EXPERIMENT 3 – Stop word removal

Problem Statement: To understand and implement **Stop word removal**

Theory:

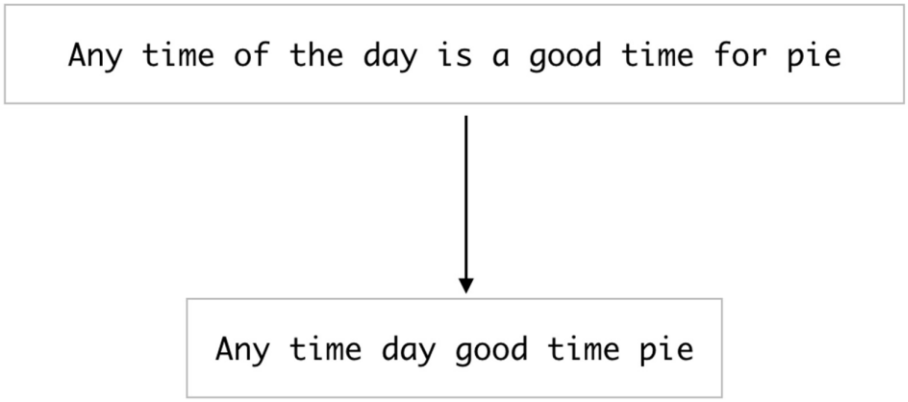
The process of converting data to something a computer can understand is referred to as pre-processing. One of the primary forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

There is also a corpus of stop words, that is, high-frequency of words like "the, to and also" that we sometimes want to filter out of a document before further processing. In computing, stop words are words that are filtered out before or after the natural language data (text) are processed. While "stop words" typically refers to the most common words in a language, all-natural language processing tools don't use a single universal list of stop words.

Stopwords are the **words** in any language which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For some search engines, these are some of the most common, short function words, 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.

Stop words usually have little lexical content, and their presence in a text fails to distinguish it from other texts

Example:



Different Methods to remove stop words are as follows:

- **Using NLTK library:**
  The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging, and semantic reasoning.
- **Using SpaCy Library:**
  spaCy is an open-source software library for advanced natural language processing. spaCy is designed specifically for production use and helps you build applications that process and "understand" large volumes of text. It can be used to build information extraction or natural language understanding systems or to pre-process text for deep learning.
- **Using Gensim Library:**
  Gensim is an open-source library for unsupervised topic modeling and natural language processing, using modern statistical machine learning. Gensim is designed to handle large text collections using data streaming and incremental online algorithms, which differentiates it from most other machine learning software packages that target only in-memory processing. For more details checkout Gensim documentation.
  Using Gensim we can directly call remove_stopwords(), which is a method of gensim.parsing.preprocessing. Next, we need to pass our sentence from which you want to remove stop words, to the remove_stopwords() method which returns the text string without the stop words. We can then tokenize the returned sentences.
- **Custom stop words:**
  If you feel that the default stop words in any python NLP language tool are too many and are causing loss of information, or are too less to remove all unnecessary words in your corpus, then we can opt for custom stop words list.
  For this, you can simply obtain the default stop words to a list and append or delete the required words from the list as per the requirement.

Code:

Link to Colab Notebook:
https://colab.research.google.com/drive/1TKw8h2tWNng5g0zteUB78zbX6_exuwzf?usp=sharing

Remove stop words using the NLTK python library

```
# importing NLTK libarary stopwords
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
from nltk.tokenize import word_tokenize

print(stopwords.words('english'))

# random sentecnce with lot of stop words
sample_text = "Oh man, this is pretty cool. We will do more such things."
text_tokens = word_tokenize(sample_text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words('english')]

print(text_tokens)
print(tokens_without_sw)
```

NLTK Stop word list:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

```
Output:

Tokenized text with stop words :

['Oh', 'man', ',', 'this', 'is', 'pretty', 'cool', '.', 'We', 'will', 'do', 'more', 'such', 'things', '.']

Tokenized text with out stop words :

['Oh', 'man', ',', 'pretty', 'cool', '.', 'We', 'things', '.']
```

Remove stop words using spacy library

```
import spacy
from nltk.tokenize import word_tokenize
# loading english language model of spaCy
en_model = spacy.load('en_core_web_sm')
# gettign the list of default stop words in spaCy english model
stopwords = en_model.Defaults.stop_words

sample_text = "Oh man, this is pretty cool. We will do more such things."
text_tokens = word_tokenize(sample_text)
tokens_without_sw= [word for word in text_tokens if not word in stopwords]

print(text_tokens)
print(tokens_without_sw)
```

Spacy Stop word list:

