Section-A: (Sentiment Analysis)

For performing sentiment analysis [1], I have used the twitter API key to extract the data and cleaned it by using the code in figure 1. In the assignment folder, I have provided the python file of the code used for cleaning and CSV file of the twitter data in the folder "Kuramsetti, Pavansai, B00846500_A4\Sentiment Analysis" under the names "test.py" and "datatwitter" respectively. Then, I have transformed the extracted data to another CSV file "datatwitterchanged" by removing retweets. I have removed retweets as they are repeated and provides the same output as the original tweet. By using the latest CSV file which contains 1857 tweets, I have performed sentimental analysis on the data, where the code is provided under the name "sent.py". I have collected negative words from [2] and positive words from [3] to perform analysis. While performing the analysis, I have provided the polarity as neutral if none of the words in the tweet matched with the positive or negative words. The output file is stored in the folder "Kuramsetti, Pavansai, B00846500_A4\Sentiment Analysis" under the CSV file "Resultschange". I have also provided text files of the positive and negative words under the same folder using names "Positivewords" and "Negativewords" respectively.

Code logic used for cleaning twitter data: (Figure 1)

```
p1 = open("datatwitterr.csv", 'a', newline='', encoding="utf-8-sig")
p2 = csv.writer(p1)
data = ["created_at", "text", "User_Location", "lang", "retweeted", "reweet_count"]
p2.writerow(data)
for tweet in tweepy.Cursor(api.search,
                           q='Canada' or 'University' or 'Dalhousie University' or 'Halifax' or 'Canada Education').items(3010)
    data = []
    data.append(tweet._json["created_at"])
    # cleaning the data by attribute
    regex = re.sub(r'[^\x00-\x7F]+|http\S+', '', str(tweet._json["text"]))
    regex = [re.sub(r"[^a-zA-Z0-9]+", '', j) for j in regex.split(" ")]
    regex = " ".join(regex)
    data.append(regex)
    # appending each attribute after cleaning
    regex = re.sub(r'[^\x00-\x7F]+|http\S+', '', str(tweet._json["user"]["location"]))
    regex = [re.sub(r"[^a-zA-Z0-9]+", ' ', j) for j in regex.split(" ")]
    regex = " ".join(regex)
    data.append(regex)
    data.append(tweet._json["lang"])
    data.append(tweet._json["retweeted"])
    data.append(tweet._json["retweet_count"])
    p2.writerow(data)
```

Obtained output format: (Figure 2)

Tweet	Message/tweets	Match	Polarity
1	EmilyLazatin PM JustinTrudeau says gov is teaming up with Arcteryx canadagoose and Stanfields1856 to produce medical gownsCanad		Neutral
2	TPostMillennial Stay strong Boris Your friends in Canada are supporting you through this	strong,sup	Positive
3	haroldmunro What a great story Coquitlam company retools and starting Tuesday 100000 medicalgrade surgical masks will be rolling o	great	Positive
4	Bourassa1963 Critical COVID19 masks heading to Canada after Trump and 3M reach deal		Neutral
5	JPTasker Canada Goose and Stanfields among other clothing manufacturers will make some PPE like gowns Trudeau says theyve signed	like	Positive

Code for Sentiment Analysis: (Figure 3)

Conversion of positive and negative word files to list:

```
import csv
# Stored Tweets
inputfile = open(r"C:\Users\pavan\OneDrive\Documents\datatwitterchanged.csv", 'r')
# Output file of sentimental analysis
outputfile = open(r"C:\Users\pavan\OneDrive\Documents\Resultschange.csv", 'w', newline='')
csv_reader = csv.DictReader(inputfile)
csv_writer = csv.writer(outputfile)
# Adding Header to the output file
data = ["Tweet", "Message/tweets", "Match", "Polarity"]
csv_writer.writerow(data)
# Positive words text file
positivewords = open(r"C:\Users\pavan\OneDrive\Documents\Positivewords.txt", 'r')
# Negative words text file
negativewords = open(r"C:\Users\pavan\OneDrive\Documents\Negativewords.txt", 'r')
pos = []
neg = []
# for tweet count
i = 0
# Converting positive words to string
for line in positivewords:
    pos.append(line.rstrip('\n'))

# Converting negative words to string
for line in negativewords:
    neg.append(line.rstrip('\n'))
```

