INTRODUCTION: This assignment is basically for sentiment and semantic analysis of text.

PART A-SENTIMENT ANALYSIS OF TWEETS: The basic objective of this is to find out the sentiment of a tweet extracted from [1] as positive, negative, or neutral. The steps involved in the process are explained below:

STEP 1: Collecting the data from twitter API and filtering the text from it by removing any special characters etc. This is done by using the python script in jupyter notebook.

```
import Credentials
from tweepy import API
import csv
from collections import Counter
from tweepy import Cursor
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import heapq
import nltk
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from sklearn.feature_extraction.text import CountVectorizer
import re


import numpy as np
import pandas as pd

class TwitterAuthenticator():

    def authenticate_twitter_app(self):
        auth=OAuthHandler(Credentials.consumer_key,Credentials.consumer_secret)
        auth.set_access_token(Credentials.access_token,Credentials.access_token_secret)
        return auth

class Streamthetweet():
    def get_tweets(self,savitinfile,list_of_tweets):
        listener=Listenthetweets(saveitinfile)
        stream=Stream(auth,listener)
        stream.filter(track=list_of_tweets)
```

Here is the code for getting the data from the twitter API, converting it into a dataframe and then that data frame is converted into a list.

```
    list_of_tweets=["Canada", "University", "Dalhousie University", "Halifax",
"Canada Education"]
    saveitinfile="totalcollectedtweets.csv"
    twitter_client = TwitterClient()
    api = twitter_client.get_twitter_client_api()
    tweets = api.home_timeline(count=3000)
    authenticateit=TwitterAuthenticator()
    authenticateit.authenticate_twitter_app()
    tweet_analyzer = TweetAnalyzer()
    df = tweet_analyzer.tweets_to_data_frame(tweets)
    print(df.head(10))
    dataset=df['Tweets'].to_list()
```

Cleaning the tweets was also done using regex from regular expression. It is shown below:

```
def clean_tweet(self, tweet):
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())
```

STEP 2: Creating the bag of words for each tweet for further analysis. The script written for doing that is given below:

```
v=CountVectorizer()
nltk.download('stopwords')
nltk.download('punkt')
postweet=[]
negtweet=[]
neutratweets=[]
nosentimenttweet=[]
final_df=pd.DataFrame(columns=["Tweet","tweets","Match","Polarity"])
print(final_df.head())
for i in range(len(dataset[:2000])):

    dataset[i] = dataset[i].lower()
    dataset[i] = re.sub(r'\W', ' ', dataset[i])
    dataset[i] = re.sub(r'\s+', ' ', dataset[i])
    dataset[i] = re.sub(r'https\s+', ' ', dataset[i])
    dataset[i] = re.sub(r't co\s+', ' ', dataset[i])
    tweet1=[dataset[i]]

    print(tweet1)

    bagofwords = {}
```

Here we are iterating through the dataset, that are the tweets and in the loop the tweets are cleaned if any other links or characters are left. After that, a dictionary is created that will contain the bag of words, that is the word along with its frequency. The loop for that is shown below:

```
    for data in tweet1:
        words = nltk.word_tokenize(data)
        for word in words:

            if word not in bagofwords.keys():
                bagofwords[word] = 1
            else:
                bagofwords[word] += 1



    for data in tweet1:
        words = nltk.word_tokenize(data)
        for word in words:
            if word in poswords:
                postweet.append(tweet1)
                final_df=final_df.append({"Tweet":i+1,"tweets":tweet1,"Match":word,"Polarity":'positive'},ignore_index=True)
                print(word)
            elif word in negwords:
                negtweet.append(tweet1)
                final_df=final_df.append({"Tweet":i+1,"tweets":tweet1,"Match":word,"Polarity":'negative'},ignore_index=True)
                print(word)
            elif word in neutralwords:
                neutratweets.append(tweet1)
                final_df=final_df.append({"Tweet":i+1,"tweets":tweet1,"Match":word,"Polarity":'neutral'},ignore_index=True)
                print(word)

    print(bagofwords)

    positive_tweets=unique_tweets(postweet)

    negative_tweets=unique_tweets(negtweet)
```

The nltk library is used for separating the words in the list and the bag of words is created as shown by above code.

