Project Report

Artificial Intelligence

CS-415-A

Group Members:

Muhammad Nadeem	201980050
Muniem Billah	191400021
Muhammad Danish Javed	191400055
Abdul Haseeb	191400011
Muhammad Haris	191400035

Topic Name:

Twitter Sentiment Analysis.

Here is complete explanation:

First we Scrape Real time data:

For Real Time Tweets Scrape

```
In [2]:
import snscrape.modules.twitter as sntwitter
import pandas as pd
```

```
In [3]: query="(from:elonmusk) until:2020-02-02 since:2010-02-01"
tweets=[]
limit=1000

for tweet in sntwitter.TwitterSearchScraper(query).get_items():
    #print(vars(tweet))
    #break
    if len(tweets)==limit:
        break
    else:
        tweets.append([tweet.date,tweet.username,tweet.content,tweet.retweets,tweet.likes])
df=pd.DataFrame(tweets,columns=['date','username','content','retweets','likes'])
df.to_csv("Dataset.csv")
```

Then we import some libraries that we are used:

```
In [4]: import snscrape.modules.twitter as sntwitter
        import pandas as pd
        import numpy as np
        import nltk
        from nltk.corpus import stopwords
        from nltk.stem import PorterStemmer
        from nltk.tokenize import word_tokenize
```

Here we read file:

```
In [6]: #Reading csv file
        dataset= pd.read_csv("dataset.csv")
        dataset
```

Out[6]:

	date	username	content	retweets	likes
0	2022-10-31 08:42:02+00:00	CMShehbaz	I extend my congratulations to H.E. Luiz Ináci...	644	2436
1	2022-10-30 14:07:08+00:00	ImranKhanPTI	Shocked & deeply saddened by the terrible ...	24416	83314
2	2022-10-30 08:36:47+00:00	CMShehbaz	I am saddened at the tragic death of 146 peopl...	862	5045
3	2022-10-29 17:37:22+00:00	ImranKhanPTI	For all those spreading rumours about my mtg i...	28635	86544
4	2022-10-29 07:15:54+00:00	CMShehbaz	On the 99th Republic Day of Turkiye, I extend ...	622	2883

Then we find dataTypes of each column and dimension of columns:

```
In [8]: #shape of the dataset
        dataset.shape

Out[8]: (9000, 5)
```

```
In [9]: #DataSet Information
        dataset.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 9000 entries, 0 to 8999
        Data columns (total 5 columns):
         #   Column    Non-Null Count  Dtype
        ---  ------    --------------  -----
         0   date      9000 non-null   object
         1   username  9000 non-null   object
         2   content   9000 non-null   object
         3   retweets  9000 non-null   int64
         4   likes     9000 non-null   int64
        dtypes: int64(2), object(3)
        memory usage: 351.7+ KB
```

```
In [10]: #Data Types of all columns
         dataset.dtypes

Out[10]: date        object
         username    object
         content     object
         retweets     int64
         likes        int64
         dtype: object
```

```
In [11]: #total rows and columns
         print('Count of columns in the data is:  ', len(dataset.columns))
         print('Count of rows in the data is:  ', len(dataset))

         Count of columns in the data is:   5
         Count of rows in the data is:   9000
```

Then we check frequency of null values in each column:

```
In [12]: #checking null values
         dataset.isnull().sum()

Out[12]: date        0
         username    0
         content     0
         retweets    0
         likes       0
         dtype: int64
```

Then we Apply Tokenization:

Tokenization

In [13]:
```
import nltk
from nltk.tokenize import TweetTokenizer
```

In [14]:
```
# Initialize the tokenizer
tokenizer = TweetTokenizer()

# Apply tokenization on the clean_tweet column
dataset["clean_tweet_tokenized"] = dataset["content"].apply(lambda x: tokenizer.tokenize(x))

# Print the tokenized tweets
print(dataset["clean_tweet_tokenized"])
```

```
0       [I, extend, my, congratulations, to, H, ., E, ...
1       [Shocked, &, deeply, saddened, by, the, terrib...
2       [I, am, saddened, at, the, tragic, death, of, ...
3       [For, all, those, spreading, rumours, about, m...
4       [On, the, 99th, Republic, Day, of, Turkiye, ,,...
                              ...
8995    [Had, a, quick, look, -, interesting, ., Pl, c...
8996    [@TabishChawla, Not, really, -, we, are, not, ...
8997    [Pak, is, facing, internal, /, external, chall...
8998    [@DaBieberLoverr, Thanks, -, all, praise, be, ...
8999    [Missed, it, -, but, sublime, performance, onc...
Name: clean_tweet_tokenized, Length: 9000, dtype: object
```

Tokenization is the most common method is to split text based on whitespace and punctuation. For example, the sentence "I love NLP!" can be tokenized into the tokens "I", "love", "NLP", and "!". However, tokenization can be more complex for languages that do not use whitespace or punctuation in a consistent way.

