Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Program in
Artificial Inteliigence & Machine Learning

Natural Language Processing
Question Bank

edureka!

Natural Language Processing

Table of Content

Text Pre-processing

Text pre-processing is a technique of deriving meaningful information from Natural text for further analysis using NLP

It usually involves the process of structuring the input text, deriving patterns within the structured data, and finally evaluating and interpreting the output

Scenario-1: Trump Tweet Analysis using NLP

As we all know, Donald Trump was elected as U.S President on January 20, 2017. He became very popular on Twitter through his tweets.

One of the Analysts, in the Twitter company, was following Donald on Twitter and was very impressed by his tweets. He decided to apply some NLP techniques and finding out some insights about his tweets.

Problem statement:

Analyst of Twitter wants to find out what was the most positive and negative tweet of Donald Trump in the year 2017. Hence he decides to apply NLP techniques to the data he collected about Trump's tweet over the year

Tasks to be performed:

Our objective is to find the best positive and negative tweet Trump has made. Inorder, to find that you have to perform the below tasks,

- Import the data and cleaning is priority-Beginner
- Find out how often did Trump Tweet in 2017- Beginner
- Clean the data (Use functions from Python libraries such as re, string and NLTK to remove these unnecessary elements) and store cleaned data as a separate column to the DataFrame- Intermediate
- Process the data to remove elements which may cause issues in analysis and store processed data as a separate column to the DataFrame- Intermediate
- Apply Ngrams to processed data and print first 3 rows- Beginner
- Apply porter stemmer technique to processed data and store stemmer(porter) data as a separate column to the DataFrame- Intermediate
- Apply Lancaster stemmer technique to processed data and store stemmer(lancaster) data as a separate column to the DataFrame- Intermediate
- Apply Snowball stemmer technique to processed data and store stemmer(snowball) data as a separate column to the DataFrame- Intermediate
- Apply Lemmatization technique to processed data and store lemmatized data as a separate column to the DataFrame- Intermediate
- Calculate Sentiment scores for each text- Beginner
- Calculate polarity for processed data and store polarity as a separate column to DataFrame- Beginner
- Find the number of tweets of each category of polarity (Positive, Negative and Neutral)- Advance
- Find the most positive and most negative tweet using polarity Score- Advance
- Find the Parts of Speech (POS) for most positive and most negative tweet- Advance
- Apply Named Entity Recognition (NER) for most positive and most negative tweet and write your Inference- Advance

Dataset Description:

The dataset consist of 7375 tweets with 10 columns

- **Date** - Date of the Tweet
- **Time** - Time of the Tweet
- **Tweet_Text**- Text of Tweet
- **Type**- Type of the Tweet (Text, Image etc)
- **Media_Type**- Type of Media Involved
- **Hashtags**- Hashtags tagged along with Tweet
- **Tweet_Id**- Twitter ID
- **Tweet_Url**- Webpage of the Tweet
- **twt_favourites_IS_THIS_LIKE_QUESTION_MARK**- How many people have liked his tweet
- **Retweets**- Retweets he recived for his tweets

Topics Covered:

- Tokenization
- Ngrams
- Stemming
- Lemmatization
- Stop words Removal
- POS tags
- Named Entity Recognition (NER)

Before Moving on to solving the problem we need to install some of NLTK libraries

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

```
Requirement already satisfied: nltk in /usr/local/lib/python3.6/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from nltk) (1.12.0)
```

In [ ]:
```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

Out[ ]:
```
True
```

In [ ]:
```
# Load the required libraries from Python
# Make sure all the libraries have been download else download using nltk.download command
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import re
import nltk
```

In [ ]:
```
!wget https://www.dropbox.com/s/v0gmlmnxiqt1vga/Donald-Tweets%21.csv?dl=0
```

```
--2020-07-21 06:37:22--  https://www.dropbox.com/s/v0gmlmnxiqt1vga/Donald-Tweets%21.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.81.1, 2620:100:6031:1::a27d:5101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.81.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/v0gmlmnxiqt1vga/Donald-Tweets%21.csv [following]
--2020-07-21 06:37:22--  https://www.dropbox.com/s/raw/v0gmlmnxiqt1vga/Donald-Tweets%21.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucac88617e5ff72dd6ba92eeab38.dl.dropboxusercontent.com/cd/0/inline/A74np5YVJkmpoZoSbYEmRI8bWHkuSWpp
afO8n0v_QIyQDxQ9vTQfyzpCnTGzNR8b6odmHlABXi0YjidYx0DPYvmZVvM6vmGC3IHIF-i88jtHUC50pJbBp0DpNfwMWi2Pi3o/file# [following]
--2020-07-21 06:37:23--  https://ucac88617e5ff72dd6ba92eeab38.dl.dropboxusercontent.com/cd/0/inline/A74np5YVJkmpoZoSb
YEmRI8bWHkuSWppafO8n0v_QIyQDxQ9vTQfyzpCnTGzNR8b6odmHlABXi0YjidYx0DPYvmZVvM6vmGC3IHIF-i88jtHUC50pJbBp0DpNfwMWi2Pi3o/fi
le
Resolving ucac88617e5ff72dd6ba92eeab38.dl.dropboxusercontent.com (ucac88617e5ff72dd6ba92eeab38.dl.dropboxusercontent.
com)... 162.125.81.15, 2620:100:6031:15::a27d:510f
Connecting to ucac88617e5ff72dd6ba92eeab38.dl.dropboxusercontent.com (ucac88617e5ff72dd6ba92eeab38.dl.dropboxusercont
ent.com)|162.125.81.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1703362 (1.6M) [text/plain]
Saving to: ‘Donald-Tweets!.csv?dl=0’

Donald-Tweets!.csv? 100%[===================>]   1.62M  --.-KB/s    in 0.03s

2020-07-21 06:37:23 (63.9 MB/s) - ‘Donald-Tweets!.csv?dl=0’ saved [1703362/1703362]
```

Question-1: Import the data and cleaning is priority

In [ ]:
```python
tweet = pd.read_csv("/content/Donald-Tweets!.csv?dl=0")
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

In [ ]:
```python
# dropping unnamed columns
tweet.drop(columns=['Unnamed: 10','Unnamed: 11'],inplace=True)
```

In [ ]:
```python
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-2: Find out how often did Trump Tweet in 2017

```
In [ ]: tweet_by_date = tweet.copy()
        tweet_by_date['Date'] = pd.to_datetime(tweet['Date'], yearfirst=True)
        tweet_by_date['Date'] = tweet_by_date['Date'].dt.month
        tweet_by_date = pd.DataFrame(tweet_by_date.groupby(['Date']).size().sort_values(ascending=True).rename('Tweets'))
        tweet_by_date
```

Out[ ]:

Date	Tweets
6	258
4	287
5	357
3	456
1	507
2	516
12	579
11	709
8	726
9	740
7	893
10	1347

We can see that he twitted more in the tenth month of 2017

Question-3: Clean the data (Use functions from Python libraries such as re, string and NLTK to remove these unnecessary elements) and store cleaned data as a separate column to the DataFrame

Observe that the tweet text contains various elements such as 'punctuation marks' and 'stop words' Use functions from Python libraries such as re, string and NLTK to remove these unnecessary elements

```
In [ ]: # Load the required libraries for cleaning
        import string,re
        from nltk.tokenize import word_tokenize
        from nltk.corpus import stopwords
```

```
In [ ]: # Create a function to generate cleaned data from raw text
        def clean_text(tweet):
            tweet = word_tokenize(tweet) # Create tokens
            tweet = tweet[4:] # Remove RT@
            tweet= " ".join(tweet) # Join tokens
            tweet= re.sub('https','',tweet) # Remove 'https' text with blank
            tweet = [char for char in tweet if char not in string.punctuation] # Remove punctuations
            tweet = ''.join(tweet) # Join the leters
            tweet = [word for word in tweet.split() if word.lower() not in stopwords.words('english')] # Remove common english
        words (I, you, we,...)
            return " ".join(tweet)
```

```
In [ ]: # Apply the function to 'cleaned_text' to clean it
        # Add cleaned data as a separate column to the DataFrame

        tweet['cleaned_text']=tweet['Tweet_Text'].apply(clean_text)
```

```
In [ ]: # Print the first 5 values of cleaned tweet data

        tweet['cleaned_text'].head()
```

```
Out[ ]: 0    deepest gratitude served armed forces ThankAVe...
        1    New York soon making important decisions peopl...
        2    small groups protesters last night passion gre...
        3    open successful presidential election professi...
        4    DC Met President Obama first time Really good ...
        Name: cleaned_text, dtype: object
```

Question-4: Process the data to remove elements which may cause issues in analysis and store processed data as a separate column to the DataFrame

Apart from cleaning, data also needs to be processed to remove elements which may cause issues in analysis.Examples of such elements are 'single characters', 'multiple spaces', 'Upper-cased'.Apply various text pre-processing techniques one-by-one to the cleaned data

- Remove all the special characters
- Remove single characters appearing in the text except the start
- Remove single characters appearing at the start
- Substitute multiple spaces with a single space Convert to lowercase

In [ ]:
```
features = tweet['cleaned_text']
processed_features = []

for sentence in range(0, len(features)):
    # Remove all the special characters
    processed_feature = re.sub(r'\W', ' ', str(features[sentence]))

    # Remove single characters appearing in the text except the start
    processed_feature= re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

    # Remove single characters appearing at the start
    processed_feature = re.sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)

