Experiment No.3
Apply Stop Word Removal on given English and Indian Language Text
Date of Performance:
Date of Submission:

Aim: Apply Stop Word Removal on given English and Indian Language Text.

Objective: To write 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 stop words 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.

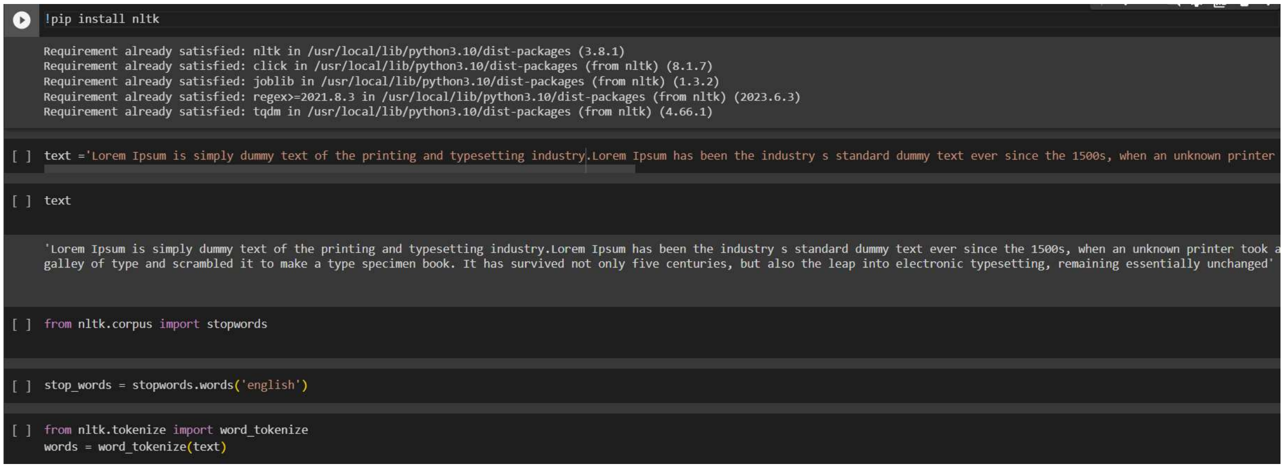
   You can use the below code to see the list of stopwords in NLTK:

   ```
   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:



```
!pip install nltk

Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
[ ] text ='Lorem Ipsum is simply dummy text of the printing and typesetting industry.Lorem Ipsum has been the industry s standard dummy text ever since the 1500s, when an unknown printer
```

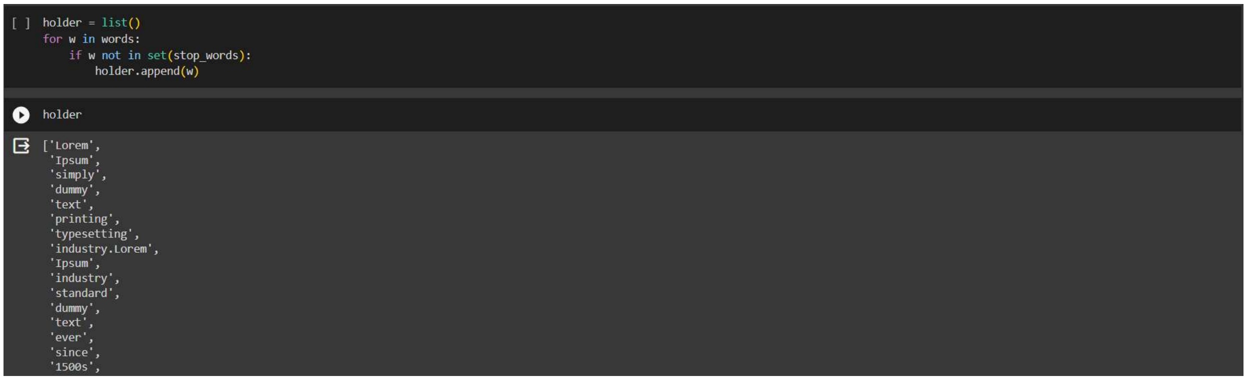
```
[ ] text

'Lorem Ipsum is simply dummy text of the printing and typesetting industry.Lorem Ipsum has been the industry s standard dummy text ever since the 1500s, when an unknown printer took a
galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged'
```

```
[ ] from nltk.corpus import stopwords
```

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

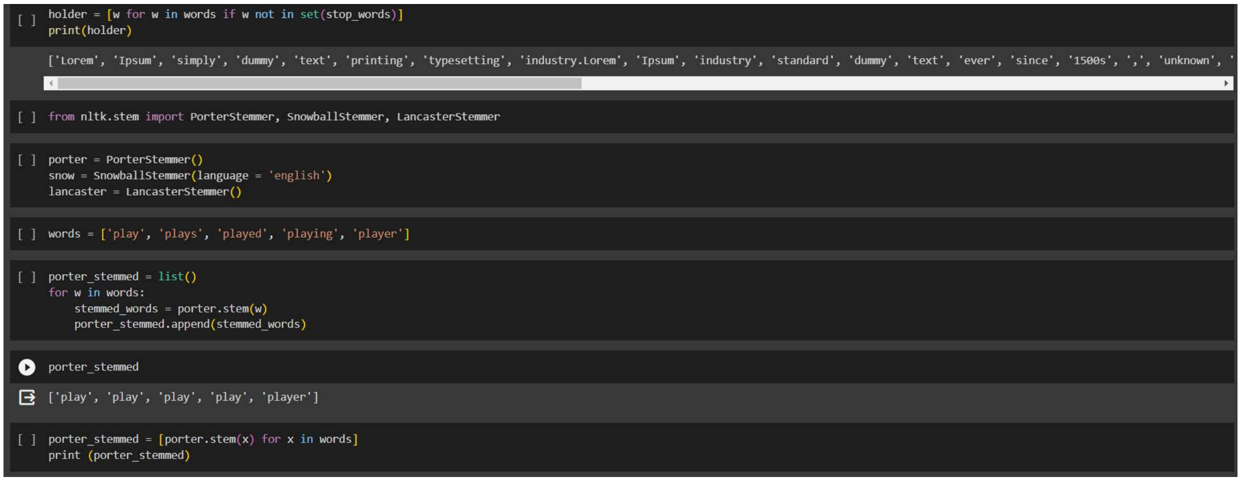
```
[ ] from nltk.tokenize import word_tokenize
    words = word_tokenize(text)
```



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