In [15]:
```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

import re
import pandas as pd
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())
```

```
[nltk_data] Downloading package vader_lexicon to C:\Users\Mr.
[nltk_data]     Nadeem\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package words to C:\Users\Mr.
[nltk_data]     Nadeem\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

In [16]:
```

sentence = dataset['content'][0]
sid.polarity_scores(sentence)['compound']
```

Out[16]:
```
0.9337
```

In [17]:
```
#The output of the code above is -0.6249, indicating that the sentence is of negative sentiment.
```

nltk.download('vader_lexicon'): This line downloads the VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon, which is a pre-trained sentiment analysis tool included in nltk.

from nltk.sentiment.vader import SentimentIntensityAnalyzer: This line imports the SentimentIntensityAnalyzer class from the vader module in nltk.

sid = SentimentIntensityAnalyzer(): This line creates an instance of the SentimentIntensityAnalyzer class, which is used to analyze the sentiment of text.

import re: This line imports the regular expression (regex) library, which can be used to manipulate strings.

import pandas as pd: This line imports the pandas library, which is a popular tool for data manipulation and analysis.

nltk.download('words'): This line downloads a list of English words from the nltk corpus.

words = set(nltk.corpus.words.words()): This line creates a set of English words from the nltk corpus.

sentence = dataset['content'][0]: This line selects the first row of the content column from a dataset and assigns it to the variable sentence. The dataset variable is assumed to be a pandas DataFrame.

sid.polarity_scores(sentence)['compound']: This line uses the polarity_scores method of the SentimentIntensityAnalyzer class to calculate a sentiment score for sentence. The compound score is returned, which is a value between -1 and 1 that indicates the overall sentiment of the text. A score of -1 indicates very negative sentiment, a score of 0 indicates neutral sentiment, and a score of 1 indicates very positive sentiment.

Now Going to Data Cleaning:

Data Cleaning

```
In [18]: def cleaner(content):
             content = re.sub("@[A-Za-z0-9]+", "", content) # Remove @ sign
             content = re.sub(r"(?:\@|http?\://|https?\://|www)\S+", "", content) # Remove http links
             content = " ".join(content.split())
             content = content.replace("#", "").replace("_", " ") # Remove hashtag sign but keep the text
             content = " ".join(re.findall(r'\b\w+\b', content.lower())) # Remove non-alphabetic characters and convert to lowercase
             return content

         dataset['clean_tweet'] = dataset['content'].apply(cleaner)
```

```
In [19]: print(dataset['clean_tweet'])

         0       i extend my congratulations to h e luiz inácio...
         1       shocked amp deeply saddened by the terrible ac...
         2       i am saddened at the tragic death of 146 peopl...
         3       for all those spreading rumours about my mtg i...
         4       on the 99th republic day of turkiye i extend o...
                                       ...
         8995    had a quick look interesting pl come and see m...
         8996    not really we are not perfect but we have trie...
         8997    pak is facing internal external challenges pml...
```

This is a Python function called cleaner that takes a string as input and performs several text cleaning operations, including removing Twitter usernames, URLs, non-alphabetic characters, and hashtag symbols. The cleaned text is then converted to lowercase and returned.

The function is then applied to the content column of a pandas DataFrame called dataset using the apply method, and the cleaned text is stored in a new column called clean_tweet.

```
In [20]: from textblob import TextBlob

         def get_tweet_sentiment(clean_tweet):
             # create TextBlob object of the tweet
             blob = TextBlob(clean_tweet)

             # get sentiment polarity (ranges from -1 to 1)
             sentiment_polarity = blob.sentiment.polarity

             # assign label based on sentiment polarity
             if sentiment_polarity > 0:
                 return "positive"
             elif sentiment_polarity < 0:
                 return "negative"
             else:
                 return "neutral"
```

```
In [21]: # create a new 'label' column and assign labels to each tweet
         dataset['Sentiment'] = dataset['clean_tweet'].apply(get_tweet_sentiment)

         print(dataset[['clean_tweet', 'Sentiment']])