STEP 3: Comparing bag of words with the list of positive and negative words: The list of positive and negative words is downloaded and stored in a csv file named Polarity2.csv. If a word from the tweet has a polarity value greater than 2, it is a positive word, similarly if a word from the tweet has

polarity value less than -2, it is considered as negative word and if the word from the tweet has the polarity value less than equal to 1 and greater than -1, then it's a neutral word. This is written in code.

```
if __name__ == '__main__':
    df2=pd.read_csv(r'G:\DALHOUSIE Term 1\Data management and ware dalhousie\Polarity2.csv')
    positive=df2[df2['Polarity']>=2].Word
    negative=df2[df2['Polarity']<=-2].Word
    neutral=df2[(df2['Polarity']>=1)&(df2['Polarity']>=-1)].Word
    poswords=positive.to_list()
    negwords=negative.to_list()
    neutralwords=neutral.to_list()
```

```
for data in tweet1:
    words = nltk.word_tokenize(data)
    for word in words:
        if word in poswords:
            postweet.append(tweet1)
            final_df=final_df.append({"Tweet":i+1,"tweets":tweet1,"Match":word,"Polarity":'positive'},ignore_index=True)
            print(word)
        elif word in negwords:
            negtweet.append(tweet1)
            final_df=final_df.append({"Tweet":i+1,"tweets":tweet1,"Match":word,"Polarity":'negative'},ignore_index=True)
            print(word)
        elif word in neutralwords:
            neutratweets.append(tweet1)
            final_df=final_df.append({"Tweet":i+1,"tweets":tweet1,"Match":word,"Polarity":'neutral'},ignore_index=True)
            print(word)

print(bagofwords)

positive_tweets=unique_tweets(postweet)

negative_tweets=unique_tweets(negtweet)
neutral_tweets=unique_tweets(neutratweets)
negative_word_tweets=final_df[final_df['Polarity']=='negative'].Match
list_of_negativewords=negative_word_tweets.to_list()
print(list_of_negativewords)
pos_word_tweets=final_df[final_df['Polarity']=='positive'].Match
list_of_positivewords=pos_word_tweets.to_list()
print(list_of_positivewords)
neutral_word_tweets=final_df[final_df['Polarity']=='neutral'].Match
list_of_neutralwords=neutral_word_tweets.to_list()
print(list_of_neutralwords)
```

STEP 4: Tagging the tweets as positive, negative, neutral: In this step, the words that are coming in the tweets which will be responsible for the polarity of the overall tweet are calculated, and a table is made to display the findings. The table is made using Pandas Dataframe and it has 4 columns, Tweet column shows the count of the tweet, tweets column shows the actual tweet, Match column shows the matching word that will decide the polarity and lastly the polarity column that displays the overall polarity of the tweet. Snapshot of the table is given below.

Out[247]:

	Tweet	tweets	Match	Polarity
0	1	[as the new arthur b mcdonald chair of researc...	excellence	positive
1	4	[although this list is from earlier this year ...	good	positive
2	5	[dalplex is encouraging the dalhousieu commun...	accept	neutral
3	7	[dsu president aisha went down to the dsufoodb...	support	positive
4	8	[mata amritanandamayi ji is known for her self...	helping	positive
5	9	[be it education or community service the dav ...	positive	positive
6	10	[very good also appreciable to see that the st...	good	positive
7	11	[remarkable gesture wishing you a happy and he...	remarkable	positive
8	11	[remarkable gesture wishing you a happy and he...	wishing	neutral
9	11	[remarkable gesture wishing you a happy and he...	happy	positive
10	11	[remarkable gesture wishing you a happy and he...	healthy	positive
11	12	[good aaz27zjpxm]	good	positive
12	13	[had a good discussion on phone today with pri...	good	positive
13	13	[had a good discussion on phone today with pri...	support	positive
14	17	[it may feel like there s a lot of information...	like	positive
15	19	[wow so grateful tyg6tfbm35]	wow	positive
16	19	[wow so grateful tyg6tfbm35]	grateful	positive
17	22	[t 3500 happy baisakhi brruuuuruaahhhhh i00p...	happy	positive
18	24	[hahaha yesss lahsmfj3sz]	hahaha	positive
19	25	[i bow to those martyrs who were killed mercil...	killed	negative
20	28	[dalplex intramural staff are interested in r...	interested	positive