```
holder
```

```
['Lorem',
 'Ipsum',
 'simply',
 'dummy',
 'text',
 'printing',
 'typesetting',
 'industry.Lorem',
 'Ipsum',
 'industry',
 'standard',
 'dummy',
 'text',
 'ever',
 'since',
 '1500s',
```



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

```
['Lorem', 'Ipsum', 'simply', 'dummy', 'text', 'printing', 'typesetting', 'industry.Lorem', 'Ipsum', 'industry', 'standard', 'dummy', 'text', 'ever', 'since', '1500s', ',', 'unknown', '
```

```
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer
```

```
porter = PorterStemmer()
snow = SnowballStemmer(language = 'english')
lancaster = LancasterStemmer()
```

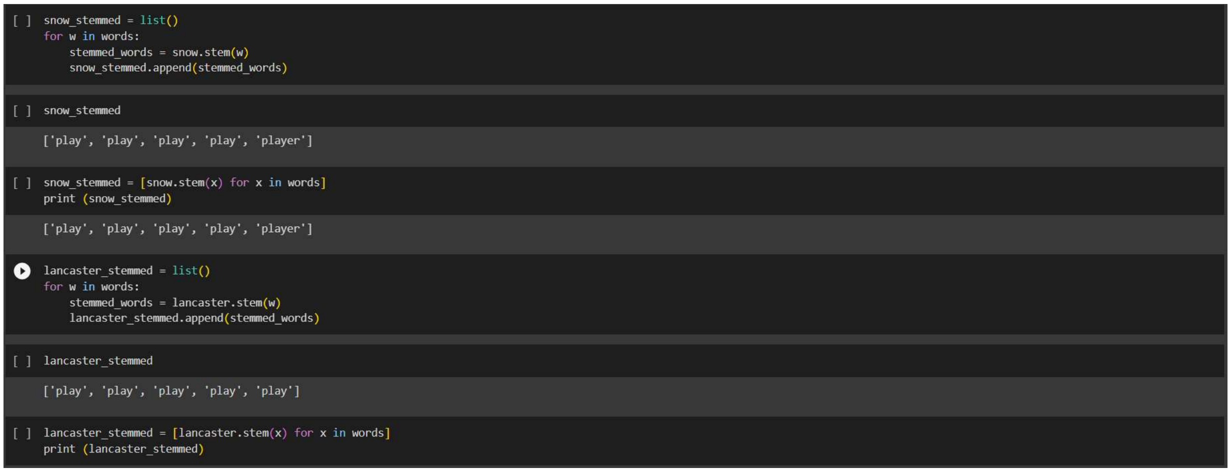
```
words = ['play', 'plays', 'played', 'playing', 'player']
```

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

```
porter_stemmed
```

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

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



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

```
snow_stemmed
```

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

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

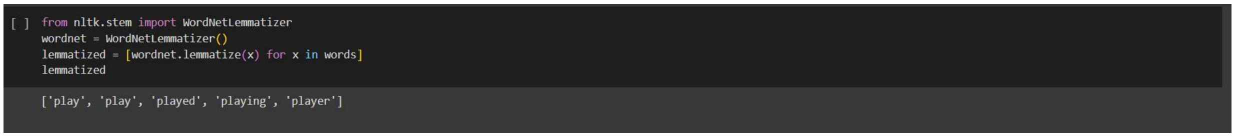
```
['play', 'play', 'play', 'play', 'player']
```

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

```
lancaster_stemmed
```

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

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



```
from nltk.stem import WordNetLemmatizer
wordnet = WordNetLemmatizer()
lemmatized = [wordnet.lemmatize(x) for x in words]
lemmatized
```

```
['play', 'play', 'played', 'playing', 'player']
```

Conclusion:

Steps for Stop Word Removal:

- Tokenization: The text is first tokenized, breaking it down into individual words or tokens. Tokenization identifies the boundaries between words and helps separate stop words from content words.
- Stop Word List: A predefined list of stop words for the specific Indian language is required. These lists typically include common words that are considered non-informative and can be safely removed.
- Comparison: Each token from the tokenized text is compared to the stop word list to determine if it's a stop word.
- Removal: If a token is identified as a stop word, it is removed from the text. If it's not a stop word, it is retained in the text.

Tools for Stop Word Removal:

- Language-Specific Libraries: Some Indian languages have language-specific NLP libraries that include stop word lists and functions for stop word removal. For example, libraries like NLTK for Python may have stop word lists and functions for a variety of languages.
- NLP Libraries: General-purpose NLP libraries like SpaCy, Gensim, and NLTK often provide stop word lists and functions for various languages. You can use these libraries to remove stop words from your Indian language text.
- Custom Stop Word Lists: You can create or curate custom stop word lists for your specific Indian language. These lists can be based on common linguistic characteristics and the specific needs of your NLP project.