Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai-401208
Accredited B+ Grade by NAAC

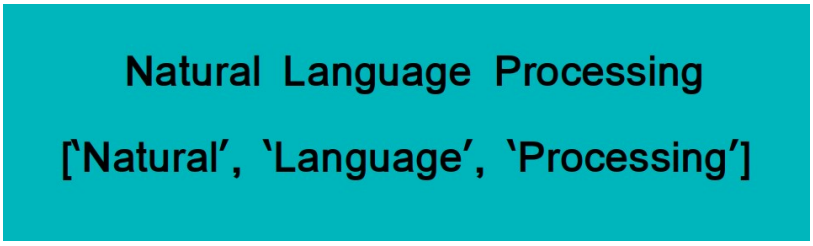
Experiment 2

Name of the Student: - Sanket Belekar
Roll No.: 67 Batch: A3 Branch: TE COMPS-A
Date of Practical Performed: - Staff Signature with Date & Marks

Aim: Write a Program to perform Tokenization and Filtration & Script Validation.

Theory:

Tokenization: Tokenization is one of the most common tasks when it comes to working with text data. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.



The tokens could be words, numbers or punctuation marks. In tokenization, smaller units are created by locating word boundaries. Wait – what are word boundaries?

These are the ending point of a word and the beginning of the next word. These tokens are considered as a first step for stemming and lemmatization

Before processing a natural language, we need to identify the words that constitute a string of characters. That's why tokenization is the most basic step to proceed with NLP (text data). This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

Let's take an example. Consider the below string:

"This is a cat."

What do you think will happen after we perform tokenization on this string? We get ['This', 'is', 'a', cat']. There are numerous uses of doing this. We can use this tokenized form to:

- Count the number of words in the text
- Count the frequency of the word, that is, the number of times a particular word is present

And so on. We can extract a lot more information which we'll discuss in detail in future articles. For now, it's time to dive into the meat of this article – the different methods of performing tokenization in NLP.

Filtering:
Filtering is the process of removing stop words or any unnecessary data from the sentence. With our text split into a stream of tokens, we have the ability to start filtering out any tokens that we might not find helpful for our application. In the Text Pre- processing tool, we currently have the option to filter out digit, punctuation, and stop word tokens (we address stop words in the next section).

Stop-words are words that don't necessarily add meaning to a body of text but are necessary for the text to be grammatically correct. For example, words like "the" or "a" these words aren't providing much information, but are important for understanding.

Python's split function: This is the most basic one, and it returns a list of strings after splitting the string based on a specific separator. The separators can be changed as needed. Sentence Tokenization: Here, the structure of the sentence is analyzed. As we know, a sentence ends with a period(.); therefore, it can be used as a separator.

● Word Tokenizer: It works similarly to a sentence tokenizer. Here the text is split up into token based on (' ') as the separator. We give nothing as the parameter it splits by space by default.

Tokenization:

Code:

```
import nltk
from nltk.tokenize import word_tokenize

def tokenize_script(text):
  tokens = word_tokenize(text)
  return tokens

text = "Tokenization is an important step in natural language processing."
tokens=tokenize_script(text)
print(tokens)
```

Output:

```
['Tokenization', 'is', 'an', 'important', 'step', 'in', 'natural', 'language', 'processing', '.']
```

Filtration:

Code:

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

def filter_text(text):
    tokens = word_tokenize(text)
    stop_word = set(stopwords.words('english'))
    filtered_text = [token for token in tokens if token.lower() not in
stop_word and len(token) > 1]
    return filtered_text #Added return statement to return the filtered
tokens

text = "Tokenization is an important step in natural language processing."
filtered_tokens = filter_text(text)
print(filtered_tokens)
```

Output:

```
['Tokenization', 'important', 'step', 'natural', 'language', 'processing']
```

Script Validation:

Code:

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

def validate_script(script):
  tokens = word_tokenize(script)
  stop_words = set(stopwords.words('english'))
  filtered_tokens = [token for token in tokens if token.lower() not in
stop_words and len(token) > 1]

  min_num_words = 5

  if len(filtered_tokens) >= min_num_words:
    print(f"Script '{script}'")
    return True
  else:
    print(f"Script '{script}'")
    return False

script1= "this is a short script."
script2= "aim : write a program to perform tokenization and script
validation in python."
```

```
script3= "python program to test script."

print(validate_script(script1))
print(validate_script(script2))
print(validate_script(script3))
```

Output:

```
Script 'this is a short script.'
False
Script 'aim : write a program to perform tokenization and script validation in python.'
True
Script 'python program to test script.'
False
```

Conclusion: -

We have thus examined the idea of tokenization and filtering. Each word in the statement was tokenized, and the specific alphabet was then taken out of the statement during filtering.