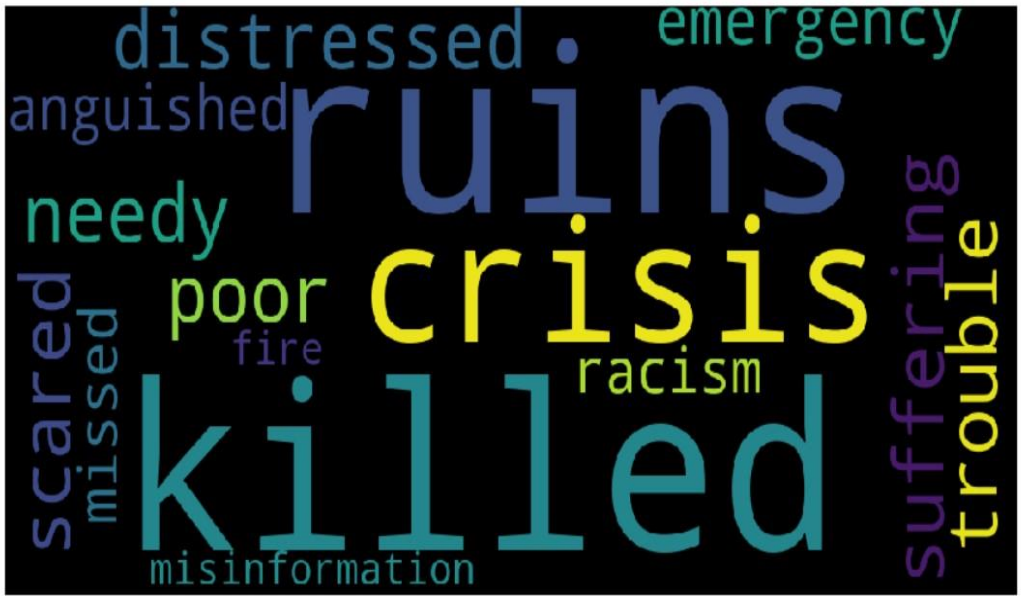
STEP 5: Visualizing the tweet sentiments using the word cloud: Word cloud basically represents the words in the form of different sizes based on their number of occurrences. The word that is coming for a greater number of times will be displayed as larger and the size of the words decreases as their occurrence declines. The code for creating the word cloud is displayed below.

```
word_could_dict=Counter(list_of_positivewords)
print(word_could_dict)
wordcloud = WordCloud(width = 1000, height = 500).generate_from_frequencies(word_could_dict)
plt.figure(figsize=(15,20))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

word_could_dict=Counter(list_of_negativewords)
print(word_could_dict)
wordcloud = WordCloud(width = 1000, height = 500).generate_from_frequencies(word_could_dict)
plt.figure(figsize=(15,20))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

word_could_dict=Counter(list_of_neutralwords)
print(word_could_dict)
wordcloud = WordCloud(width = 1000, height = 500).generate_from_frequencies(word_could_dict)