    # Substitute multiple spaces with a single space
    processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)

    # Convert to lowercase
    processed_feature = processed_feature.lower()

    processed_features.append(processed_feature)
```

In [ ]:
```
# Print first five values of processed data
processed_features[:5]
```

Out[ ]:
```
['deepest gratitude served armed forces thankavet tcowpk7qwpk8z',
 'new york soon making important decisions people running government',
 'small groups protesters last night passion great country come together proud',
 'open successful presidential election professional protesters incited media protesting unfair',
 'dc met president obama first time really good meeting great chemistry melania liked mrs lot']
```

In [ ]:
```
# Add the processed data as a separate column to the DataFrame

tweet['processed_text'] = processed_features
```

In [ ]:
```
# Observe the entire data

tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-5: Apply Ngrams to processed data and print first 3 rows

In [ ]:
```
processed_text_ngrams=list(nltk.ngrams(tweet['processed_text'], 8))
```

```
In [ ]: processed_text_ngrams[:3]
```

```
Out[ ]: [('deepest gratitude served armed forces thankavet tcowpk7qwpk8z',
          'new york soon making important decisions people running government',
          'small groups protesters last night passion great country come together proud',
          'open successful presidential election professional protesters incited media protesting unfair',
          'dc met president obama first time really good meeting great chemistry melania liked mrs lot',
          'us marine corps thank service tcolz2dhrxzo4',
          'important evening forgotten man woman never forgotten come together never',
          '945pm electionnight maga tcohfujerzbod'),
         ('new york soon making important decisions people running government',
          'small groups protesters last night passion great country come together proud',
          'open successful presidential election professional protesters incited media protesting unfair',
          'dc met president obama first time really good meeting great chemistry melania liked mrs lot',
          'us marine corps thank service tcolz2dhrxzo4',
          'important evening forgotten man woman never forgotten come together never',
          '945pm electionnight maga tcohfujerzbod',
          'surreal moment vote father president united states make voice heard vote election2'),
         ('small groups protesters last night passion great country come together proud',
          'open successful presidential election professional protesters incited media protesting unfair',
          'dc met president obama first time really good meeting great chemistry melania liked mrs lot',
          'us marine corps thank service tcolz2dhrxzo4',
          'important evening forgotten man woman never forgotten come together never',
          '945pm electionnight maga tcohfujerzbod',
          'surreal moment vote father president united states make voice heard vote election2',
          'join family incredible movement makeamericagreatagain please vote america')]
```

Question-6: Apply porter stemmer technique to processed data and store stemmer(porter) data as a separate column to the DataFrame

Porter stemmer remove the commoner morphological and inflexional endings from words in English

```
In [ ]: # Stemming
        from nltk.stem.porter import PorterStemmer

        def get_stemmed_text(corpus):
            stemmer = PorterStemmer()
            return [' '.join([stemmer.stem(word) for word in review.split()]) for review in corpus]

        tweet['stemmedtext(porter)'] = get_stemmed_text(tweet['processed_text'])
```

```
In [ ]: tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIKI
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-7: Apply Lancaster stemmer technique to processed data and store stemmer(lancaster) data as a separate column to the DataFrame

Lancaster Stemmer is used to check the frequency of words

In [ ]:
```python
from nltk.stem import LancasterStemmer

def get_stemmed_text(corpus):
    stemmer = LancasterStemmer()
    return [' '.join([stemmer.stem(word) for word in review.split()]) for review in corpus]

tweet['stemmedtext(lancaster)'] = get_stemmed_text(tweet['processed_text'])
```

In [ ]:
```python
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-8: Apply Snowball stemmer technique to processed data and store stemmer(snowball) data as a separate column to the DataFrame

Snowball Stemmer is similar to porter stemmer, but here we have to specify the English

In [ ]:
```python
from nltk.stem import SnowballStemmer

def get_stemmed_text(corpus):
    stemmer = SnowballStemmer('english')
    return [' '.join([stemmer.stem(word) for word in review.split()]) for review in corpus]

tweet['stemmedtext(snowball)'] = get_stemmed_text(tweet['processed_text'])
```

In [ ]:
```python
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

As You can see we have applied all the stemming techniques because, this is the preprocessing phase. It will be helpful to analyze which stemming techniques can be used while building the model

Question-9: Apply Lemmatization technique to processed data and store lemmatized data as a separate column to the DataFrame

Lemmatize maps several words into one coomon root

In [ ]:
```
# Lemmatization
from nltk.stem import WordNetLemmatizer
def get_lemmatized_text(corpus):
    lemmatizer = WordNetLemmatizer()
    return [' '.join([lemmatizer.lemmatize(word) for word in review.split()]) for review in corpus]

tweet['lemmatext'] = get_lemmatized_text(tweet['processed_text'])
```

In [ ]:
```
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIKI
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

In [ ]:
```
tweet['lemma_str'] = [' '.join(map(str,l)) for l in tweet['lemmatext']]
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIKI
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

```
In [ ]: tweet['word_count'] = tweet['lemmatext'].apply(lambda x: len(str(x).split()))
        tweet['review_len'] = tweet['lemma_str'].astype(str).apply(len)
        tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIKI
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-10: Calculate Sentiment scores for each text

Polarity is a measure which ranges from [-1,1]. Where 1 means positive Statement and -1 means Negative Statement. 0 is Neutral

```
In [ ]: from textblob import TextBlob
```

```
In [ ]: # Create a function to calculate Sentiment scores for each text
        def generate_polarity(text):
            sentiment = TextBlob(text).sentiment
            return sentiment
```

```
In [ ]: # Apply the function to processed data
        sentiment = tweet['lemmatext'].apply(generate_polarity)
        sentiment = sentiment.to_frame()
        sentiment.head()
```

Out[ ]:

	lemmatext
0	(0.0, 0.0)
1	(0.2681818181818182, 0.7272727272727273)
2	(0.3375, 0.5541666666666667)
3	(0.0875, 0.6375)
4	(0.5875, 0.6208333333333333)

```
In [ ]: tweet['sentiment_score'] = sentiment
        tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-11: Calculate polarity for processed data and store polarity as a separate column to DataFrame

```
In [ ]: # Use the first element as Polarity
        sentiment['polarity'] = sentiment['lemmatext'].apply(lambda x:x[0])
```

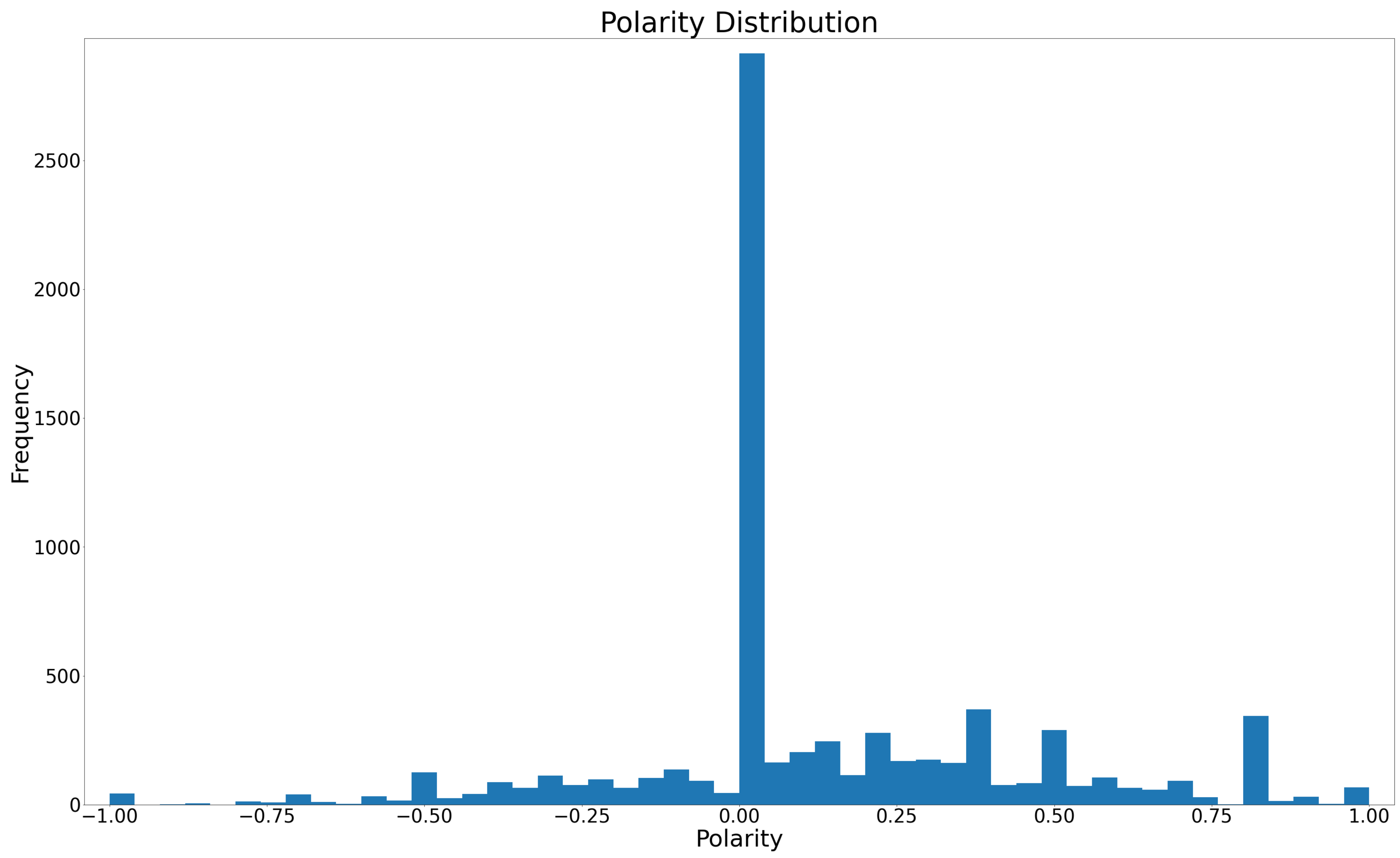
```
In [ ]: # Add a column to DataFrame for Polarity score respectively

        tweet['polarity'] = sentiment['polarity']
        tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