                                                   clean_tweet Sentiment
         0    i extend my congratulations to h e luiz inácio...  positive
         1    shocked amp deeply saddened by the terrible ac...  negative
         2    i am saddened at the tragic death of 146 peopl...  negative
         3    for all those spreading rumours about my mtg i...  positive
         4    on the 99th republic day of turkiye i extend o...  positive
```

This is a Python function called get_tweet_sentiment that takes a string as input, which is assumed to be a cleaned tweet. The function uses the TextBlob library to perform sentiment analysis on the tweet by creating a TextBlob object and getting its sentiment polarity, which is a value between -1 and 1 that indicates the sentiment of the text.

The function then assigns a label to the tweet based on its sentiment polarity. If the polarity is greater than 0, the tweet is labeled as "positive". If the polarity is less than 0, the tweet is labeled as "negative". If the polarity is 0, the tweet is labeled as "neutral".

The sentiment polarity is calculated by taking the average sentiment score of all the words in the tweet, using a pre-trained model included in the TextBlob library. The model assigns a sentiment score to each word based on its context and meaning, and then calculates the overall sentiment polarity of the text based on the scores of all the words.

This is a Python script that adds a new column called Sentiment to a pandas DataFrame called dataset, which is assumed to contain cleaned tweets. The get_tweet_sentiment function is applied to the clean_tweet column using the apply method, and the sentiment label of each tweet is stored in the new Sentiment column.

The script then prints a subset of the dataset DataFrame, including the clean_tweet and Sentiment columns, to the console using the print statement. This allows the user to see the cleaned text of each tweet and its corresponding sentiment label.

```
In [22]: new_df = pd.DataFrame(dataset[['clean_tweet', 'Sentiment']])
         new_df
```

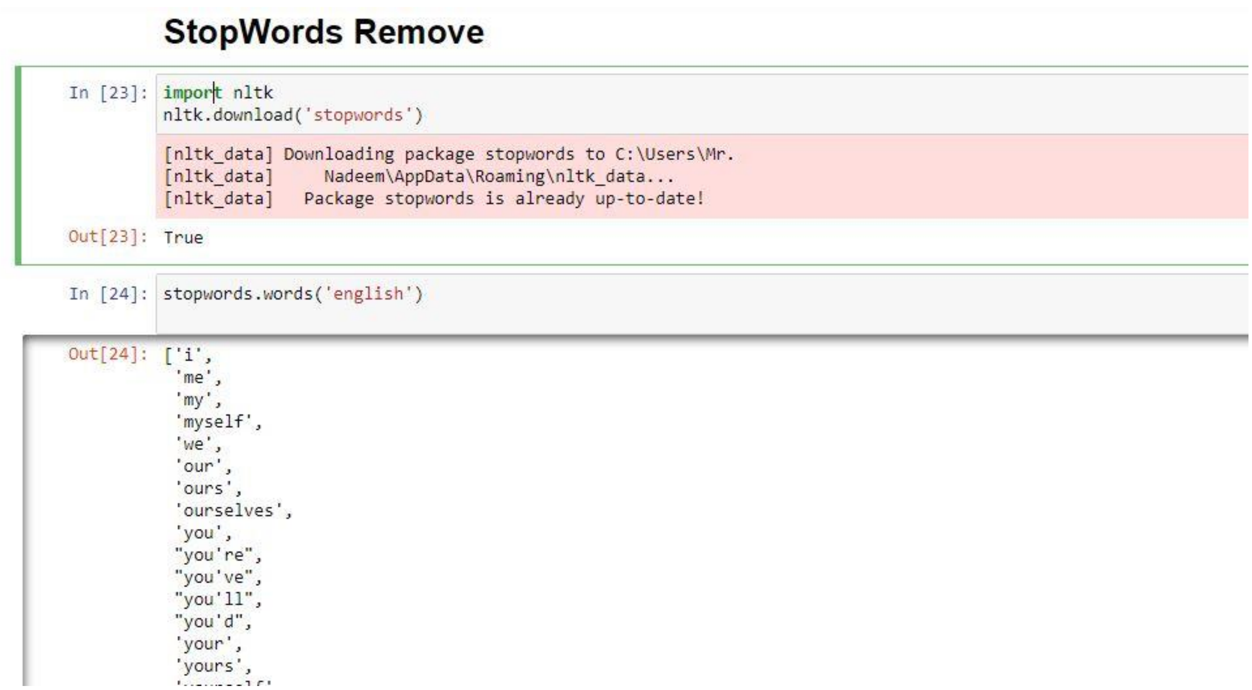
Out[22]:

	clean_tweet	Sentiment
0	i extend my congratulations to h e luiz inácio...	positive
1	shocked amp deeply saddened by the terrible ac...	negative
2	i am saddened at the tragic death of 146 peopl...	negative
3	for all those spreading rumours about my mtg i...	positive
4	on the 99th republic day of turkiye i extend o...	positive
...	...	...
8995	had a quick look interesting pl come and see m...	positive
8996	not really we are not perfect but we have trie...	positive
8997	pak is facing internal external challenges pml...	neutral
8998	thanks all praise be to allah who is giving us...	positive

This is a Python script that creates a new pandas DataFrame called new_df, which is a subset of the dataset DataFrame containing only the clean_tweet and Sentiment columns.

The pd.DataFrame() function is used to create a new DataFrame from the selected columns of the dataset DataFrame. The resulting DataFrame new_df is then printed to the console using the variable name, which displays the contents of the DataFrame. This allows the user to view the cleaned text and sentiment labels of each tweet in a more concise format.

Now we remove Stopwords:



After Stopwords remove we get:

Stopwords

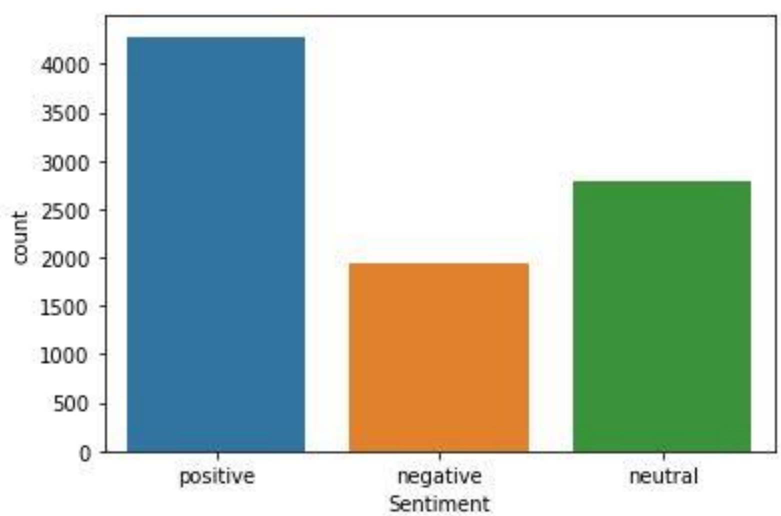
```
In [25]: import nltk
         from nltk.corpus import stopwords
         from nltk.tokenize import word_tokenize