plt.figure(figsize=(15,20))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Counter({'good': 11, 'support': 8, 'like': 7, 'thank': 7, 'help': 6, 'important': 6, 'happy': 5, 'healthy': 3, 'thanks': 3, 'hope': 3, 'great': 3, 'helping': 2, 'positive': 2, 'best': 2, 'noble': 2, 'wonderful': 2, 'creative': 2, 'stronger': 2, 'devoted': 2, 'courage': 2, 'love': 2, 'benefit': 2, 'excellence': 1, 'remarkable': 1, 'wow': 1, 'grateful': 1, 'hahaha': 1, 'interested': 1, 'fun': 1, 'generous': 1, 'peace': 1, 'beautiful': 1, 'won': 1, 'inspiring': 1, 'helps': 1, 'greatest': 1, 'excited': 1, 'enjoy': 1, 'passionate': 1, 'comfort': 1, 'awesome': 1, 'proactive': 1, 'proud': 1, 'welcome': 1, 'success': 1, 'excellent': 1, 'kudos': 1, 'effective': 1, 'brave': 1, 'awards': 1, 'compassionate': 1, 'highlight': 1, 'sincere': 1, 'strength': 1, 'effectively': 1, 'gracious': 1, 'affection': 1, 'better': 1, 'commend': 1, 'outstanding': 1, 'appreciate': 1, 'cares': 1, 'win': 1, 'dearly': 1, 'perfect': 1, 'greater': 1, 'joy': 1, 'opportunity': 1, 'focused': 1, 'greetings': 1, 'visionary': 1, 'smarter': 1, 'worth': 1})

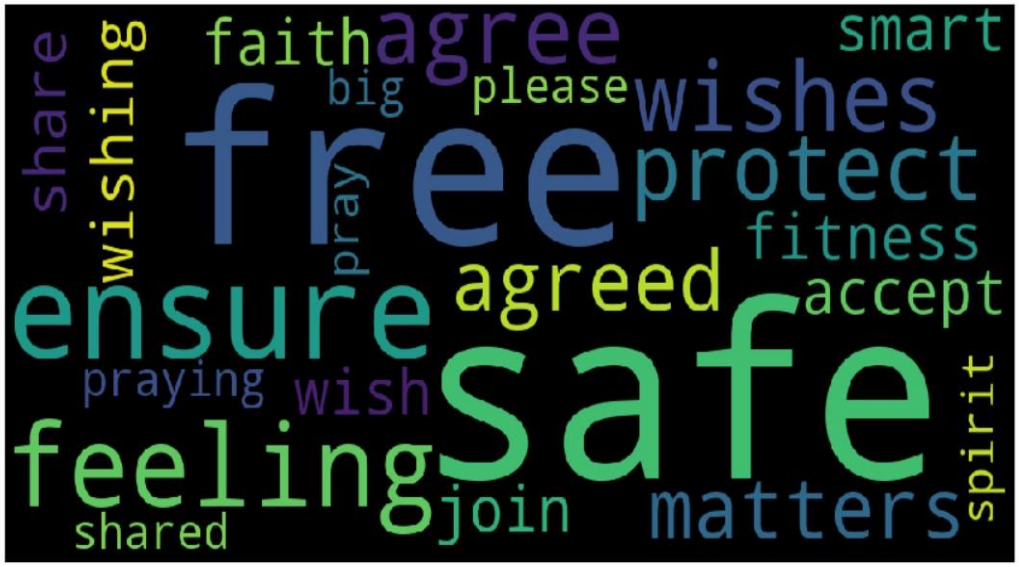
POSITIVE WORD CLOUD

Counter({'killed': 1, 'ruins': 1, 'crisis': 1, 'distressed': 1, 'needy': 1, 'suffering': 1, 'scared': 1, 'poor': 1, 'trouble': 1, 'emergency': 1, 'anguished': 1, 'racism': 1, 'missed': 1, 'misinformation': 1, 'fire': 1})



NEGATIVE WORD CLOUD