Question-12: Visualize the sentiment of the sentences using histogram

In [ ]:
```python
import matplotlib.pyplot as plt
plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.xlabel('Polarity', fontsize=50)
plt.xticks(fontsize=40)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.hist(tweet['polarity'], bins=50)
plt.title('Polarity Distribution', fontsize=60)
plt.show()
```



Question-13: Find the number of tweets of each category of polarity (Positive, Negative and Neutral)

In [ ]:
```python
# Encode polarity into 'positive', 'negative' and 'neutral' based on the score

tweet['polarity_encoded'] = ['positive' if x > 0 else 'negative' if x < 0 else 'neutral' for x in tweet['polarity']]
```

In [ ]:
```python
# Print the number of tweets of each category of polarity
tweet['polarity_encoded'].value_counts()
```

Out[ ]:
```
positive    3278
neutral     2851
negative    1246
Name: polarity_encoded, dtype: int64
```

We can observe that he received more positive and neutral comments rather than negative comments which shows that his actions were appreciated by people of U.S

Question-14: Build a ML model and calculate the accuracy

In [ ]:
```
tweet.head()
```

Out[ ]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	Tweet_Url	twt_favourites_IS_THIS_LIK
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/797...	
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realDonaldTrump/status/796...	

In [ ]:
```
#new_df = tweet.drop(columns = ['sentiment_score','polarity'],axis = 1)
#new_df.head()
```

In [ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(lowercase=True,ngram_range = (1,1),stop_words='english')
```

In [ ]:
```
X = tweet['processed_text']
y = tweet['polarity_encoded']
```

In [ ]:
```
X_vect = vectorizer.fit_transform(X)
df_vect = pd.DataFrame(X_vect.toarray(),columns=vectorizer.get_feature_names())
print(df_vect.head())
print(df_vect.shape)
```

```
    00  007llisav  00patriot   08  08102015  ...  zuckermans  ωtime   ɢɢ  ʊʊʊ   ʟʟ
0  0.0        0.0        0.0  0.0       0.0  ...         0.0    0.0  0.0  0.0  0.0
1  0.0        0.0        0.0  0.0       0.0  ...         0.0    0.0  0.0  0.0  0.0
2  0.0        0.0        0.0  0.0       0.0  ...         0.0    0.0  0.0  0.0  0.0
3  0.0        0.0        0.0  0.0       0.0  ...         0.0    0.0  0.0  0.0  0.0
4  0.0        0.0        0.0  0.0       0.0  ...         0.0    0.0  0.0  0.0  0.0

[5 rows x 11778 columns]
(7375, 11778)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df_vect, y,test_size=0.20,random_state = 7)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(5900, 11778)
(1475, 11778)
(5900,)
(1475,)
```

In [ ]:
```
del X_vect
del df_vect
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
```

```
In [ ]: from sklearn.model_selection import cross_val_score,KFold
        #machine learning algorithms
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.linear_model import LogisticRegression
        #from sklearn.ensemble import GradientBoostingClassifier
        from sklearn.ensemble import RandomForestClassifier


        models=[]
        models.append(('knn',KNeighborsClassifier()))

        #models.append(('lr',LogisticRegression()))
        models.append(('Random Forest',RandomForestClassifier()))
        #models.append(('gradient boosting classifier',GradientBoostingClassifier()))




        for name,model in models:
          kfold=KFold(n_splits=5,random_state=7)
          cross_val_sc=cross_val_score(model,X_train,y_train,scoring='accuracy',cv=kfold)
          print('{} : acc: {}(standard deviation: {})'.format(name,cross_val_sc.mean(),cross_val_sc.std()))
```

```
knn : acc: 0.39966101694915255(standard deviation: 0.016568609369288043)
Random Forest : acc: 0.8179661016949152(standard deviation: 0.012968040178568542)
```

Question-15: Find the most positive and most negative tweet using polarity Score

```
In [ ]: # Print the most positive and most negative tweet

        print("The most positive tweet:",tweet.iloc[tweet['polarity'].idxmax()]['processed_text'])
        print("The most negative tweet:",tweet.iloc[tweet['polarity'].idxmin()]['processed_text'])
```

```
The most positive tweet: thank law enforcement officers vpdebate police officers best us mikepence
The most negative tweet: lead border securityno solutions ideas credibilityshe supported nafta worst deal us history
debate
```

Question-16: Find the Parts of Speech (POS) for most positive and most negative tweet

POS tagging indicates how a word functions in a meaning as well as grammatically within the sentence

```
In [ ]: sent= tweet.iloc[tweet['polarity'].idxmax()]['processed_text']
        sent_tokens= word_tokenize(sent)
```

```
In [ ]: for token in sent_tokens:
          print(nltk.pos_tag([token]))
```

```
[('thank', 'NN')]
[('law', 'NN')]
[('enforcement', 'NN')]
[('officers', 'NNS')]
[('vpdebate', 'NN')]
[('police', 'NNS')]
[('officers', 'NNS')]
[('best', 'JJS')]
[('us', 'PRP')]
[('mikepence', 'NN')]
```

```
In [ ]: sent= tweet.iloc[tweet['polarity'].idxmin()]['processed_text']
        sent_tokens= word_tokenize(sent)
```

```
In [ ]: for token in sent_tokens:
          print(nltk.pos_tag([token]))
```

```
[('lead', 'NN')]
[('border', 'NN')]
[('securityno', 'NN')]
[('solutions', 'NNS')]
[('ideas', 'NNS')]
[('credibilityshe', 'NN')]
[('supported', 'VBN')]
[('nafta', 'NN')]
[('worst', 'JJS')]
[('deal', 'NN')]
[('us', 'PRP')]
[('history', 'NN')]
[('debate', 'NN')]
```

Question-17: Apply Named Entity Recognition (NER) for most positive and most negative tweet and write your Inference

NER is a method of associating the named entities to their appropriate types

It also helps in automatic identification and counting of occurances of named entities in a collection of information

In [ ]:
```
from nltk import ne_chunk
ne_sent=tweet.iloc[tweet['polarity'].idxmax()]['processed_text']
ne_tokens = word_tokenize(ne_sent)
ne_tags= nltk.pos_tag(ne_tokens)
ne_ner= ne_chunk(ne_tags)
print(ne_ner)
```

```
(S
  thank/NN
  law/NN
  enforcement/NN
  officers/NNS
  vpdebate/VBP
  police/NNS
  officers/NNS
  best/VBP
  us/PRP
  mikepence/NN)
```

In [ ]:
```
from nltk import ne_chunk
ne_sent=tweet.iloc[tweet['polarity'].idxmin()]['processed_text']
ne_tokens = word_tokenize(ne_sent)
ne_tags= nltk.pos_tag(ne_tokens)
ne_ner= ne_chunk(ne_tags)
print(ne_ner)
```

```
(S
  lead/JJ
  border/NN
  securityno/JJ
  solutions/NNS
  ideas/NNS
  credibilityshe/VBP
  supported/VBD
  nafta/JJ
  worst/JJS
  deal/VB
  us/PRP
  history/NN
  debate/NN)
```

We can Infer that, NER like POS tagger is also not 100% accurate and sometimes returns wrong answers just like above

Text Summarization

Text Summarization is a process of creating a short and coherent version of the longer document

The 2 Main Approaches involved in Summarizing Text Documents are -

- Extractive Method

> Involves pulling critical phrases from the Source document and then, combining them to make a summary

- Abstractive Method

> Create new phrases and sentences that relay the most useful information from the original text

Scenario 1:Text Summarization on New Year's Resolution Tweets

AV Analytics is an Analytics company headquartered in New York, the United States, with a registered office in San Jose. It is one of the world's largest Data Analytics company. They have collected tweets data of the user's on New Year's Resolution of the year 2015.

In-Text Summarization, we created a short and coherent version of the longer document . It is a useful method because AV Analytics can analyze people's distinct New Years Resolution.

Problem Statement

An Analytics Company called **AV Analytics** has collected the data of twitter user's on New Year's Resolution. The data set is extensive & they do not waste resources on analyzing such a vast data set.

So, being an NLP Developer, you must -

Perform **Text Summarization** on the given dataset to generate a summary of the same

Tasks to be Performed

- Load, analyze, pre-process the dataset to extract the tweet - Beginner
- Create a function **frequency_table** to calculate the Frequency of Words present in the **text** - Intermediate
- Create a score function **score_sentences** that calculates the score for every sentence in the text - Intermediate
- Create a function **find_average_score** to Find the Average Score - Intermediate
- Create a function called **run_summarization** to call all the above functions and then, generate the summary - Beginner

Dataset Description

The dataset describes the Twitter sentiment analysis of users' 2015 New Year's resolutions. Contains demographic and geographical data of users and resolution categorizations.

There are **5011** observations with **8** columns. The original dataset is credited to CrowdFlower.

Here's a brief description of the dataset:

- **resolution_topics** - Topic of the Resolution
- **gender** - Gender
- **name** - Name of the User
- **resolution_category** - Category of the Resolution
- **retweet_count** - Number of Retweets
- **text** - **Actual Tweet to be analyzed**
- **tweet_id** - Unique ID of the Tweets
- **tweet_location** - Location of the Tweeet

Topics Covered

- Text Summarization

In [1]:

```
!wget https://www.dropbox.com/s/bkzr8khi295ysqe/New-years-resolutions-DFE.csv
```

In [ ]:

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[ ]:

```
True
```

Question 1:

Load, analyze, pre-process the dataset to extract the tweet - Beginner

In [ ]:

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

df = pd.read_csv('/content/New-years-resolutions-DFE.csv',encoding= 'unicode_escape')
df.head()
```

Out[ ]:

	resolution_topics	gender	name	Resolution_Category	retweet_count	
0	Eat healthier	female	Dena_Marina	Health & Fitness	0.0	#NewYears :: Read n
1	Humor about Personal Growth and Interests Reso...	female	ninjagirl325	Humor	1.0	#NewYears Finally mas
2	Be More Confident	male	RickyDelReyy	Personal Growth	0.0	#NewYears to stc c
3	Other	male	CalmareNJ	Philanthropic	0.0	#NewYears is to help r
4	Be more positive	female	welovatoyoudemi	Personal Growth	0.0	#NewYears # #2015

In [ ]:

```
df.shape
```

Out[ ]:

```
(5011, 8)
```

In [ ]:

```
text = " "
for i in df.text:
  text = text + i

print(text[0:90])
```

```
 #NewYearsResolution :: Read more books, No scrolling FB/checking email b4
breakfast, stay
```

In [ ]:

```
print(len(text))
```

```
471086
```

Question 2:

Create a function **frequency_table** to calculate the Frequency of Words present in the **text**

In [ ]:

```
def frequency_table(text_string):
    stopWords = set(stopwords.words("english"))
    words = word_tokenize(text_string)
    ps = PorterStemmer()

    freqTable = dict()
    for word in words:
        word = ps.stem(word)
        if word in stopWords:
            continue
        if word in freqTable:
            freqTable[word] += 1
        else:
            freqTable[word] = 1

    return freqTable
```

Question 3:

Create a score function **score_sentences** that calculates the score for every sentence in the text.

In [ ]:

```
def score_sentences(sentences, freqTable):
    sent_value = dict()

    for sentence in sentences:
        word_count = (len(word_tokenize(sentence)))
        word_count_except_sw = 0
        for wordValue in freqTable:
            if wordValue in sentence.lower():
                word_count_except_sw += 1
                if sentence[:10] in sent_value:
                    sent_value[sentence[:10]] += freqTable[wordValue]
                else:
                    sent_value[sentence[:10]] = freqTable[wordValue]

        if sentence[:10] in sent_value:
            sent_value[sentence[:10]] = sent_value[sentence[:10]] / word_count_except_s
w

    return sent_value
```

Adding the frequency of every non-stop word in a sentence divided by total no of words in a sentence is the score of that word.

Disadvantage -
Long sentences will have an advantage over short sentences. To solve this, we're dividing every sentence score by the number of words in the sentence.
Note -
Here sentence[:10] is the first 10 character of any sentence, this is to save memory while saving keys of the dictionary.

Question 4:

Create a function **find_average_score** to Find the Average Score

In [ ]:

```
def find_average_score(sent_value):
    sumValues = 0
    for entry in sent_value:
        sumValues += sent_value[entry]

    # Average value of a sentence from original text
    average = (sumValues / len(sent_value))

    return average
```

Question 5:

Create a function **generate_summary** to generate the summary

In [ ]:

```
def generate_summary(sentences, sent_value, threshold):
    sentence_count = 0
    summary = ''

    for sentence in sentences:
        if sentence[:10] in sent_value and sent_value[sentence[:10]] >= (threshold):
            summary += " " + sentence
            sentence_count += 1

    return summary
```

Average score of the sentence is the threshold

Question 6:

Create a function called **run_summarization** to call all the above functions and then, generate the summary

In [ ]:

```
def run_summarization(text):
    # 1) Create the word frequency table
    freq_table = frequency_table(text)

    # 2) Tokenize the sentences
    sentences = sent_tokenize(text)

    # 3) Important Algorithm: score the sentences
    sentence_scores = score_sentences(sentences, freq_table)

    # 4) Find the threshold
    threshold = find_average_score(sentence_scores)

    # 5) Important Algorithm: Generate the summary
    summary = generate_summary(sentences, sentence_scores, 1.3 * threshold)

    return summary
```

In [ ]:

```
result = run_summarization(text)
```

In [ ]:

```
#print(result)
```

In [ ]:

```
print("Length of Text before Text Summarization:",len(text))
print("Length of Text after Applying Text Summarization:",len(result))
```

```
Length of Text before Text Summarization: 471086
Length of Text after Applying Text Summarization: 43105
```

Scenario 2: Summarizing Famous Quotes

Simon & Schuster, a subsidiary of ViacomCBS, is an American publishing company founded in New York City in 1924 by Richard L. Simon and M. Lincoln Schuster. It is one of the largest publishing houses. They have collected Quotes from famous people.

In-Text Summarization, we create a short and coherent version of the longer document. It is a useful method because Simon & Schuster can analyze these Famous Quotes as a summary.

Problem Statement

A Publishing Company called **Simon & Schuster** has collected the Quotes of Famous people along with its popularity and Category such as Humour, Inspiration, etc. The data is enormous because of which it is difficult for them to analyze.

So, being an NLP Developer, you must -

Perform **Text Summarization** on the given dataset to generate a summary of the same

Tasks to be Performed

- Load, analyze, pre-process the dataset to extract the tweet - Beginner
- Create a function **frequency_table** to calculate the Frequency of Words present in the **text** - Intermediate
- Create a score function **score_sentences** that calculates the score for every sentence in the text - Intermediate
- Create a function **find_average_score** to Find the Average Score - Intermediate
- Create a function called **run_summarization** to call all the above functions and then, generate the summary - Beginner

Dataset Description

The dataset describes the famous Quotes by many great Authors such as Daryl Hall, Lynda Barry, etc. and also the popularity and the category of the Quotes.

There are **48391** observations with **5** columns.

Here's a brief description of the dataset:

- **Quote** - Quote
- **Author** - Author of the Quote
- **Tags** - Quote Tag
- **Popularity** - Popularity of the Quote
- **Category** - Category of the Quote

Topics Covered

- Text Summarization

In [2]:

```
!wget https://www.dropbox.com/s/rregnjsrldt5uld/quotes.json
```

Question 1:

Load, analyze, pre-process the dataset to extract the tweet - Beginner

In [ ]:

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

df = pd.read_json('/content/quotes.json')
df.tail()
```

Out[ ]:

	Quote	Author	Tags	Popularity	Category
48386	In Buddhism, they say attachment to anything o...	Jason Mraz	[Suffering, Laugh, Stage]	0.000000	humor
48387	I love British humor. It's just so - surreal.	Beck	[Love, British, Surreal]	0.000000	humor
48388	I've got a sense of humor. I'm a funny guy.	Daryl Hall	[Funny, Guy]	0.000000	humor
48389	Humor is such a wonderful thing, helping you r...	Lynda Barry	[Time, Beautiful, Fool]	0.000000	humor
48390	Life Is Full of Obstacles, Stumble Upon !!	Pratik Shelar	[inspirational-quotes]	-0.000001	inspiration

In [ ]:

```
df.shape
```

Out[ ]:

```
(48391, 5)
```

In [ ]:

```
text = " "
for i in df.Quote:
  text = text + i

print(text[0:56])
```

```
 Don't cry because it's over, smile because it happened.
```

In [ ]:

```
print(len(text))
```

```
6606710
```

Question 2:

Create a function **frequency_table** to calculate the Frequency of Words present in the **text**

In [ ]:

```
def frequency_table(text_string):
    stopWords = set(stopwords.words("english"))
    words = word_tokenize(text_string)
    ps = PorterStemmer()

    freqTable = dict()
    for word in words:
        word = ps.stem(word)
        if word in stopWords:
            continue
        if word in freqTable:
            freqTable[word] += 1
        else:
            freqTable[word] = 1

    return freqTable
```

Question 3:

Create a score function **score_sentences** that calculates the score for every sentence in the text.

In [ ]:

```
def score_sentences(sentences, freqTable):
    sent_value = dict()

    for sentence in sentences:
        word_count = (len(word_tokenize(sentence)))
        word_count_except_sw = 0
        for wordValue in freqTable:
            if wordValue in sentence.lower():
                word_count_except_sw += 1
                if sentence[:10] in sent_value:
                    sent_value[sentence[:10]] += freqTable[wordValue]
                else:
                    sent_value[sentence[:10]] = freqTable[wordValue]

        if sentence[:10] in sent_value:
            sent_value[sentence[:10]] = sent_value[sentence[:10]] / word_count_except_s
w

    return sent_value
```

Adding the frequency of every non-stop word in a sentence divided by total no of words in a sentence is the score of that word.

Disadvantage -
Long sentences will have an advantage over short sentences. To solve this, we're dividing every sentence score by the number of words in the sentence.
Note -
Here sentence[:10] is the first 10 character of any sentence, this is to save memory while saving keys of the dictionary.

Question 4:

Create a function **find_average_score** to Find the Average Score

In [ ]:

```python
def find_average_score(sent_value):
    sumValues = 0
    for entry in sent_value:
        sumValues += sent_value[entry]

