# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

AY:2024-25

| | | | |
|---|---|---|---|
| **Class:** | **BE** | **Semester:** | **VII** |
| **Course Code:** | **CSDOL7011** | **Course Name:** | **Natural Language Processing** |

| | |
|---|---|
| **Name of Student:** | **Parth Raut** |
| **Roll No.:** | **40** |
| **Experiment No.:** | **3** |
| **Title of the Experiment:** | **Advanced Text Preprocessing: Stop Word Removal & Lemmatization** |
| **Date of Performance:** | |
| **Date of Submission:** | |

## Evaluation

| Performance Indicator | Max. Marks | Marks Obtained |
|---|---|---|
| Performance | 5 | |
| Understanding | 5 | |
| Journal work and timely submission | 10 | |
| Total | 20 | |

| Performance Indicator | Exceed Expectations (EE) | Meet Expectations (ME) | Below Expectations (BE) |
|---|---|---|---|
| Performance | 4-5 | 2-3 | 1 |
| Understanding | 4-5 | 2-3 | 1 |
| Journal work and timely submission | 8-10 | 5-8 | 1-4 |

Checked by

Name of Faculty :

Signature :

Date :

**Experiment 3**

**Aim:** Apply various other text preprocessing techniques for any given text: Stop Word Removal, Lemmatization / Stemming.

**Objective:** To write a program for Stop word removal from a sentence given in English and any Indian Language.

**Theory:**

The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

Stopwords are the most common words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document.

Stop Words: 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. We need to perform tokenization before removing any stopwords.

**Why do we need to Remove Stopwords?**

Removing stopwords is not a hard and fast rule in NLP. It depends upon the task that we are working on. For tasks like text classification, where the text is to be classified into different categories, stopwords are removed or excluded from the given text so that more focus can be given to those words which define the meaning of the text.

Here are a few key benefits of removing stopwords:

- On removing stopwords, dataset size decreases and the time to train the model also decreases
- Removing stopwords can potentially help improve the performance as there are fewer and only meaningful tokens left. Thus, it could increase classification accuracy
- Even search engines like Google remove stopwords for fast and relevant retrieval of data from the database

We can remove stopwords while performing the following tasks:

- Text Classification
    - Spam Filtering
    - Language Classification
    - Genre Classification
- Caption Generation
- Auto-Tag Generation

**Avoid Stopword Removal**

- Machine Translation
- Language Modeling
- Text Summarization
- Question-Answering problems

**Different Methods to Remove Stopwords**

1. **Stopword Removal using NLTK**

   NLTK, or the Natural Language Toolkit, is a treasure trove of a library for text preprocessing. It's one of my favorite Python libraries. NLTK has a list of stopwords stored in 16 different languages.

   You can use the below code to see the list of stopwords in NLTK:

```
import nltk
from nltk.corpus import stopwords
set(stopwords.words('english'))
```

2. **Stopword Removal using spaCy:**

   **spaCy** is one of the most versatile and widely used libraries in NLP. We can quickly and efficiently remove stopwords from the given text using SpaCy.

   It has a list of its own stopwords that can be imported as **STOP_WORDS** from the **spacy.lang.en.stop_words** class.

3. **Stopword Removal using Gensim**

**Gensim** is a pretty handy library to work with on NLP tasks. While pre-processing, gensim provides methods to remove stopwords as well. We can easily import the remove_stopwords method from the class gensim.parsing.preprocessing.

## Code:

### Library required

In [1]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\admin\appdata\local\programs\python\python37\lib\site-packages (3.6.2)
Requirement already satisfied: joblib in c:\users\admin\appdata\local\programs\python\python37\lib\site-packages (from nltk)
(1.0.0)
Requirement already satisfied: click in c:\users\admin\appdata\local\programs\python\python37\lib\site-packages (from nltk)
(7.1.2)
Requirement already satisfied: regex in c:\users\admin\appdata\local\programs\python\python37\lib\site-packages (from nltk)
(2021.4.4)
Requirement already satisfied: tqdm in c:\users\admin\appdata\local\programs\python\python37\lib\site-packages (from nltk)
(4.60.0)
WARNING: You are using pip version 22.0; however, version 23.2.1 is available.
You should consider upgrading via the 'c:\users\admin\appdata\local\programs\python\python37\python.exe -m pip install --upgr
ade pip' command.
```

### Text

In [2]:
```
text = 'TON 618 is a hyperluminous, broad-absorption-line, radio-loud quasar and Lyman-alpha blob located near the border of
```

In [3]:
```
text
```

Out[3]:
```
'TON 618 is a hyperluminous, broad-absorption-line, radio-loud quasar and Lyman-alpha blob located near the border of the co
nstellations Canes Venatici and Coma Berenices, with the projected comoving distance of approximately 18.2 billion light-yea
rs from Earth.'
```

### Stopwords

In [4]:
```
from nltk.corpus import stopwords
```

In [5]:
```
stop_words = stopwords.words('english')
```

In [6]:
```
from nltk.tokenize import word_tokenize
words = word_tokenize(text)
```

### Applying stop words

In [7]:
```
holder = list()
for w in words:
    if w not in set(stop_words):
        holder.append(w)
```

In [8]:
```
holder
```

Out[8]:
```
['TON',
 '618',
 'hyperluminous',
 ',',
 'broad-absorption-line',
 ',',
 'radio-loud',
 'quasar',
 'Lyman-alpha',
 'blob',
 'located',
 'near',
 'border',
 'constellations',
 'Canes',
 'Venatici',
 'Coma',
 'Berenices',
 ',',
```

### List Comprehension for stop words

In [9]:
```
holder = [w for w in words if w not in set(stop_words)]
print(holder)
```