         # Download stopwords
         nltk.download('stopwords')

         # Tokenize the text data
         tokens = new_df['clean_tweet'].apply(word_tokenize)

         # Remove stopwords from the tokens
         stop_words = set(stopwords.words('english'))
         tokens_without_stopwords = tokens.apply(lambda x: [word for word in x if word.lower() not in stop_words])
         print(tokens_without_stopwords)
```

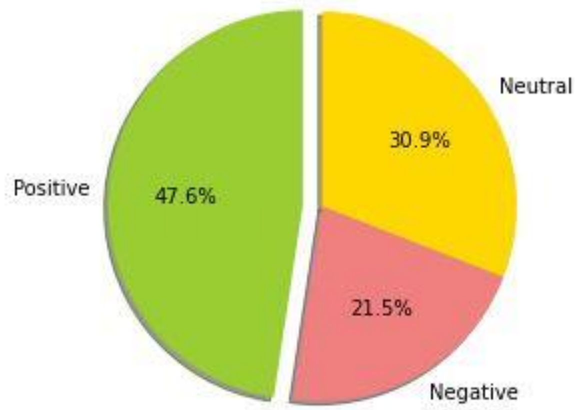
```
[nltk_data] Downloading package stopwords to C:\Users\Mr.
[nltk_data]     Nadeem\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
0       [extend, congratulations, h, e, luiz, inácio, ...
1       [shocked, amp, deeply, saddened, terrible, acc...
2       [saddened, tragic, death, 146, people, stamped...
3       [spreading, rumours, mtg, lahore, reason, retu...
4       [99th, republic, day, turkiye, extend, heartie...
                              ...
8995    [quick, look, interesting, pl, come, see, pak,...
