Assignment No. 07

Aim:

Text Analytics

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency

Prerequisites:

1. Prior knowledge of Python programming.
2. Google Colab / Python IDE
3. Jupyter Notebook

Objectives: To apply document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization and calculate TF-IDF

Theory:

Step 1 : Data Preprocessing

- Tokenization — convert sentences to words
- Removing unnecessary punctuation, tags
- Removing stop words — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- Stemming — words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.
- Lemmatization — Another approach to remove inflection by determining the part of speech and utilizing detailed database of the language.

Text Analytics and NLP

Text communication is one of the most popular forms of day to day conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine. All of these activities are generating text in a significant amount, which is unstructured in nature. I this area of the online marketplace and social media, It is essential to analyze vast quantities of data, to understand peoples opinion.NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information extraction. Analyzing movie review is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

Text Analysis Operations using NLTK

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, open source, easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

Tokenization
```
!pip install nltk
pip install --upgrade pip
```

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences.

```
from nltk.tokenize import sent_tokenize
```

Word Tokenization

Word tokenizer breaks text paragraph into words
Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

Lexicon Normalization

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

Stemming

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

Lemmatization

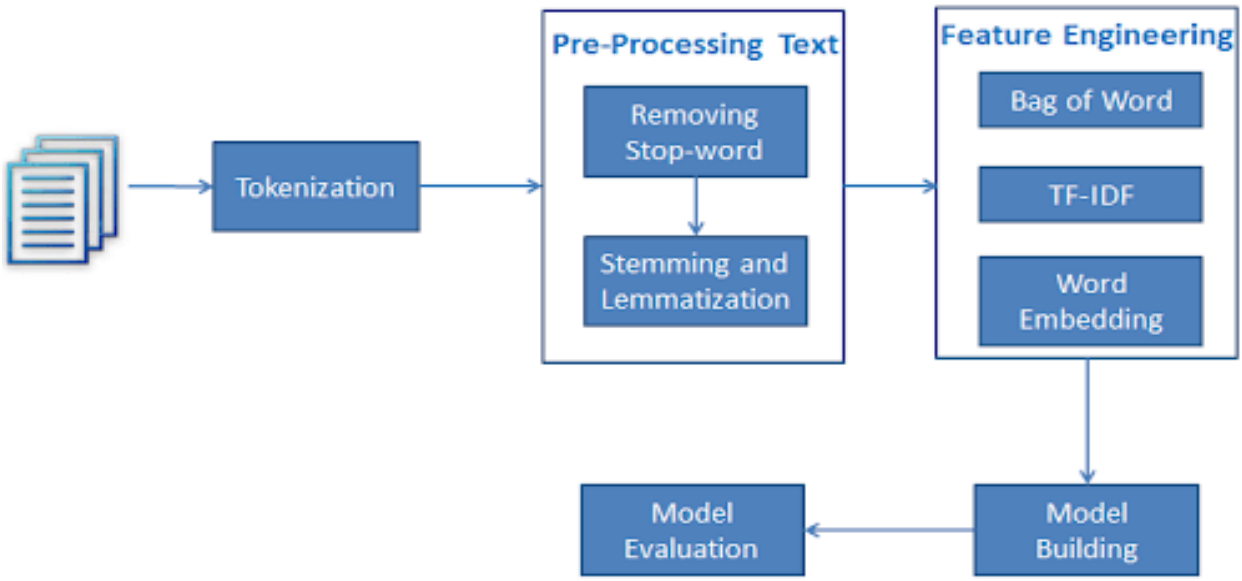
Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, The word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

POS Tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

Text Classification

Text classification is one of the important tasks of text mining. It is a supervised approach. Identifying category or class of given text such as a blog, book, web page, news articles, and tweets. It has various applications in today's computer world such as spam detection, task categorization in CRM services, categorizing products on E-retailer websites, classifying the content of websites for a search engine, sentiments of customer feedback, etc.



Loading Data

Till now, you have learned data pre-processing using NLTK. Now, you will learn Text Classification. You will perform Multi-Nomial Naive Bayes Classification using scikit-learn.
In the model the building part, you can use the "Sentiment Analysis of Movie, Reviews" dataset available on Kaggle. The dataset is a tab-separated file. Dataset has four columns PhraseId, SentenceId, Phrase, and Sentiment

```
import pandas as pd
data=pd.read_csv('train.tsv', sep='\t') (tab separated values)
data.head()
```

Feature Generation using Bag of Words

In the Text Classification Problem, we have a set of texts and their respective labels. But we directly can't use text for our model. You need to convert these text into some numbers or vectors of numbers.Bag-of-words model(BoW) is the simplest way of extracting features from the text. BoW converts text into the matrix of occurrence of words within a document. This model concerns about whether given words occurred or not in the document.
Example: There are three documents:
Doc 1: I love dogs. Doc 2: I hate dogs and knitting. Doc 3: Knitting is my hobby and passion.
Now, you can create a matrix of document and words by counting the occurrence of words in the given document. This matrix is known as Document-Term Matrix(DTM).

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

Split train and test set

Model Building and Evaluation

Let's build the Text Classification Model using TF-IDF.
First, import the MultinomialNB module and create a Multinomial Naive Bayes classifier object using MultinomialNB() function.Then, fit your model on a train set using fit() and perform prediction on the test set using predict().

Feature Generation using TF-IDF

In Term Frequency(TF), you just count the number of words occurred in each document. The main issue with this Term Frequency is that it will give more weight to longer documents. Term frequency is basically the output of the BoW model.IDF(Inverse Document Frequency) measures the amount of information a given word provides across the document. IDF is the logarithmically scaled inverse ratio of the number of documents that contain the word and the total number of documents.

$$\text{idf(W)} = \log \frac{\#(\text{documents})}{\#(\text{documents containing word W})}$$

TF-IDF(Term Frequency-Inverse Document Frequency) normalizes the document term matrix. It is the product of TF and IDF. Word with high tf-idf in a document, it is most of the times occurred in given documents and must be absent in the other documents. So the words must be a signature word.

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	0.18	**0.48**	0.18							
Doc 2	0.18		0.18	**0.48**	0.18	0.18				
Doc 3					0.18	0.18	**0.48**	**0.95**	**0.48**	**0.48**

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
```

Split train and test set (TF-IDF)

Model Building and Evaluation (TF-IDF)

Conclusion:

Thus we have studied document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization and calculate TF-IDF