```
['TON', '618', 'hyperluminous', ',', 'broad-absorption-line', ',', 'radio-loud', 'quasar', 'Lyman-alpha', 'blob', 'located',
'near', 'border', 'constellations', 'Canes', 'Venatici', 'Coma', 'Berenices', ',', 'projected', 'comoving', 'distance', 'appr
oximately', '18.2', 'billion', 'light-years', 'Earth', '.']
```

## Stemming

In [10]:
```
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer
```

In [11]:
```
porter = PorterStemmer()
snow = SnowballStemmer(language = 'english')
lancaster = LancasterStemmer()
```

In [12]:
```
words = ['play', 'plays', 'played', 'playing', 'player']
```

### Porter Stemmer

In [13]:
```
porter_stemmed = list()
for w in words:
    stemmed_words = porter.stem(w)
    porter_stemmed.append(stemmed_words)
```

In [14]:
```
porter_stemmed
```

Out[14]:
```
['play', 'play', 'play', 'play', 'player']
```

### Porter Stemmer List Comprehension

In [15]:
```
porter_stemmed = [porter.stem(x) for x in words]
print (porter_stemmed)
```

```
['play', 'play', 'play', 'play', 'player']
```

### Snowball Stemmer

In [16]:
```
snow_stemmed = list()
for w in words:
    stemmed_words = snow.stem(w)
    snow_stemmed.append(stemmed_words)
```

In [17]:
```
snow_stemmed
```

Out[17]:
```
['play', 'play', 'play', 'play', 'player']
```

### Snowball Stemmer List Comprehension

In [18]:
```
snow_stemmed = [snow.stem(x) for x in words]
print (snow_stemmed)
```

```
['play', 'play', 'play', 'play', 'player']
```

### Lancaster Stemmer

In [19]:
```
lancaster_stemmed = list()
for w in words:
    stemmed_words = lancaster.stem(w)
    lancaster_stemmed.append(stemmed_words)
```

```
In [20]: lancaster_stemmed

Out[20]: ['play', 'play', 'play', 'play', 'play']
```

Lancaster Stemmer List Comprehension

```
In [21]: lancaster_stemmed = [lancaster.stem(x) for x in words]
         print (lancaster_stemmed)

['play', 'play', 'play', 'play', 'play']
```

Lemmatization : This has a more expansive vocabulary than Stemming

```
In [22]: from nltk.stem import WordNetLemmatizer
         wordnet = WordNetLemmatizer()

In [23]: lemmatized = [wordnet.lemmatize(x) for x in words]

In [24]: lemmatized

Out[24]: ['play', 'play', 'played', 'playing', 'player']
```

**Conclusion:**

Stop word removal is an essential preprocessing step in natural language processing (NLP) that involves eliminating common words (e.g., "the," "is," "and") that carry little meaning in a given context. For Indian languages, the process can be more challenging due to the diversity and complexity of languages. Here are some commonly used tools and methods for stop word removal in Indian languages, along with the steps involved:

Tools for Stop Word Removal in Indian Languages:

1. NLTK: While primarily focused on English, NLTK offers stop word lists for some Indian languages and can be customized with user-defined stop word lists.
2. spaCy: This library supports multiple languages and can be extended with custom stop word lists for Indian languages.
3. Indic NLP Library: A dedicated library for Indian languages that includes functionalities for tokenization, stop word removal, and more, specifically tailored for languages like Hindi, Bengali, Tamil, etc.
4. Gensim: A Python library for topic modeling that includes options for removing stop words. It can be customized with language-specific stop word lists.
5. Custom Lists: Creating tailored stop word lists for specific Indian languages based on domain requirements can significantly improve the effectiveness of stop word removal.

Steps Involved in Stop Word Removal:

1. Text Input: Start with the raw text data in the target Indian language.
2. Tokenization: Break down the text into individual tokens (words) using a suitable tokenizer that can handle the complexities of the language.
3. Stop Word List: Use an existing stop word list specific to the target language, or create a custom list based on the context of your application.
4. Filtering Tokens: Iterate through the tokenized words and remove those that are present in the stop word list. This can be done using set operations for efficiency.
5. Output Processed Text: Collect the filtered tokens back into a single string or a list, which can then be used for further NLP tasks, such as text classification or sentiment analysis.