```
Counter({'safe': 4, 'free': 3, 'ensure': 3, 'feeling': 3, 'wishes': 2, 'protect': 2, 'agree': 2, 'agreed': 2, 'matters': 2, 'ac
cept': 1, 'wishing': 1, 'fitness': 1, 'faith': 1, 'join': 1, 'smart': 1, 'share': 1, 'wish': 1, 'praying': 1, 'please': 1, 'spi
rit': 1, 'shared': 1, 'big': 1, 'pray': 1})
```



NEUTRAL WORD CLOUD

PART B: SEMANTIC ANALYSIS FROM NEWS API: In this part, data from the news API is collected, filtered and then semantic analysis is performed on the content of the articles. It involves step by step process which is explained below:

STEP 1: Getting data from news API, cleaning the data and then filtering the data: The data is extracted from the news API by writing the code as shown below:

```
headers = {'Authorization': '1a144a80798743fb9771c38a2287619d'}


top_headlines_url = 'https://newsapi.org/v2/top-headlines'

everything_news_url = 'https://newsapi.org/v2/everything'

sources_url = 'https://newsapi.org/v2/sources'

headlines_payload = {'category': 'education', 'country': 'Canada'}
everything_payload = {'q': 'Dalhousie University', 'language': 'en', 'sortBy': 'popularity','pageSize': 100}
sources_payload = {'q': 'Halifax','category': 'general', 'language': 'en', 'country': 'Canada'}


response = requests.get(url=everything_news_url, headers=headers, params=everything_payload)


pretty_json_output = json.dumps(response.json(), indent=4)
print(pretty_json_output)



response_json_string = json.dumps(response.json())


response_dict = json.loads(response_json_string)
print(response_dict)


articles_list = response_dict['articles']
```

By this, the list of articles is retrieved in json format, now it is stored in Pandas data frame and converted to a csv file named as newsarticles.csv. As specified in the requirement of the assignment, only the title, description and content are needed from the news API, so it is filtered in the pandas data frame as shown below by the code.

```
df = pd.read_json(json.dumps(articles_list))
df.to_csv('G:/DALHOUSIE Term 1/Data management and ware dalhousie/newsarticles.csv')
df3=pd.read_csv('G:/DALHOUSIE Term 1/Data management and ware dalhousie/newsarticles.csv')
df3=df3[["title","description","content"]]
listofcontents=df3['content'].to_list()
```

STEP 2: Computing the term frequency-inverse document frequency: In this section the objective is to create a table that has 4 columns. The first column has the search string, that is these are the strings that are basis for searching in the news articles. The second column contains the number of documents (news articles) that have a string in them. Third column is the ratio of the total number of articles that are used divided by the number of articles that have the search keyword. The last column is the logarithmic value of third column. The code created for doing this is shown below:

```
subs="Canada"
subs2="business"
subs3="Dalhousie University"
subs4="University"
subs5="Halifax"
res =[i for i in cleanedList if subs in i]
res2=[j for j in cleanedList if subs2 in j]
res3=[k for k in cleanedList if subs3 in k]
res4=[l for l in cleanedList if subs4 in l]
res5=[m for m in cleanedList if subs5 in m]
final_df2=pd.DataFrame(columns=["Searchquery","noofdocsmatchquery","ratio","Log10(N/df)"])
final_df2=final_df2.append({"Searchquery":'Canada',"noofdocsmatchquery":len(res),"ratio":100/(len(res)),
                            "Log10(N/df)":math.log10(100/len(res))},ignore_index=True)
final_df2=final_df2.append({"Searchquery":'business',"noofdocsmatchquery":len(res2),"ratio":100/(len(res2)),
                            "Log10(N/df)":math.log10(100/len(res2))},ignore_index=True)
final_df2=final_df2.append({"Searchquery":'Dalhousie University',"noofdocsmatchquery":len(res3),"ratio":100/(len(res3)),
                            "Log10(N/df)":math.log10(100/len(res3))},ignore_index=True)
final_df2=final_df2.append({"Searchquery":'University',"noofdocsmatchquery":len(res4),"ratio":100/(len(res4)),
                            "Log10(N/df)":math.log10(100/len(res4))},ignore_index=True)
final_df2=final_df2.append({"Searchquery":'Halifax',"noofdocsmatchquery":len(res5),"ratio":100/(len(res5)),
                            "Log10(N/df)":math.log10(100/len(res5))},ignore_index=True)
print(final_df2.head())
```