    # Average value of a sentence from original text
    average = (sumValues / len(sent_value))

    return average
```

Question 5:

Create a function **generate_summary** to generate the summary

In [ ]:

```python
def generate_summary(sentences, sent_value, threshold):
    sentence_count = 0
    summary = ''

    for sentence in sentences:
        if sentence[:10] in sent_value and sent_value[sentence[:10]] >= (threshold):
            summary += " " + sentence
            sentence_count += 1

    return summary
```

Average score of the sentence is the threshold

Question 6:

Create a function called **run_summarization** to call all the above functions and then, generate the summary

In [ ]:

```
def run_summarization(text):
    # 1) Create the word frequency table
    freq_table = frequency_table(text)

    # 2) Tokenize the sentences
    sentences = sent_tokenize(text)

    # 3) Important Algorithm: score the sentences
    sentence_scores = score_sentences(sentences, freq_table)

    # 4) Find the threshold
    threshold = find_average_score(sentence_scores)

    # 5) Important Algorithm: Generate the summary
    summary = generate_summary(sentences, sentence_scores, 1.3 * threshold)

    return summary
```

In [ ]:

```
result = run_summarization(text)
```

In [ ]:

```
#print(result)
```

In [ ]:

```
print("Length of Text before Text Summarization:",len(text))
print("Length of Text after Applying Text Summarization:",len(result))
```

```
Length of Text before Text Summarization: 6606710
Length of Text after Applying Text Summarization: 298873
```

Chunking and Chinking

Chunking is also referred to as shallow parsing, which includes Part-Of-Speech (POS) Tagging and adds more structure to the sentence. The result is a grouping of the words in “chunks”

Chinking helps us define, what we want to exclude from a chunk

In [ ]:

```
import os
path_to_gs = 'C:\\Program Files\\gs\\gs9.52\\bin' #Defining the path to the ghost scrip
t file
```

In [ ]:

```
#Modifying environment variable
os.environ['PATH']+=os.pathsep + path_to_gs
```

In [3]:

```
os.environ['PATH']
```

Import the NLTK Libraries

In [4]:

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
from nltk import RegexpParser as regex_parser
```

Question 1: Tokenize the sentence and extract the Part of Speech (POS) Tag

In [ ]:

```
sent = "You must not go right now." #Input Sentence
#sent = "You can't eat that!"
#sent = "My mother is fixing us some dinner."
#sent = "Words were spoken."
#sent = "These cards may be worth hundreds of dollars!"
#sent = "The teacher is writing a report."
#sent = "You have woken up everyone in the neighborhood."

sent_tokens = nltk.pos_tag(word_tokenize(sent)) #Tokenizing and extracting the pos_tag
sent_tokens
```

Out[ ]:

```
[('You', 'PRP'),
 ('must', 'MD'),
 ('not', 'RB'),
 ('go', 'VB'),
 ('right', 'RB'),
 ('now', 'RB'),
 ('.', '.')]
```

Question 2: Define a Grammar for a verb phrase that begins with a Preposition and then have any number of verbs followed by the adverbs

In [ ]:

```
grammar_vp = r"vp: {<PRP>?<VB|VBD|VBZ|VBG>*<RB|RBR>?}"
```

Question 3: Define a Chunk Parser and then, pass the verb phrase string to it and then, use the parse() function to parse the given sentence and display the result

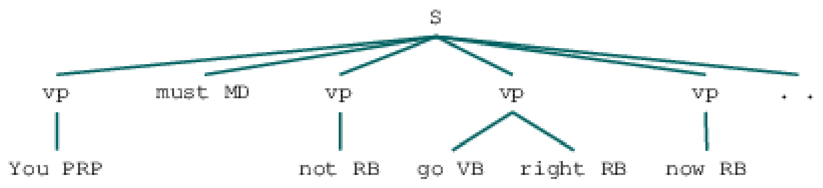
In [ ]:

```
chunk_parser = regex_parser(grammar_vp)
```

In [ ]:

```
chunk_result = chunk_parser.parse(sent_tokens)
chunk_result
```

Out[ ]:



Question 4: Define a Chinking Grammar that removes the Preposition

In [ ]:

```
chunk_grammar = r"""
chk_name: #chunk name

{<PRP>?<VB|VBD|VBZ>*<RB|RBR>?} #chunk regex sequence

}<PRP>+{ #chink regex sequence
"""
```

Question 4: Parse the Chinking Grammar to the RegexpParser and then, pass the given sentence it to display the result

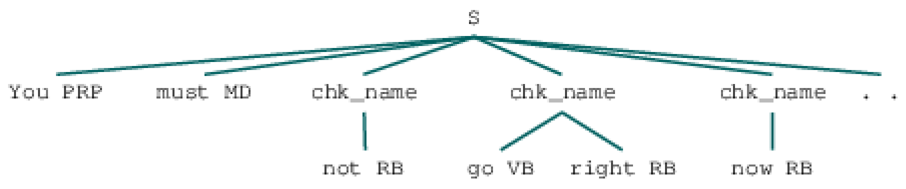
In [ ]:

```
chink_parser = nltk.RegexpParser(chink_grammar)
```

In [ ]:

```
chink_parser.parse(sent_tokens)
```

Out[ ]:



Context Free Grammar

A Context Free Grammar (CFG) is a set of recursive rewriting rules (or productions) used to generate patterns of strings. In Layman's terms, it is a simple grammar, where certain rules describe possible combinations of words and phrases.

It is is a tuple with 4 values:

- N, a finite set of terminal symbols
- Σ is the alphabet, a finite set of terminal symbols
- R, set of production rule
- S, start of symbol € N

Automatic Text Paraphrasing

Dataset Description

The dataset contains English sentences randomly selected from a number of books. It contains only one attribute **sentences**

Load the Dataset and Import the Required Libraries

In [5]:

```
!wget https://www.dropbox.com/s/frf1gdsl2vcrrrw/Book1.xlsx
```

In [ ]:

```
import pandas as pd
df = pd.read_excel('/content/Book1.xlsx')
df.head()
```

Out[ ]:

	sentences
0	Ronak saw a Boat
1	Aditi saw a Dog
2	Devanshu saw an Airplane
3	Raj saw a very big car
4	Divyam saw a Sheep

In [ ]:

```
from nltk.parse.generate import generate, demo_grammar
from nltk import CFG
```

Question 1:

Define a Function that takes an input sentence and tokenizes and pos tag it and then define CFG for the same to generate sentences

In [ ]:

```
def cfg_parse(sentence):
    print("Original Sentence:", sentence)
    sent_tk = nltk.pos_tag(word_tokenize(sentence)) #Tokening and Tagging the Input Sen
tences
    print("Tokenizing & POS Tagging the Sentence:",sent_tk)
    print("Results after Automatic Text Paraphrasing")
    for one in sent_tk:


        if one[1] == 'NNP':
            s_NP = "\'" + one[0] + "\'"

        if one[1] == 'VBD' or one[1]=='VBN':
            s_V = "\'" + one[0] + "\'"

        if one[1] == 'NN':
            s_N = "\'" + one[0] + "\'"



        else: pass
    cfg_grammar2 = nltk.CFG.fromstring("""
    S -> NP VP

    VP -> V N
    NP -> {}
    V -> {}
    N -> {}
    """.format(s_NP,s_V,s_N))
    for sentence in generate(cfg_grammar2):
        print(" ".join(sentence))
    return
```

Question 2:

Loop through the Dataset and generate sentences using the **cfg_parser** function

In [ ]:

```
for i in df.sentences:
    cfg_parse(i)
```

```
Original Sentence: Ronak saw a Boat
Tokenizing & POS Tagging the Sentence: [('Ronak', 'NNP'), ('saw', 'VBD'),
('a', 'DT'), ('Boat', 'NN')]
Results after Automatic Text Paraphrasing
Ronak saw Boat
Original Sentence: Aditi saw a Dog
Tokenizing & POS Tagging the Sentence: [('Aditi', 'NNP'), ('saw', 'VBD'),
('a', 'DT'), ('Dog', 'NN')]
Results after Automatic Text Paraphrasing
Aditi saw Dog
Original Sentence: Devanshu saw an Airplane
Tokenizing & POS Tagging the Sentence: [('Devanshu', 'NNP'), ('saw', 'VB
D'), ('an', 'DT'), ('Airplane', 'NN')]
Results after Automatic Text Paraphrasing
Devanshu saw Airplane
Original Sentence: Raj saw a very big car
Tokenizing & POS Tagging the Sentence: [('Raj', 'NNP'), ('saw', 'VBD'),
('a', 'DT'), ('very', 'RB'), ('big', 'JJ'), ('car', 'NN')]
Results after Automatic Text Paraphrasing
Raj saw car
Original Sentence: Divyam saw a Sheep
Tokenizing & POS Tagging the Sentence: [('Divyam', 'NNP'), ('saw', 'VBD'),
('a', 'DT'), ('Sheep', 'NN')]
Results after Automatic Text Paraphrasing
Divyam saw Sheep
Original Sentence: Raj saw a mouse there
Tokenizing & POS Tagging the Sentence: [('Raj', 'NNP'), ('saw', 'VBD'),
('a', 'DT'), ('mouse', 'NN'), ('there', 'EX')]
Results after Automatic Text Paraphrasing
Raj saw mouse
Original Sentence: Mike saw an animal
Tokenizing & POS Tagging the Sentence: [('Mike', 'NNP'), ('saw', 'VBD'),
('an', 'DT'), ('animal', 'NN')]
Results after Automatic Text Paraphrasing
Mike saw animal
```

From above, you can see that the sentences are tokenized & pos tagged followed by its paraphrase

Text Classification

In natural language processing tasks, “Text Classification” is widely used.