Code for creating a bag of words: (Figure 4)

```
# Passing Each tweet to function
def word_count(tweet):
    Eachword = {}
    # print(Eachword)
    tweets = tweet.split()
    for word in tweets:
        if word in Eachword:
            Eachword[word] = Eachword[word] + 1
            # print(Eachword)
        else:
            Eachword[word] = 1
            # print(Eachword)
    return Eachword

# Taking tweets from csv file
for line in csv_reader:
    line["text"] = line["text"].replace("RT", "")
    i = i + 1
    data = []
    # For the tweet number
    data.append(i)
    data.append(line["text"])
    # passing to function
    word = word_count(line["text"])
    print(word)
    print(line["text"])
```

Positive and negative word count: (Figure 5)

```
word = word_count(line["text"])
print(word)
words = []
x1 = 0
y1 = 0
for x in range(len(pos)):
    if pos[x] in word:
        words.append(pos[x])
        # positive words count
        x1 = x1 + word[pos[x]]

for y in range(len(neg)):
    if neg[y] in word:
        words.append(neg[y])
        # Negative words count
        y1 = y1 + word[neg[y]]

print(x1)
print(y1)
# converting list of match words to string
matchwords = ','.join(words)
print(matchwords)
data.append(matchwords)
if x1 > y1:
    print("positive word")
    data.append("Positive")
```

(Figure 6)

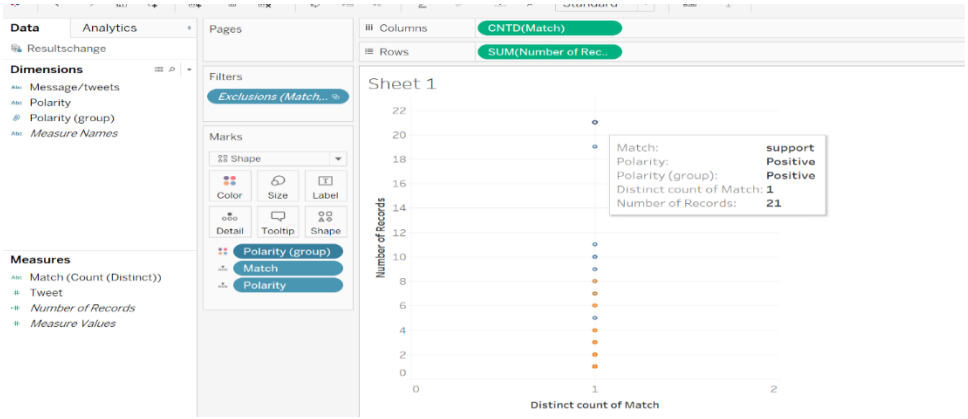
```
elif x1 < y1:
    print("Negative word")
    data.append("Negative")
else:
    print("Neutral")
    data.append("Neutral")
print(data)
# appending each row to csv file
csv_writer.writerow(data)
```

