Department of Computer Engineering
Class: B.E. (Computer) (Div- B)
(Sem - VIII)

Subject: Computational Lab-II (NLP)(CSL804)

Sr. No.	Title of Experiment
2.	**To implement Pre-processing of text using following steps:** **Tokenization, Filtration, Script Validation, Stop Word Removal, Stemming**

Course Name: Computational Lab-II(NLP)

Course Code: CSL804

Experiment No.: 02

Lab outcome: Acquire practical knowledge within the chosen area of technology for project development.

Name of Student: Moin Memon

Student Roll No.:275

Year/Semester/Div: B.E./VIII/B

Experiment No. 02

Aim: To implement Pre-processing of text using following steps:
Tokenization, Filtration, Script Validation, Stop Word Removal, Stemming

Theory:

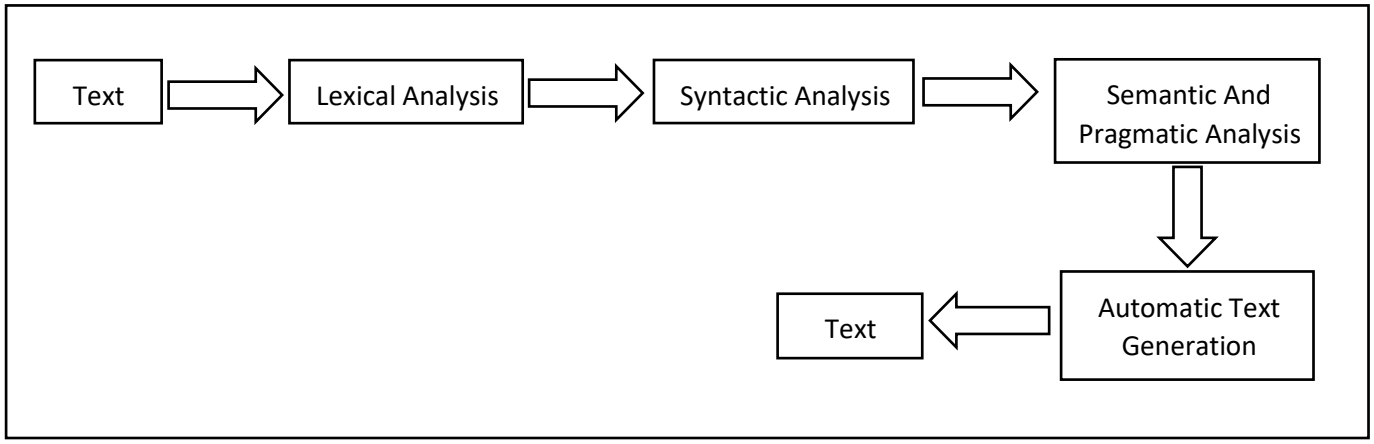
Pre-processing:

Natural Language Processing (NLP) is a subfield of computer science, artificial intelligence, information engineering, and human-computer interaction. This field focuses on how to program computers to process and analyze large amounts of natural language data. It is difficult to perform as the process of reading and understanding languages is far more complex than it seems at first glance.For example, we can use NLP to create systems like speech recognition, document summarization, machine translation, spam detection, named entity recognition, question answering, autocomplete, predictive typing and so on.

Text preprocessing is a method to clean the text data and make it ready to feed data to the model. Text data contains noise in various forms like emotions, punctuation, text in a different case. When we talk about Human Language then, there are different ways to say the same thing, And this is only the main problem we have to deal with because machines will not understand words, they need numbers so we need to convert text to numbers in an efficient manner.

1.Tokenization:

Tokenization is the process of tokenizing or splitting a string, text into a list of tokens One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph. The five phases of NLP involve lexical (structure) analysis, parsing, semantic analysis. discourse integration, and pragmatic analysis. Tokenization is used in prior steps of nlp, in analysis.



Tokenization process Tokenization process.

It consists of two types Word tokenization and Sentence tokenization:

Word Tokenization:

Word tokenization becomes a crucial part of the text (string) to numeric data conversion. The method word _tokenize() to split a sentence into words. The output of word tokenization can be converted to Data Frame for better text understanding in machine learning applications. It can also be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming. Machine learning models need numeric data to be trained and make a prediction.

Command: **nltk.tokenize.word_tokenize(sentence) Sentence**

Tokenization:

An obvious question in your mind would be why sentence tokenization is needed when we have the option of word tokenization. Imagine we need to count average words per sentence, how you will calculate? For accomplishing such a task, you need both NLTK sentence tokenizer as well as NLTK word tokenizer to calculate the ratio. Such output serves as an important feature for machine training as the answer would be numeric. An Sub-module available for the above is sent_tokenize(). Sentence tokenizer plays an important role in machine training for the tokenization of sentence.

Command: **sent_tokenize(text))**

2.Filtration:

Filtering is the process of removing stop words or any unnecessary data from the sentence. It can also be defined as the process of removing other language components from the given input text.

Eg: " कौन हे class monitor ? "

The above statement consists of the statement which involves the words which are not be considered as input data. Post filtration the output would be as **class monitor?**

Command: **join(list(filter(lambda x: ord(x) < 123, sentence_mix)))**

3.Script Validation:

Script Validation is the process of removing special characters and punctuations. Special characters, as you know, are non-alphanumeric characters. These characters are most often found in comments, references, currency numbers etc. These characters add no value to text-understanding and induce noise into algorithms. Removing punctuation is fairly easy. It can be achieved by using **string.Punctuation** and keeping everything, which is not in this list.

4.Stop word removal:

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. These stop words carry minimal to no importance and are available plenty on open texts, articles, comments etc. These should be removed so machine learning algorithms can better focus on words which define the meaning/idea of the text.