The goal of text classification is to automatically classify the text documents into one or more categories

Scenario 1: Therapy chatbot - Intent classification

Based on user response the bot has to flag the response.

For example:

Bot said: 'Describe a time when you have acted as a resource for someone else'. User responded. If a response is 'not flagged', the user can continue talking to the bot. If it is 'flagged', the user is referred to help

Problem Statement:

You have been provided a 'text-response' dataset to create a model which can predict if the text would be flagged or not?

Data Description:

The dataset contains 3 columns:

- response_id: It contains the index
- class: It contains the class, if the response would be 'flagged' or 'not_flagged'
- response_text: It contains the text data

Tasks to be performed:

1. Load and analyze the dataset - Beginner
2. Convert categorical to numerical feature using Label Encoder - Beginner
3. Implement TF-IDF - Intermediate
4. Split the dataset for training and testing - Beginner
5. Perform K-fold cross validation - Beginner
6. Train a Multinomial Naive Bayes Classifier model, perform prediction and evaluate the model - Beginner

Topics Covered:

Data collection

Pre-processing text data

Implement IF-IDF

Train/Test Algorithms

Predicting using the trained model

Evaluating a model: accuracy score

```
In [1]: !wget https://www.dropbox.com/s/2h4bggnpsze0rkf/Sheet_1.csv
```

```
--2020-07-24 07:19:22--  https://www.dropbox.com/s/2h4bggnpsze0rkf
/Sheet_1.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:
100:6021:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443.
.. connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/2h4bggnpsze0rkf/Sheet_1.csv [following]
--2020-07-24 07:19:22--  https://www.dropbox.com/s/raw/2h4bggnpsze
0rkf/Sheet_1.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc4add22da11f3db4ba13736c165.dl.dropboxuserconte
nt.com/cd/0/inline/A8GbA0SIZHndaYuAj-E755cEZZt7Hy1pS0uXeJuifO_6iz3
f5URRiq9_1sSYkXrImRDjXOpuzoSuO8ejYKlAg-dwxPrexLslPlHxBZfKAj6t8mNYa
9yOQk3QKvEU9ZhXq-g/file# [following]
--2020-07-24 07:19:23--  https://uc4add22da11f3db4ba13736c165.dl.d
ropboxusercontent.com/cd/0/inline/A8GbA0SIZHndaYuAj-E755cEZZt7Hy1p
S0uXeJuifO_6iz3f5URRiq9_1sSYkXrImRDjXOpuzoSuO8ejYKlAg-dwxPrexLslPl
HxBZfKAj6t8mNYa9yOQk3QKvEU9ZhXq-g/file
Resolving uc4add22da11f3db4ba13736c165.dl.dropboxusercontent.com (
uc4add22da11f3db4ba13736c165.dl.dropboxusercontent.com)... 162.125
.65.15, 2620:100:6021:15::a27d:410f
Connecting to uc4add22da11f3db4ba13736c165.dl.dropboxusercontent.c
om (uc4add22da11f3db4ba13736c165.dl.dropboxusercontent.com)|162.12
5.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15560 (15K) [text/plain]
Saving to: ‘Sheet_1.csv’

Sheet_1.csv         100%[===================>]  15.20K  --.-KB/s
in 0s

2020-07-24 07:19:23 (243 MB/s) - ‘Sheet_1.csv’ saved [15560/15560]
```

Question-1: Load and analyze the data

Tasks to do:

Load the data in a pandas DataFrame

Have a look at the first five rows

Check if the dataset contains any null values

Check the shape of the dataset

Drop the last 5 columns

In [2]:
```
import pandas as pd
df = pd.read_csv('/content/Sheet_1.csv')
pd.set_option('display.max_colwidth', -1)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Fu
tureWarning: Passing a negative integer is deprecated in version 1
.0 and will not be supported in future version. Instead, use None
to not limit the column width.
  This is separate from the ipykernel package so we can avoid doin
g imports until
```

In [3]:
```
df.head()
```

Out[3]:

	response_id	class	response_text	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	U
0	response_1	not_flagged	I try and avoid this sort of conflict	NaN	NaN	NaN	NaN	
1	response_2	flagged	Had a friend open up to me about his mental addiction to weed and how it was taking over his life and making him depressed	NaN	NaN	NaN	NaN	
2	response_3	flagged	I saved a girl from suicide once. She was going to swallow a bunch of pills and I talked her out of it in a very calm, loving way.	NaN	NaN	NaN	NaN	
3	response_4	not_flagged	i cant think of one really...i think i may have indirectly	NaN	NaN	NaN	NaN	
4	response_5	not_flagged	Only really one friend who doesn't fit into the any of the above categories. Her therapist calls it spiraling." Anyway she pretty much calls me any time she is frustrated by something with her boyfriend to ask me if it's logical or not. Before they would just fight and he would call her crazy. Now she asks me if it's ok he didn't say "please" when he said "hand me the remote."		NaN	NaN	NaN	

```
In [4]: df.shape
```

```
Out[4]: (80, 8)
```

```
In [5]: df.isnull().sum()
```

```
Out[5]: response_id       0
        class             0
        response_text     0
        Unnamed: 3        78
        Unnamed: 4        80
        Unnamed: 5        79
        Unnamed: 6        80
        Unnamed: 7        79
        dtype: int64
```

```
In [6]: for i in df.columns[3:]:
          df.drop(i,axis=1,inplace=True)
        df.drop('response_id',axis=1,inplace=True)
```

```
In [7]: df.tail()
```

Out[7]:

	class	response_text
75	not_flagged	Now that I've been through it, although i'm not even where I'd like to be, I'm extremely open about sharing my experience with others and helping friends going through similar situations. And PLEASE if you have any other questions about my situation don't hesitate to email me. I'm an open book and excited to see how many people you're going to help.
76	flagged	when my best friends mom past away from od'ing when he was in grade 5
77	not_flagged	As a camp counselor I provide stability in kids lives who may have troubled home situations.
78	flagged	My now girlfriend used to have serious addiction troubles before we started dating and felt as though her addiction defined her as a person. She thought that all people saw when they looked at her was the addiction. I spent many nights with her talking and letting her vent. I was one of the only people supporting her and she felt as though I could help because I had been in her spot.
79	not_flagged	The one person I ever talked to it was because we were both going through the same thing. Us talking together helped, it was important to realize you aren't alone

Question-2: We cannot use string objects for prediction, so convert categorical feature to numerical values

Tasks to do:

Convert the categorical features to numerical values using Label Encoder from sklearn

In [8]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['class']=le.fit_transform(df['class'])
print(df.head())
print(df['class'].unique())
```

```
   class
response_text
0   1       I try and avoid this sort of conflict
1   0       Had a friend open up to me about his mental addiction to
weed and how it was taking over his life and making him depressed
2   0       I saved a girl from suicide once. She was going to swall
ow a bunch of pills and I talked her out of it in a very calm, lov
ing way.
3   1       i cant think of one really...i think i may have indirect
ly
4   1       Only really one friend who doesn't fit into the any of t
he above categories. Her therapist calls it spiraling." Anyway she
pretty much calls me any time she is frustrated by something with
her boyfriend to ask me if it's logical or not. Before they would
just fight and he would call her crazy. Now she asks me if it's ok
he didn't say "please" when he said  "hand me the remote."
[1 0]
```

Question-3: Implement TF-IDF to create features

Tasks to do:

Implement TF-IDF using sklearn

Convert all the text to lower case

Remove stopwords

Transform the data using TF-IDF and put it into dataframe

In [9]:
```
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(lowercase=True,ngram_range = (1,5),sto
p_words='english')
```

In [10]:
```
X = df['response_text']
y = df['class']
```

In [11]:
```
X_tfidf = vectorizer.fit_transform(X)
df_idf = pd.DataFrame(X_tfidf.toarray(),columns=vectorizer.get_feat
ure_names())
print(df_idf.head())
print(X_tfidf.shape)
```

```
    1n  1n hour  ...  years stayed stopped feeling  years stayed s
topped feeling bad
0  0.0  0.0      ...  0.0                          0.0
1  0.0  0.0      ...  0.0                          0.0
2  0.0  0.0      ...  0.0                          0.0
3  0.0  0.0      ...  0.0                          0.0
4  0.0  0.0      ...  0.0                          0.0

[5 rows x 3508 columns]
(80, 3508)
```

Question-4: Split the data into training and testing datasets

Tasks to do:

Split the dataset using sklearn, with 20% for testing with random_state=7

In [12]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_tfidf, y,test_size
=0.20,random_state = 7)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(64, 3508)
(16, 3508)
(64,)
(16,)
```

In [13]:
```
import warnings
warnings.filterwarnings("ignore")
```

Question-5: Perform K-Fold cross validation for model selection

Tasks to do:

Perform K-fold with K=10 with random_state = 7

Perform K-Fold with commonly used classification algorithm

Calculate the mean score of each iteration

Take the model with highest score

In [14]:
```
from sklearn.model_selection import cross_val_score,KFold
#machine learning algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB

models=[]
models.append(('lr',LogisticRegression()))
models.append(('decision tree',DecisionTreeClassifier()))
models.append(('svm',SVC(gamma='auto')))
models.append(('knn',KNeighborsClassifier()))
models.append(('gaussian naive bayes',GaussianNB()))
models.append(('Random Forest',RandomForestClassifier()))
models.append(('multinomial naive bayes',MultinomialNB()))


for name,model in models:
  kfold=KFold(n_splits=10,random_state=7)
  cross_val_sc=cross_val_score(model,X_tfidf,y,scoring='accuracy',c
v=kfold)
  print('{} : acc: {}(standard deviation: {})'.format(name,cross_va
l_sc.mean(),cross_val_sc.std()))
```

```
lr : acc: 0.6875(standard deviation: 0.128086884574495)
decision tree : acc: 0.6875(standard deviation: 0.1505199322349037
)
svm : acc: 0.6875(standard deviation: 0.128086884574495)
knn : acc: 0.6625(standard deviation: 0.15860721925561902)
gaussian naive bayes : acc: nan(standard deviation: nan)
Random Forest : acc: 0.6875(standard deviation: 0.128086884574495)
multinomial naive bayes : acc: 0.6875(standard deviation: 0.128086
884574495)
```

Multinomial Naive Bayes, SVM, Logistic regression, VM can be used

Question-6: Train a multinomial Naive Bayes model, perform prediction and calculate accuracy score

```
In [15]: from sklearn.metrics import accuracy_score, classification_report
         model = MultinomialNB()
         model.fit(X_train, y_train)

         y_pred = model.predict(X_test)
         score = accuracy_score(y_test, y_pred)
         print ("Accuracy: ",score)
```

```
Accuracy:  0.6875
```

Scenario 2: Categorizing Tweets Using Natural Language Processing

People post on Twitter very frequently. Sometimes these posts are about disasters. It could be a help to the police if they can use this information to identify and respond quickly to disasters happening in the area?

Problem Statement:

You as a data scientist have been provided a twitter data to create a model which can predict if a tweet(for the police) can be of some interest or not?

Data Description:

The dataset contains 3 columns:

- text: It contain the tweets
- choose_one: It contains if the data is relevant, not relevant, or can't decide
- class_label: labels for the choose_one column
    - 0: Not relevant
    - 1: Relevant
    - 2: Can't Decide

Tasks to be performed:

1. Load and analyze the dataset - Beginner
2. Pre-process text data - Intermediate
3. Tokenize the data - Beginner
4. Split the dataset for training and testing - Beginner
5. Analyze text data, like the size of our vocabulary- Beginner
6. Create word2vec word embedding using gensim - Intermediate
7. Generate features using word embeddings - Intermediate
8. Perform K-fold cross validation - Beginner
9. Train a Random Forest Classifier model - Beginner
10. Perform prediction and evaluate the model using Confusion metrics and accuracy score - Beginner

Topics Covered:

Data collection

Pre-processing text data

Word2vec word embeddings

Train/Test Algorithms

Predicting using the trained model

Evaluating a model: Confusion metrics and accuracy score

fetch and download the data

```
In [ ]: !wget https://www.dropbox.com/s/rsdr34l9xk9yean/socialmedia_relevan
        t_cols.csv
```

```
--2020-06-18 05:48:23--  https://www.dropbox.com/s/rsdr34l9xk9yean
/socialmedia_relevant_cols.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.3.1, 2620:1
00:6018:1::a27d:301
Connecting to www.dropbox.com (www.dropbox.com)|162.125.3.1|:443..
. connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/rsdr34l9xk9yean/socialmedia_relevant_cols.csv [fo
llowing]
--2020-06-18 05:48:23--  https://www.dropbox.com/s/raw/rsdr34l9xk9
yean/socialmedia_relevant_cols.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucc76ce3e284566513c37e842494.dl.dropboxuserconte
nt.com/cd/0/inline/A53ENZ1_2cMUNEGvn1cdtU5lxGn12JgJsQZJWOsYZgGSH73
exkjMxRie4XgBLJiadK45fkll63hrVVOPWSgRM0gm-YcLH-_JuAaxFFrshKKRM2vb0
tbeb8s2NoFY_mEzVEk/file# [following]
--2020-06-18 05:48:23--  https://ucc76ce3e284566513c37e842494.dl.d
ropboxusercontent.com/cd/0/inline/A53ENZ1_2cMUNEGvn1cdtU5lxGn12JgJ
sQZJWOsYZgGSH73exkjMxRie4XgBLJiadK45fkll63hrVVOPWSgRM0gm-YcLH-_JuA
axFFrshKKRM2vb0tbeb8s2NoFY_mEzVEk/file
Resolving ucc76ce3e284566513c37e842494.dl.dropboxusercontent.com (
ucc76ce3e284566513c37e842494.dl.dropboxusercontent.com)... 162.125
.3.15, 2620:100:6018:15::a27d:30f
Connecting to ucc76ce3e284566513c37e842494.dl.dropboxusercontent.c
om (ucc76ce3e284566513c37e842494.dl.dropboxusercontent.com)|162.12
5.3.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1270924 (1.2M) [text/plain]
Saving to: ‘socialmedia_relevant_cols.csv’

socialmedia_relevan 100%[===================>]   1.21M  --.-KB/s
in 0.06s

2020-06-18 05:48:24 (19.1 MB/s) - ‘socialmedia_relevant_cols.csv’
saved [1270924/1270924]
```

Question-1: Load and anlayze the dataset

Tasks to do:

Load the data in a pandas DataFrame

Have a look at the first five rows

Check the shape of the dataset

Check unique values of the last two columns

Check the distribution of instances among different labels('class_label' column)

```
In [ ]: with open('/content/socialmedia_relevant_cols.csv', encoding="utf8"
        , errors='ignore') as f:
          text = f.read()
```

```
In [ ]: file = open('social_media.csv','w')
        file.write(text)
        file.close
```

```
Out[ ]: <function TextIOWrapper.close>
```

```
In [ ]: import pandas as pd
        df1 = pd.read_csv('/content/social_media.csv')
```

```
In [ ]: pd.set_option('display.max_colwidth', -1)
```

```
In [ ]: df1.head(10)
```

Out[ ]:

	text	choose_one	class_label
0	Just happened a terrible car crash	Relevant	1
1	Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all	Relevant	1
2	Heard about #earthquake is different cities, stay safe everyone.	Relevant	1
3	there is a forest fire at spot pond, geese are fleeing across the street, I cannot save them all	Relevant	1
4	Forest fire near La Ronge Sask. Canada	Relevant	1
5	All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are expected	Relevant	1
6	13,000 people receive #wildfires evacuation orders in California	Relevant	1
7	Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours into a school	Relevant	1
8	#RockyFire Update => California Hwy. 20 closed in both directions due to Lake County fire - #CAfire #wildfires	Relevant	1
9	Apocalypse lighting. #Spokane #wildfires	Relevant	1

```
In [ ]: print(df1.choose_one.unique())
        print(df1.class_label.unique())
```

```
['Relevant' 'Not Relevant' "Can't Decide"]
[1 0 2]
```

```
In [ ]: df1.groupby(['choose_one','class_label']).count()
Out[ ]:
```

choose_one	class_label	text
Can't Decide	2	16
Not Relevant	0	6187
Relevant	1	4673

We can see the data is evenly distributed between 'Relevant' and 'Not Relevant' column

```
In [ ]: df1.isnull().sum()
Out[ ]: text           0
        choose_one     0
        class_label    0
        dtype: int64
```

We can see there are no null values

Question-2: Pre-process text data

Tasks to do:

> Convert text data to lowercase
>
> Remove url from the tweets ('text' column)
>
> Keep only those words which have only alphabets, Remove punctuations and special symbols
>
> Remove stopwords and words with less than 3 letters
>
> Create a new feature with this clean text data

```
In [ ]: import nltk
        nltk.download('stopwords')
        from nltk.corpus import stopwords
        stopword=stopwords.words('english')

        [nltk_data] Downloading package stopwords to /root/nltk_data...
        [nltk_data]   Unzipping corpora/stopwords.zip.
```

```
In [ ]: df1['text']=df1['text'].str.lower()
```

In [ ]:
```
import re
#remove urls
df1['text']=df1['text'].apply(lambda x:re.sub(r'(http)+\S*','',x))
#remove anything except a-z
df1['text']=df1['text'].apply(lambda x:re.sub(r'[^a-z\s]','',x))
#remove stopwords and words with less than 3 letters
li=list()
for i in df1['text'][:]:
  words=i.split()
  t=[word for word in words if word not in stopword and len(word)>2
]
  li.append(' '.join(t))
```

In [ ]:
```
df1['clean']=li
df1.head()
```

Out[ ]:

	text	choose_one	class_label	clean
0	just happened a terrible car crash	Relevant	1	happened terrible car crash
1	our deeds are the reason of this earthquake may allah forgive us all	Relevant	1	deeds reason earthquake may allah forgive
2	heard about earthquake is different cities stay safe everyone	Relevant	1	heard earthquake different cities stay safe everyone
3	there is a forest fire at spot pond geese are fleeing across the street i cannot save them all	Relevant	1	forest fire spot pond geese fleeing across street cannot save
4	forest fire near la ronge sask canada	Relevant	1	forest fire near ronge sask canada

Question-3: Tokenize the data

Tasks to do:

Tokenize the cleaned text using RegexpTokenizer

Create new feature and save the tokens

In [ ]:
```
from nltk.tokenize import RegexpTokenizer
regex = RegexpTokenizer(r'\w+')
df1['tokens'] = df1.clean.apply(lambda x : regex.tokenize(x))
```

```
In [ ]: df1.head()
```