8996                  [really, perfect, tried, sincerely]
```

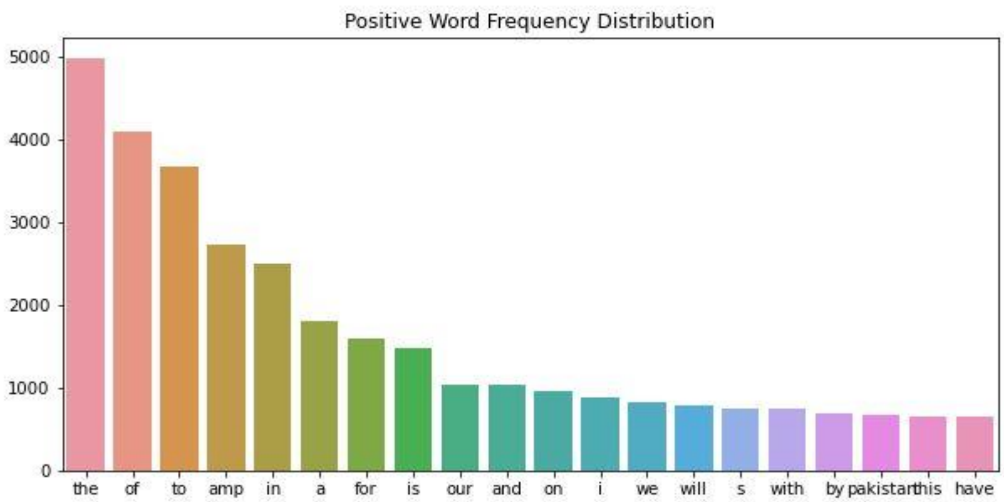
Then We going into **Exploratory Data Analysis(EDA)**:

Here is some visualizations:

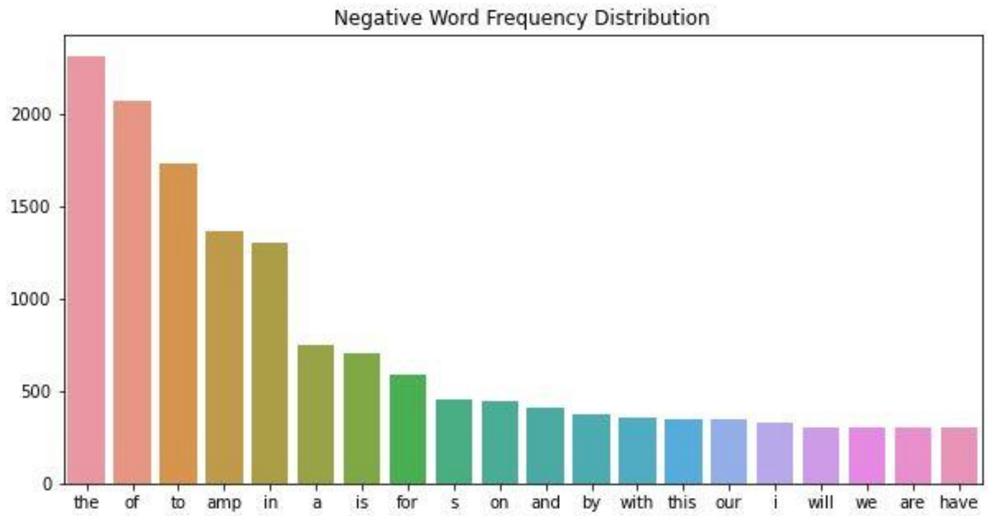




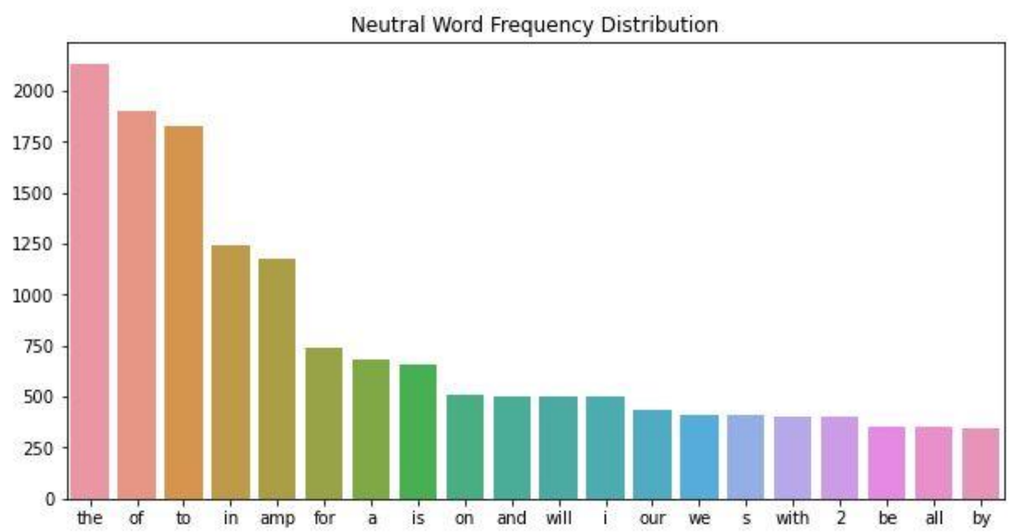
Out[30]: Text(0.5, 1.0, 'Positive Word Frequency Distribution')



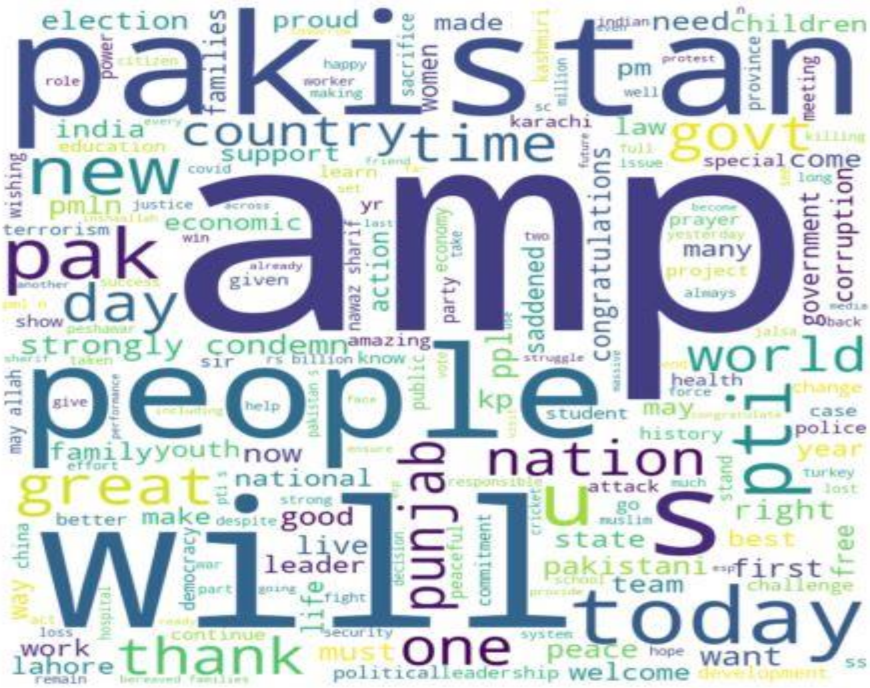
```
Out[31]: Text(0.5, 1.0, 'Negative Word Frequency Distribution')
```



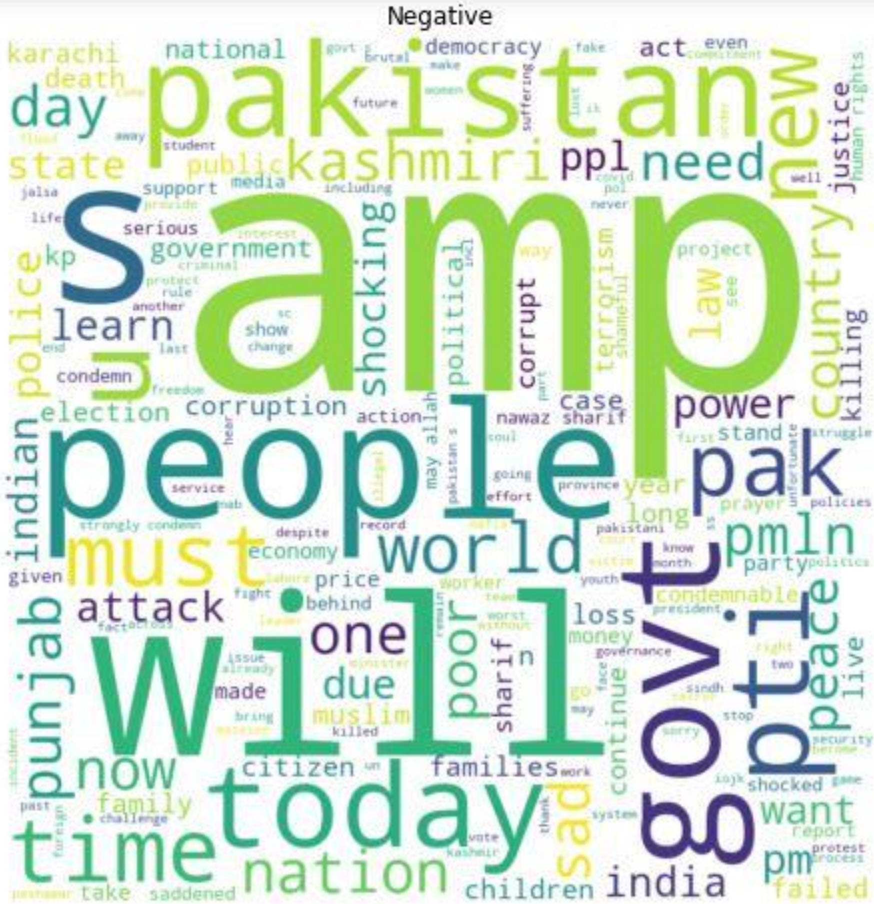