```
           Searchquery noofdocsmatchquery      ratio  Log10(N/df)
0               Canada                 16   6.250000     0.795880
1             business                  2  50.000000     1.698970
2  Dalhousie University                 9  11.111111     1.045757
3           University                 14   7.142857     0.853872
4               Halifax                  6  16.666667     1.221849
```

The news API has set limits on extraction of articles, so this research is performed for 100 articles, so the value of N is 100.

STEP 3: Calculating the frequency of the word “Canada” in articles: In this step, a table is created that has 5 columns. The first column is the article number out of 100, the second column is the total

number of words in that articles content part, the third column is the frequency of the word “Canada” in that particular article, the fourth column is the article content in short, the last column is the relative frequency of the word “Canada”. The code for creating the table is shown below along with the table.

```
final_df6=pd.DataFrame(columns=["articlenumber","totalwords","frequency","articlecontent"])
for z in range(len(cleanedList)):
    substr1="Canada"
    tokens=nltk.word_tokenize(cleanedList[z])
    print(tokens)
    res=[i for i in tokens if substr1 in i]
    print(res)
    print(len(tokens));print(len(res))
    final_df6=final_df6.append({"articlenumber":'Article#'+str(z),"totalwords":len(tokens),
                                "frequency":len(res),"articlecontent":cleanedList[z]},ignore_index=True)
final_df6['relativefrequency']=final_df6['frequency']/final_df6['totalwords']
final_df7=final_df6[final_df6['frequency']>0]
print(final_df7.loc[final_df7['relativefrequency'].astype(float).idxmax(),'articlecontent'])
```

```
In [254]: final_df7
```

Out[254]:

	articlenumber	totalwords	frequency	articlecontent	relativefrequency
10	Article#10	52	1	WINNIPEG, Manitoba/OTTAWA (Reuters) - After fa...	0.0192308
12	Article#12	46	1	While the coronavirus pandemic has prompted a ...	0.0217391
15	Article#15	49	1	TORONTO Technology companies operating in Cana...	0.0204082
19	Article#19	45	1	A national modelling paper predicting the numb...	0.0222222
43	Article#43	52	1	Politics Insider for March 31: We're tracking ...	0.0192308
51	Article#51	51	1	March 11, 2020For Immediate Release\r\nMembers...	0.0196078
53	Article#53	53	1	After failing to grow wheat in Canada's subarc...	0.0188679
58	Article#58	49	2	Canada needs to become more secure by becoming...	0.0408163
60	Article#60	50	2	Younger Canadians represent one in three of al...	0.04
64	Article#64	51	1	A mushroom spray, oil of oregano and a special...	0.0196078
69	Article#69	44	1	Canadian prison officials say they are adoptin...	0.0227273
70	Article#70	52	1	The latest novel coronavirus news from Canada ...	0.0192308
85	Article#85	53	1	When skin care entrepreneur Eliza Desmarais go...	0.0188679
87	Article#87	49	1	A Dalhousie University researcher at the foref...	0.0204082
91	Article#91	46	1	Grocery stores across Canada are preparing for...	0.0217391
97	Article#97	46	2	The following was written by a group of health...	0.0434783

It can be observed that article number 97 has the maximum relative frequency.

STEP 4: Printing the article that has the highest relative frequency for the word “Canada”: It can be clearly seen that article number 97 has the highest relative frequency of the word Canada, so it should be printed in the output. It is displayed below.

```
print(final_df7.loc[final_df7['relativefrequency'].astype(float).idxmax(),'articlecontent'])
```

CONCLUSION: The sentiment analysis of the tweets and the semantic analysis of the news articles was the primary purpose of the assignment. These two techniques are very vital in text mining and Natural Language processing. All the source code, csv files and other required documents are stored in the main folder.