Out[ ]:

	text	choose_one	class_label	clean	tokens
0	just happened a terrible car crash	Relevant	1	happened terrible car crash	[happened, terrible, car, crash]
1	our deeds are the reason of this earthquake may allah forgive us all	Relevant	1	deeds reason earthquake may allah forgive	[deeds, reason, earthquake, may, allah, forgive]
2	heard about earthquake is different cities stay safe everyone	Relevant	1	heard earthquake different cities stay safe everyone	[heard, earthquake, different, cities, stay, safe, everyone]
3	there is a forest fire at spot pond geese are fleeing across the street i cannot save them all	Relevant	1	forest fire spot pond geese fleeing across street cannot save	[forest, fire, spot, pond, geese, fleeing, across, street, cannot, save]
4	forest fire near la ronge sask canada	Relevant	1	forest fire near ronge sask canada	[forest, fire, near, ronge, sask, canada]

Question-4: Split the data into training and testing datasets

Tasks to do:

Split the dataset using sklearn, with 20% for testing with random_state=7

```
In [ ]: from sklearn.model_selection import train_test_split
        df1.drop('choose_one',axis=1,inplace=True)
        X = df1.drop('class_label',axis=1)
        y = df1["class_label"]

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
        =0.2, random_state=7)

        print("Shape of Training set",X_train.shape)
        print("Shape of Testing set",X_test.shape)
```

```
Shape of Training set (8700, 3)
Shape of Testing set (2176, 3)
```

Question-5: Analyze text data

Tasks to do:

> Find the size of vocabulary of the text data
>
> Print the 20 most common words

In [ ]:
```
complete=' '.join(X_train['clean'])
vocab=set(complete.split())
len(vocab)
```

Out[ ]:
```
17967
```

In [ ]:
```
from collections import Counter
count_vocab = Counter(complete.split())

# get the top 100 most common occuring words
count_vocab.most_common(20)
```

Out[ ]:
```
[('like', 379),
 ('amp', 325),
 ('fire', 286),
 ('get', 274),
 ('via', 265),
 ('new', 254),
 ('news', 239),
 ('one', 231),
 ('people', 215),
 ('dont', 208),
 ('video', 185),
 ('emergency', 184),
 ('disaster', 173),
 ('would', 168),
 ('police', 164),
 ('time', 145),
 ('still', 142),
 ('suicide', 140),
 ('body', 139),
 ('storm', 136)]
```

Question-6: Create word embeddings

Tasks to do:

Use gensim library to train word2vec model on training data

The size of the word embeddings be a vector of 200 X 1

Use skip-gram model of word2vec

Keep only the words which are occurring atleast 5 times in the dataset

Save the trained word2vec model

In [ ]:
```
from gensim.models import Word2Vec
model_vec_train = Word2Vec(sentences=X_train['tokens'], size=200, w
indow=5, min_count=5, workers=-1, sg=1)
model_vec_train.save("word2vec.model")
```

In [ ]:
```
model_vec_train['like'].shape
```

Out[ ]:
```
(200,)
```

In [ ]:
```
X_train.head()
```

Out[ ]:

	text	clean	tokens
63	soooo pumped for ablaze southridgelife	soooo pumped ablaze southridgelife	[soooo, pumped, ablaze, southridgelife]
1460	good diss bad beat and flow mark my words meek mill is body bagging him once he responds patient patient meek is a battle rapper	good diss bad beat flow mark words meek mill body bagging responds patient patient meek battle rapper	[good, diss, bad, beat, flow, mark, words, meek, mill, body, bagging, responds, patient, patient, meek, battle, rapper]
6095	im melting a bar of chocolate under my laptop at least this fucking hellfire is good for something	melting bar chocolate laptop least fucking hellfire good something	[melting, bar, chocolate, laptop, least, fucking, hellfire, good, something]
7610	pandemoniumiso psp	pandemoniumiso psp	[pandemoniumiso, psp]
221	usama bin ladins family dead in airplane crash naturally no accident	usama bin ladins family dead airplane crash naturally accident	[usama, bin, ladins, family, dead, airplane, crash, naturally, accident]

Question-7: Generate features using word embeddings for training and testing set

Tasks to do:

Extract word embedding from the word2vec model

Extract the word embeddings of all the words in a tweet

Take a sum of those word embedding vectors along axis 0

Use the average as a feature by dividing the sum with total number of words in that tweet

```
In [ ]: import numpy as np
        def get_embeddings(sent_token, model=model_vec_train):
          vector = [model[word] if word in model else np.zeros(200) for wor
        d in sent_token]
          l = len(vector)
          s = np.sum(vector, axis=0)
          avg = s/l
          return avg
```

```
In [ ]: def generate_embeddings(data, model=model_vec_train):
          embeddings = data.apply(lambda x:get_embeddings(x,model))
          return(embeddings)
```

```
In [ ]: X_train.head()
```

Out[ ]:

	text	clean	tokens
63	soooo pumped for ablaze southridgelife	soooo pumped ablaze southridgelife	[soooo, pumped, ablaze, southridgelife]
1460	good diss bad beat and flow mark my words meek mill is body bagging him once he responds patient patient meek is a battle rapper	good diss bad beat flow mark words meek mill body bagging responds patient patient meek battle rapper	[good, diss, bad, beat, flow, mark, words, meek, mill, body, bagging, responds, patient, patient, meek, battle, rapper]
6095	im melting a bar of chocolate under my laptop at least this fucking hellfire is good for something	melting bar chocolate laptop least fucking hellfire good something	[melting, bar, chocolate, laptop, least, fucking, hellfire, good, something]
7610	pandemoniumiso psp	pandemoniumiso psp	[pandemoniumiso, psp]
221	usama bin ladins family dead in airplane crash naturally no accident	usama bin ladins family dead airplane crash naturally accident	[usama, bin, ladins, family, dead, airplane, crash, naturally, accident]

```
In [ ]: embeddings_train=generate_embeddings(X_train['tokens'])
        embeddings_test=generate_embeddings(X_test['tokens'])
```

Converting the output word embeddings as dataframe

```
In [ ]: d=dict()
        for i in range(200):
          l=[]
          for j in range(8700):
            try:
              l.append(embeddings_train.values[j][i])
            except:
              l.append(0)
          d[i]=l
        train = pd.DataFrame(d)
```

```
In [ ]: dic=dict()
        for i in range(200):
          l=[]
          for j in range(2176):
            try:
              l.append(embeddings_test.values[j][i])
            except:
              l.append(0)
          dic[i]=l
        test = pd.DataFrame(dic)
```

```
In [ ]: train.shape

Out[ ]: (8700, 200)
```

```
In [ ]: train.head()

Out[ ]:
```

	0	1	2	3	4	5	6	7
0	0.000219	0.000077	0.000392	-0.000485	-0.000026	-0.000143	0.000041	0.000088
1	-0.000451	-0.000234	-0.000394	-0.000278	0.000185	0.000073	0.000174	-0.000576
2	0.000479	-0.000741	0.000686	-0.000071	-0.000148	-0.000066	0.000441	0.000345 -
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	-0.000405	-0.000471	0.001033	0.000017	-0.000196	0.000118	-0.000462	0.000359

5 rows × 200 columns

```
In [ ]: test.shape

Out[ ]: (2176, 200)
```

Question-8: Perform K-Fold cross validation for model selection

Tasks to do:

Perform K-fold with K=10 with random_state = 7

Perform K-Fold with commonly used classification algorithm

Calculate the mean score of each iteration

Take the model with highest score

In [ ]:

```
from sklearn.model_selection import cross_val_score,KFold
#machine learning algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

models=[]
models.append(('lr',LogisticRegression()))
models.append(('decision tree',DecisionTreeClassifier()))
models.append(('svm',SVC(gamma='auto')))
models.append(('knn',KNeighborsClassifier()))
models.append(('gaussian naive bayes',GaussianNB()))
models.append(('Random Forest',RandomForestClassifier()))



for name,model in models:
  kfold=KFold(n_splits=10,random_state=7)
  cross_val_sc=cross_val_score(model,train,y_train,scoring='accurac
y',cv=kfold)
  print('{} : acc: {}(standard deviation: {})'.format(name,cross_va
l_sc.mean(),cross_val_sc.std()))
```

```
lr : acc: 0.5648275862068965(standard deviation: 0.018109924037913
682)
decision tree : acc: 0.6183908045977011(standard deviation: 0.0143
7471098929977)
svm : acc: 0.5648275862068965(standard deviation: 0.01810992403791
3682)
knn : acc: 0.7071264367816091(standard deviation: 0.01679734736747
73)
gaussian naive bayes : acc: 0.48310344827586205(standard deviation
: 0.01184577424872922)
Random Forest : acc: 0.7126436781609196(standard deviation: 0.0164
33164138539322)
```

Accuracy of random forest is highest. So we will use random forest classifier

Question-9: Train the model

Tasks to do:

> Train a random forest classifier model for prediction
>
> Also, check the score of the model on training set

```
In [ ]: from sklearn.ensemble import RandomForestClassifier
        final_model= RandomForestClassifier()
        final_model.fit(train,y_train)
        final_model.score(train,y_train)

Out[ ]: 0.9827586206896551
```

Model score on training set is quite high

Question-10: Evaluate the model on the test data

> Print confusion matrix of the test data
>
> Also, find the accuracy score

```
In [ ]: y_pred = final_model.predict(test)
```

```
In [ ]: from sklearn.metrics import confusion_matrix,accuracy_score
        print(confusion_matrix(y_test,y_pred))
        print(accuracy_score(y_test,y_pred))

        [[1158  115    0]
         [ 527  372    0]
         [   3    1    0]]
        0.703125
```

```
In [ ]:
```