```
Out[32]: Text(0.5, 1.0, 'Neutral Word Frequency Distribution')
```

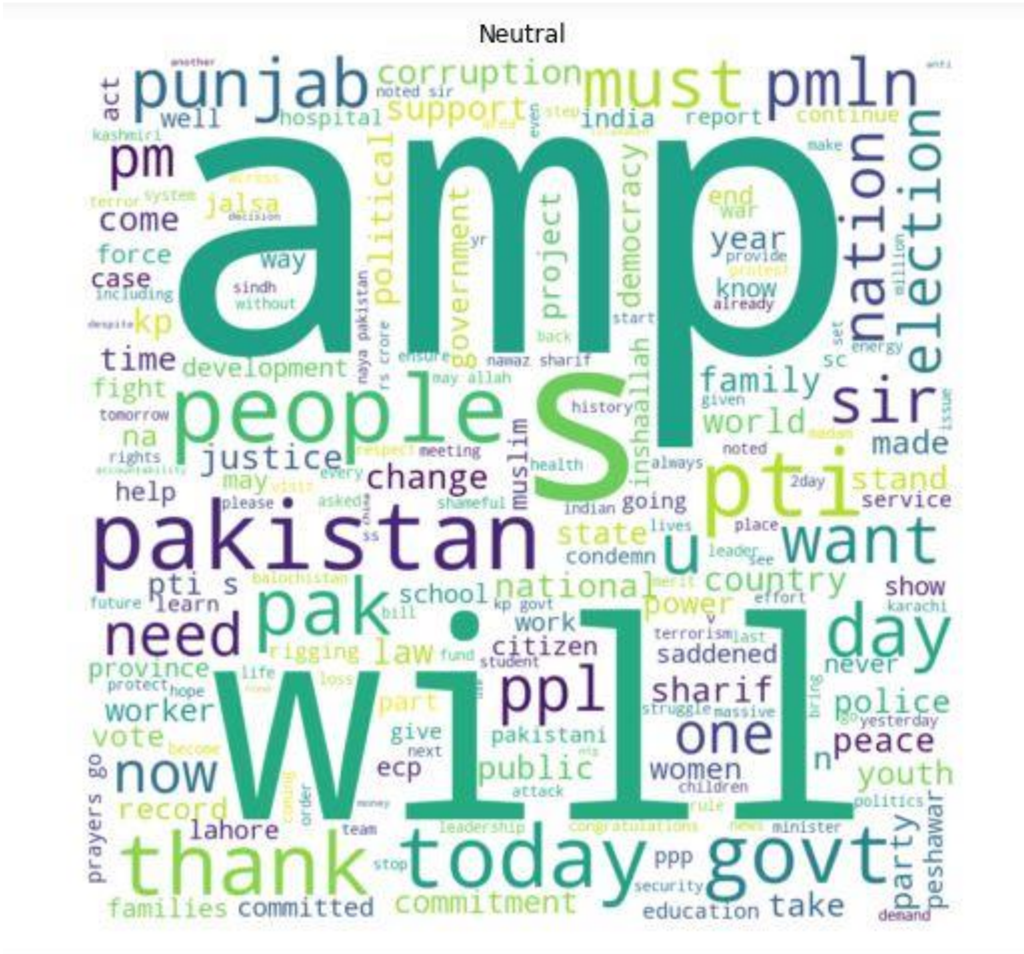


Positive



Negative





Then We find Common Words in Positive, Negative and Neutral:

In [37]:
```
# get the most common words in positive tweets
positive_tweets = ' '.join(new_df[new_df['Sentiment'] == 'positive']['clean_tweet'].tolist())
positive_word_count = get_most_common_words(positive_tweets)
print('Most common words in positive tweets:', positive_word_count)
```