Command: **stopwords = set(stopwords.words('english')) removed_stopwords = [w for w in tokenized_sent if not w in stopwords]**

5.Stemming:

Stemming is a kind of normalization where a set of words in a sentence are converted into a sequence to shorten its lookup. Due to grammatical reasons, language includes lots of variations in the sense that the language English as well as other languages too, have different forms of a word For example, the words like democracy democratic, and democratization. For machine learning projects, it is very important for machines to understand that these different words, like above, have the same base form That is why it is very useful to extract the base forms of the words while analyzing the text. Stemming is also be said as heuristic process that helps in extracting the base forms of the words by chopping of their ends.

The different packages for stemming provided by NLTK module are as follows – **i).**

PorterStemmer package

Porter's algorithm is used by this gemming package to extract the base form of the words. With the help of the following command, we can import this package- **from nltk stem porter import PorterStemmer**

For example, "write" would be the output of the word "writing" given as the input to this stemmer.

ii). LancasterStemumer package

Lancaster's algorithm is used by this stemming package to extract the base form of the words. With the help of following command, we can import this package – **from nltk stem lancaster import LancasterStemmer**

For example, "write" would be the output of the word "writing" given as the input to this stemmer **iii).**

Snow ballStemmer package

Snowball's algorithm is used by this stemming package to extract the base form of the words. With the help of following command, we can import this package- **from stem.snowball impon SnowballStemmer**

For example, "write" would be the output of the word "writing" given as the input to this stemmer **Program**

and its output:

Preprocessing libraries:

```
In [1]: import nltk
        nltk.download('punkt')
        nltk.download('wordnet')
        nltk.download('stopwords')
        from nltk.corpus import stopwords

        [nltk_data] Downloading package punkt to
        [nltk_data]     C:\Users\Rahul\AppData\Roaming\nltk_data...
        [nltk_data]   Package punkt is already up-to-date!
        [nltk_data] Downloading package wordnet to
        [nltk_data]     C:\Users\Rahul\AppData\Roaming\nltk_data...
        [nltk_data]   Package wordnet is already up-to-date!
        [nltk_data] Downloading package stopwords to
        [nltk_data]     C:\Users\Rahul\AppData\Roaming\nltk_data...
        [nltk_data]   Package stopwords is already up-to-date!
```

Tokenization:

Word tokenization:

```
In [2]: #sentence = input('Enter a sentence: ')
        sentence= "This is a test statement for tokenization."
        tokenized_sent = nltk.tokenize.word_tokenize(sentence)
        print(f'After Sentence Tokenization: {tokenized_sent}')

        After Sentence Tokenization: ['This', 'is', 'a', 'test', 'statement', 'for', 'tokenization', '.']
```

Sentence tokenization:

```
In [3]: from nltk.tokenize import sent_tokenize
        text = "This is a test statement for tokenization."
        print(sent_tokenize(text))

        ['This is a test statement for tokenization.']
```

Filtration:

```
In [9]: sentence_mix ="कौन हे  class monitor ?"
        print('After Filtration: ' + ''.join(list(filter(lambda x: ord(x) < 123, sentence_mix))))

        After Filtration:    class monitor ?
```

Script Validation:

```
In [10]: validated = []
         for w in removed_stopwords:
             validated.append(''.join([e for e in w if e.isalnum()]))
         print(f'After script validation: {validated}')

         After script validation: ['In', 'computing', '', 'term', 'text', 'processing', 'refers', 'theory', 'practice', 'automating', 'c
         reation', 'manipulation', 'electronic', 'text', '']
```

Stop word removal:

```
In [7]: stopwords = set(stopwords.words('english'))
        removed_stopwords = [w for w in tokenized_sent if not w in stopwords]
        print(f'After removing Stopwords: {removed_stopwords}')

        After removing Stopwords: ['In', 'computing', ',', 'term', 'text', 'processing', 'refers', 'theory', 'practice', 'automating',
        'creation', 'manipulation', 'electronic', 'text', '.']

In [8]: new_words= [word for word in removed_stopwords if word. isalnum()]
        print(new_words)

        ['In', 'computing', 'term', 'text', 'processing', 'refers', 'theory', 'practice', 'automating', 'creation', 'manipulation', 'el
        ectronic', 'text']
```

Stemming:

```
In [4]: stemmer = nltk.stem.PorterStemmer()
        print(f'After Stemming: {[stemmer.stem(x) for x in tokenized_sent]}')

        After Stemming: ['thi', 'is', 'a', 'test', 'statement', 'for', 'token', '.']

In [5]: tokenized_sent = nltk.tokenize.word_tokenize("In computing, the term text processing refers to the theory and practice of automat
        print(f'After Stemming: {[stemmer.stem(x) for x in tokenized_sent]}')

        After Stemming: ['In', 'comput', ',', 'the', 'term', 'text', 'process', 'refer', 'to', 'the', 'theori', 'and', 'practic', 'of',
        'autom', 'the', 'creation', 'or', 'manipul', 'of', 'electron', 'text', '.']
```

PortStemmer :

In [6]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
sentence="Recent researchs reported that the virus can remain viable for up to 72 hours on plastic and stainless steel and up to
words = word_tokenize(sentence)
ps = PorterStemmer()
for w in words:
    rootWord=ps.stem(w)
    print(rootWord)
```

```
recent
research
report
that
the
viru
can
remain
viabl
for
up
to
72
hour
on
plastic
and
stainless
steel
and
up
to
24
hour
on
cardboard
.
```

Conclusion: Thus,We have successfully implemented Pre-processing of text using NLP libraries.