    sentences = sent_tokenise(text)
    return sentences
```

OUTPUT:

(1)
1) "NLTK is a leading platform for building python programs to work with human language data."

2) "It provides easy to use interface to over 50 corpora and lexical resources such as WordNet."

(2)
"NLTK", is, a, leading, platform, for, building, python, program, to, work, with, human, language, data

```
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 word Net """
sentences = tokenise - sentences (text)
for i, sentence in enumerate (sentences):
    print (f" sentence {i+1} : {sentence} ")
```

(2)

```
import nltk
nltk.download ("punkt")
from nltk.tokenize import word_tokenize
def tokenize_words (text):
    words = word_tokenize (text)
    return words
text = "NLTK is a leading platform for
        building python programs to work with
        human language data"
words = tokenize_words (text)
print (words)
```

Result:

Thus the program to perform tokenization by word and sentences executed successfully.

(B)

AIM:

To Write a python program to eliminate Stopwords using NLTK

ALGORITHM:

Step 1: Start

Step 2: Import the NLTK library

Step 3: Import stopwords from nltk.corpus.

Step 4: Define the Input text

Step 5: Create a function remove_stopwords()

Step 6: Call the function with the Input text

Step 7: Store the Outputs and then display it

Step 8: Stop.

Program:

```
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"))
```

Output:

NLTK leading platform building

python programs work human langu

```
    filtered_words = [words for words in words if
                      words.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"
filtered_text = remove_stopwords(text)
print(filtered_text)
```

Result:

Hence the python program for removing stopwords executed successfully.

Output:

NLTK is a lead platform for build

python program to work with human langu

(c)

Aim:

To write a python program to perform stemming using NLTK.

Algorithm:

Step 1: start

Step 2: Import required libraries

Step 3: Define a function stem_text(text):

Step 4: Call the function with input text

Step 5: Display the output

step 6: Stop.

Program:

```
import ~~NLTK~~ nltk
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".
stemmed_text = stem_text(text)
print(stemmed_text)
```

Result: Hence the python program for stemming executed successfully.

(10)

AIM:

To Write a python program for "Tokenization, filteration, Script validation, Stopword Removal, Stemming the given code".

Algorithm:

Step 1: Start

Step 2: Read the Input text

Step 3: Tokenize the text into words

Step 4: Remove special characters & validate english words.

Step 5: Remove stopwords and apply stemming.

Step 6: Display the final pre-processed tokens and stop.

Program:

```
import re, nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
nltk.download("punkt")
nltk.download("stopwords")
text = "Natural Language Processing is a fascinating
        field of AI, it involves processing
        human language, like English!."
def tokenize(text):
    return words_tokenize(text)
def filter_text(tokens):
    filtered_tokens = [re.sub('[^|w|s]', '',
                      token) for token in tokens]
    filtered_tokens
```

Output:

Original: Natural Language processing is a fascinating field of AI. It involves processing like English.

Tokenized: (1) Natural Language Processing is a fascinating field of AI
(2) It involves processing like English.

filtered token:

['Natural', 'language', 'processing', 'is', 'a', 'fascinating', 'field', 'of', 'AI', 'It', 'involves', 'processing', 'like', 'english']

Validate Tokens: ['Natural', 'language', 'processing', 'is', 'a', 'fascinating', 'field', 'of', 'AI', 'It', 'involves', 'processing', 'English']

Tokens after stopwords removal: ['Natural', 'Language', 'processing', 'fascinating', 'field', 'involves', 'processing', 'English']

Stemmed tokens:

['Natur', 'languag', 'process', 'fascin', 'field', 'ai', 'involv', 'process', 'like', 'English'].

xl Language processing is a
ing field of AI. It involves
like English.

ural Language Processing is
ting field of AI
ues processing like English

ge', 'processing', 'is', 'a
', 'AI', 'It', 'involves
lish',]
l', 'language', 'processing',
ld', 'of', 'AI', 'It',
sh']
oval: ['Natural',
ssing', 'field', 'AI',
nglish',]
ocess', 'text' 'fascin',
'process', 'like',

```
def validate_script(tokens):
    return [token for token in tokens if re.match
                (r'^[a-z A-Z]+$', token)]
def remove_stopwords(tokens):
    stop_words = set(stopwords.words("english"))
    return ("token for token in tokens if token.
                lower() not in stop_words]
def stem_tokens(tokens):
    stemmer = PorterStemmer()
    return [stemmer.stem(token) for token in tokens].

def Preprocess_text(text):
    print("Original", text)
    print("tokenized", tokenize(text))
    print("Filtered tokens", filter_text(tokens))
    print("Validate Tokens", Validated_script
                (filtered_token))
    print("Tokens after stopwords removal",
                token_no_stopwords}
    print("Stemmed text", Stemmed_tokens")

preprocessed = preprocess_text(text)
```

Result: Hence the python program for pre-processing a text executed successfully.