```
Most common words in positive tweets: [('the', 4983), ('of', 4088), ('to', 3676), ('amp', 2730), ('in', 2497), ('a', 1797), ('f
or', 1597), ('is', 1473), ('our', 1039), ('and', 1033)]
```

In [38]:
```
# get the most common words in negative tweets
negative_tweets = ' '.join(new_df[new_df['Sentiment'] == 'negative']['clean_tweet'].tolist())
negative_word_count = get_most_common_words(negative_tweets)
print('Most common words in negative tweets:', negative_word_count)
```

```
Most common words in negative tweets: [('the', 2312), ('of', 2074), ('to', 1730), ('amp', 1362), ('in', 1306), ('a', 746), ('i
s', 703), ('for', 587), ('s', 452), ('on', 447)]
```

In [39]:
```
# get the most common words in neutral tweets
neutral_tweets = ' '.join(new_df[new_df['Sentiment'] == 'neutral']['clean_tweet'].tolist())
neutral_word_count = get_most_common_words(neutral_tweets)
print('Most common words in neutral tweets:', neutral_word_count)
```

```
Most common words in neutral tweets: [('the', 2132), ('of', 1905), ('to', 1826), ('in', 1244), ('amp', 1174), ('for', 737),
('a', 679), ('is', 655), ('on', 507), ('and', 502)]
```

Then We find total number of Positive, Negative and Neutral:

```
In [41]: positive_count = new_df['Sentiment'].value_counts()['positive']
         negative_count = new_df['Sentiment'].value_counts()['negative']
         neutral_count = new_df['Sentiment'].value_counts()['neutral']
         print("Positive Tweets:",positive_count)
         print("Negative Tweets:",negative_count)
         print("Neutral Tweets:",neutral_count)

         Positive Tweets: 4285
         Negative Tweets: 1932
         Neutral Tweets: 2783
```

After This we Implement Multinomial Naïve, SVM, Random Forest:

When we Implement Models when we have ***1000*** tweets Accuracies is:

NB: 65%

Random forest: 62%

SVM: 65%

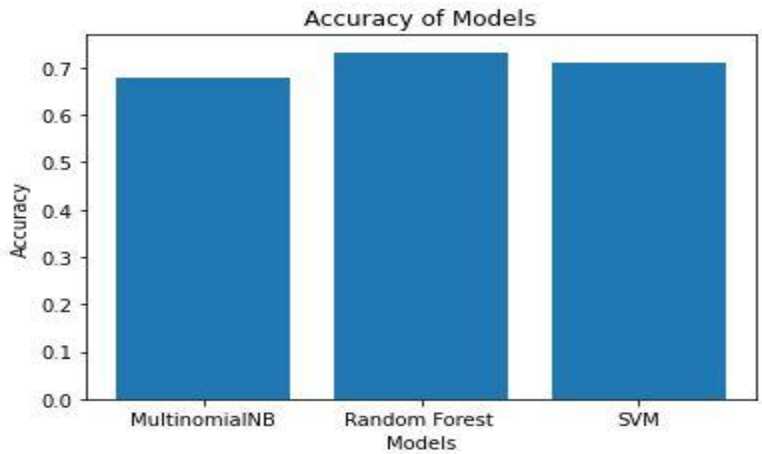
When we Implement Models when we have ***10000*** tweets Accuracies is:

NB:68

Random Forest: 73

SVM:70

Finally plot Accuracies for ***10000 tweets where we get maximum accuracies:***



Justification of our choose these three column are here:

Multinomial Naive Bayes (MultinomialNB): This algorithm is a probabilistic model that is often used for text classification tasks, including sentiment analysis. It works well with high-dimensional sparse data, such as text data, and can be trained quickly even with large datasets. MultinomialNB has been shown to be effective in sentiment analysis tasks, particularly in cases where the text data is short and simple.

Random Forest: This algorithm is an ensemble method that combines multiple decision trees to make predictions. It works well with both categorical and numerical data, and can handle high-dimensional data with many features. Random Forest is often used in sentiment analysis tasks because it can capture non-linear relationships between the input features and the sentiment label. Additionally, Random Forest is less prone to overfitting than some other algorithms.

Support Vector Machines (SVM): This algorithm is a powerful tool for binary classification tasks, including sentiment analysis. It works well with high-dimensional data and can handle both linear and non-linear relationships between input features and the sentiment label. SVM can also handle imbalanced datasets, which is important in sentiment analysis tasks where there may be more instances of one sentiment than another.

Summary:

In summary, each of these algorithms has its own strengths and weaknesses, and the choice of which one to use will depend on the specific task and the characteristics of the dataset. MultinomialNB is often used for simple text classification tasks, while Random Forest and SVM are more powerful and can handle more complex relationships between input features and the sentiment label.

The End