['whence', 'here', 'show', 'were', 'why', 'n't', 'the', 'whereupon', 'not', 'more', 'how', 'eight', 'indeed', 'i', 'only', 'via', 'nine', 're', 'themselves', 'almost', 'to', 'already', 'front', 'least', 'becomes', 'thereby', 'doing', 'her', 'together', 'be', 'often', 'then', 'quite', 'less', 'many', 'they', 'ourselves', 'take', 'its', 'yours', 'each', 'would', 'may', 'namely', 'do', 'whose', 'whether', 'side', 'both', 'what', 'between', 'toward', 'our', 'whereby', "'m", 'formerly', 'myself', 'had', 'really', 'call', 'keep', "'re", 'hereupon', 'can', 'their', 'eleven', "'m", 'even', 'around', 'twenty', 'mostly', 'did', 'at', 'an', 'seems', 'serious', 'against', "n't", 'except', 'has', 'five', 'he', 'last', "'ve", 'because', 'we', 'himself', 'yet', 'something', 'somehow', "'m", 'towards', 'his', 'six', 'anywhere', 'us', "'d", 'thru', 'thus', 'which', 'everything', 'become', 'herein', 'one', 'in', 'although', 'sometime', 'give', 'cannot', 'besides', 'across', 'noone', 'ever', 'that', 'over', 'among', 'during', 'however', 'when', 'sometimes', 'still', 'seemed', 'get', "'ve", 'him', 'with', 'part', 'beyond', 'everyone', 'same', 'this', 'latterly', 'no', 'regarding', 'elsewhere', 'others', 'moreover', 'else', 'back', 'alone', 'somewhere', 'are', 'will', 'beforehand', 'ten', 'very', 'most', 'three', 'former', "'re", 'otherwise', 'several', 'also', 'whatever', 'am', 'becoming', 'beside', "'s", 'nothing', 'some', 'since', 'thence', 'anyway', 'out', 'up', 'well', 'it', 'various', 'four', 'top', "'s", 'than', 'under', 'might', 'could', 'by', 'too', 'and', 'whom', "'ll", 'say', 'therefore', "'s", 'other', 'throughout', 'became', 'your', 'put', 'per', "'ll", 'fifteen', 'must', 'before', 'whenever', 'anyone', 'without', 'does', 'was', 'where', 'thereafter', "'d", 'another', 'yourselves', 'n't', 'see', 'go', 'wherever', 'just', 'seeming', 'hence', 'full', 'whereafter', 'bottom', 'whole', 'own', 'empty', 'due', 'behind', 'while', 'onto', 'wherein', 'off', 'again', 'a', 'two', 'above', 'therein', 'sixty', 'those', 'whereas', 'using', 'latter', 'used', 'my', 'herself', 'hers', 'or', 'neither', 'forty', 'thereupon', 'now', 'after', 'yourself', 'whither', 'rather', 'once', 'from', 'until', 'anything', 'few', 'into', 'such', 'being', 'make', 'mine', 'please', 'along', 'hundred', 'should', 'below', 'third', 'unless', 'upon', 'perhaps', 'ours', 'but', 'never', 'whoever', 'fifty', 'any', 'all', 'nobody', 'there', 'have', 'anyhow', 'of', 'seem', 'down', 'is', 'every', "'ll", 'much', 'none', 'further', 'me', 'who', 'nevertheless', 'about', 'everywhere', 'name', 'enough', "'d", 'next', 'meanwhile', 'though', 'through', 'on', 'first', 'been', 'hereby', 'if', 'move', 'so', 'either', 'amongst', 'for', 'twelve', 'nor', 'she', 'always', 'these', 'as', "'ve", 'amount', "'re", 'someone', 'afterwards', 'you', 'nowhere', 'itself', 'done', 'hereafter', 'within', 'made', 'ca', 'them']

```
Output:

Tokenized text with stop words :

['Oh', 'man', ',', 'this', 'is', 'pretty', 'cool', '.', 'We', 'will', 'do', 'more', 'such', 'things', '.']

Tokenized text with out stop words :

['Oh', 'man', ',', 'pretty', 'cool', '.', 'We', 'things', '.']
```

Remove stop words using the Gensim library

```
from gensim.parsing.preprocessing import remove_stopwords

sample_text = "Oh man, this is pretty cool. We will do more such things."
sample_text_NSW = remove_stopwords(text)

print(word_tokenize(sample_text))
print(word_tokenize(sample_text_NSW))
```

Gensim Stop word list:

frozenset({'her', 'during', 'among', 'thereafter', 'only', 'hers', 'in', 'none', 'with', 'un', 'put', 'hence', 'each', 'would', 'have', 'to', 'itself', 'that', 'seeming', 'hereupon', 'someone', 'eight', 'she', 'forty', 'much', 'throughout', 'less', 'was', 'interest', 'elsewhere', 'already', 'whatever', 'or', 'seem', 'fire', 'however', 'keep', 'detail', 'both', 'yourselves', 'indeed', 'enough', 'too', 'us', 'wherein', 'himself', 'behind', 'everything', 'part', 'made', 'thereupon', 'for', 'nor', 'before', 'front', 'sincere', 'really', 'than', 'alone', 'doing', 'amongst', 'across', 'him', 'another', 'some', 'whoever', 'four', 'other', 'latterly', 'off', 'sometime', 'above', 'often', 'herein', 'am', 'whereby', 'although', 'who', 'should', 'amount', 'anyway', 'else', 'upon', 'this', 'when', 'we', 'few', 'anywhere', 'will', 'though', 'being', 'fill', 'used', 'full', 'thru', 'call', 'whereafter', 'various', 'has', 'same', 'former', 'whereas', 'what', 'had', 'mostly', 'onto', 'go', 'could', 'yourself', 'meanwhile', 'beyond', 'beside', 'ours', 'side', 'our', 'five', 'nobody', 'herself', 'is', 'ever', 'they', 'here', 'eleven', 'fifty', 'therefore', 'nothing', 'not', 'mill', 'without', 'whence', 'get', 'whither', 'then', 'no', 'own', 'many', 'anything', 'etc', 'make', 'from', 'against', 'ltd', 'next', 'afterwards', 'unless', 'while', 'thin', 'beforehand', 'by', 'amoungst', 'you', 'third', 'as', 'those', 'done', 'becoming', 'say', 'either', 'doesn', 'twenty', 'his', 'yet', 'latter', 'somehow', 'are', 'these', 'mine', 'under', 'take', 'whose', 'others', 'over', 'perhaps', 'thence', 'does', 'where', 'two', 'always', 'your', 'wherever', 'became', 'which', 'about', 'but', 'towards', 'still', 'rather', 'quite', 'whether', 'somewhere', 'might', 'do', 'bottom', 'until', 'km', 'yours', 'serious', 'find', 'please', 'hasnt', 'otherwise', 'six', 'toward', 'sometimes', 'of', 'fifteen', 'eg', 'just', 'a', 'me', 'describe', 'why', 'an', 'and', 'may', 'within', 'kg', 'con', 're', 'nevertheless', 'through', 'very', 'anyhow', 'down', 'nowhere', 'now', 'it', 'cant', 'de', 'move', 'hereby', 'how', 'found', 'whom', 'were', 'together', 'again', 'moreover', 'first', 'never', 'below', 'between', 'computer', 'ten', 'into', 'see', 'everywhere', 'there', 'neither', 'every', 'couldnt', 'up', 'several', 'the', 'll', 'becomes', 'don', 'le', 'been', 'whereupon', 'seemed', 'most', 'noone', 'whole', 'must', 'cannot', 'per', 'my', 'thereby', 'so', 'he', 'name', 'co', 'its', 'everyone', 'if', 'become', 'thick', 'thus', 'regarding', 'didn', 'give', 'all', 'show', 'any', 'using', 'on', 'further', 'around', 'back', 'least', 'since', 'anyone', 'once', 'can', 'bill', 'hereafter', 'be', 'seems', 'their', 'myself', 'nine', 'also', 'system', 'at', 'more', 'out', 'twelve', 'therein', 'almost', 'except', 'last', 'did', 'something', 'besides', 'via', 'whenever', 'formerly', 'cry', 'one', 'hundred', 'sixty', 'after', 'well', 'them', 'namely', 'empty', 'three', 'even', 'along', 'because', 'ourselves', 'such', 'top', 'due', 'inc', 'themselves'})

```
Output:

Tokenized text with stop words :

['Oh', 'man', ',', 'this', 'is', 'pretty', 'cool', '.', 'We', 'will', 'do', 'more', 'such', 'things', '.']

Tokenized text with out stop words :

['Oh', 'man', ',', 'pretty', 'cool', '.', 'We', 'things', '.']
```

Remove Custom stop words

```
# importing NLTK libarary stopwords
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

stopwords_default = stopwords.words('english')
print(len(stopwords_defaut))

stopwords_default.append('like')
, 'marvel', 'ghost'])
print(len(stopwords_default))

# for adding multiple words
stopwords_default.extend(['marvel', 'ghost'])
print(len(stopwords_default))
```

Custom Stop word list:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", 'like', 'marvel', 'ghost']

Output

['Oh', 'man', ',', 'this', 'is', 'pretty', 'cool', '.', 'We', 'will', 'do', 'more', 'such', 'things', 'like', 'this', '.']
['Oh', 'man', ',', 'pretty', 'cool', '.', 'We', 'things', '.']

Conclusion:

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. If we have a task of text classification or sentiment analysis then we should remove stop words as they do not provide any information to our model, i.e **keeping out unwanted words out of our corpus,** but if we have the task of language translation then stopwords are useful, as they have to be translated along with other words.

Post Lab:

1] Explain importance of Stop Word Removal in NLP

- Stop words are often removed from the text before training deep learning and machine learning models since stop words occur in abundance, hence providing little to no unique information that can be used for classification or clustering.
- On removing stopwords, dataset size decreases, and the time to train the model also decreases without a huge impact on the accuracy of the model.
- Stopword removal can potentially help in improving performance, as there are fewer and only significant tokens left. Thus, the classification accuracy could be improved