6. Visualization of words:

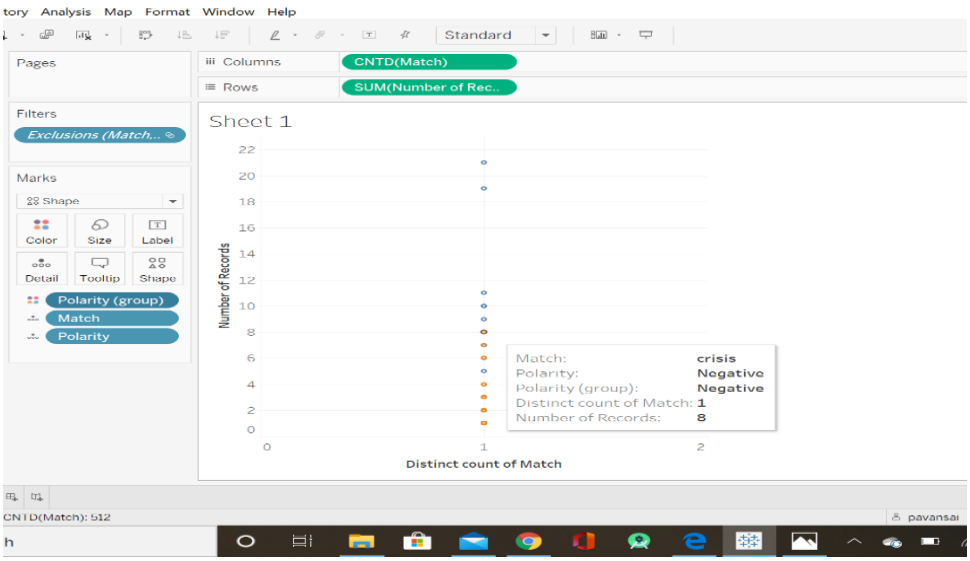
For visualizing the data, I have used the “Resultschange” CSV file obtained in sentiment analysis. I have installed tableau by following the steps provided in the lab tutorial 7 and learned basics from the documentation [3]. At first I have uploaded the CSV file into tableau using desktop server. Then I have visualized the most frequent words that occurred using the match column. I have attached all the screenshots in the folder “Kuramsetti, Pavansai, B00846500_A4\Visualization” under the name used for them as below.

Graph which shows the most frequent words in positive and negative tweets in a single line: (Figure 7)

In the below figure, we can see that the most frequent words from high to low. The most frequent word is “support” which is found in 21 tweets and it is the positive tweet. All the blue dots represent positive tweets and orange dots represent negative tweets.

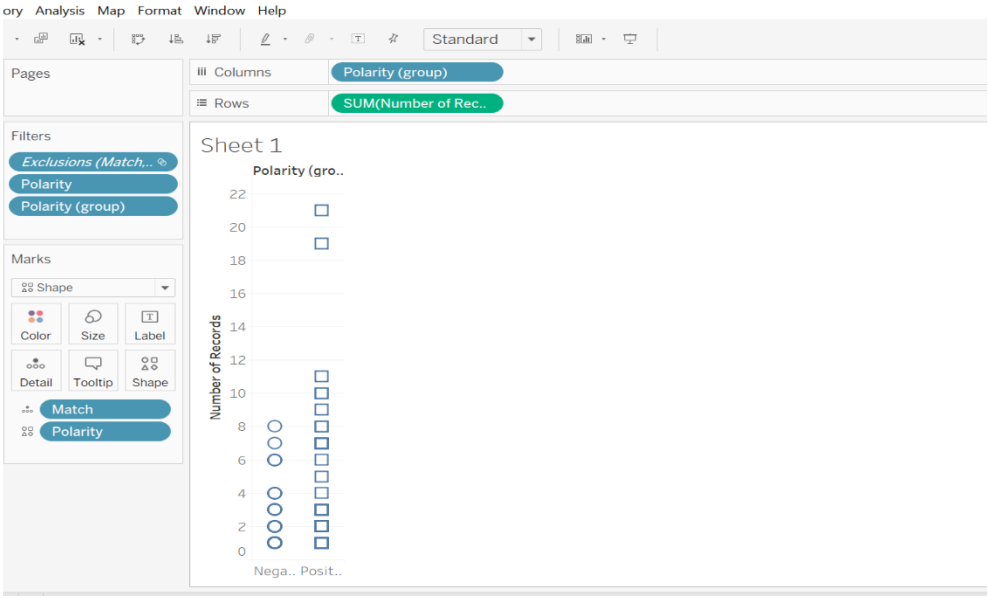


In the below picture, I have highlighted the negative word “crisis” which is the most frequent word in negative tweets. **(Figure 8)**

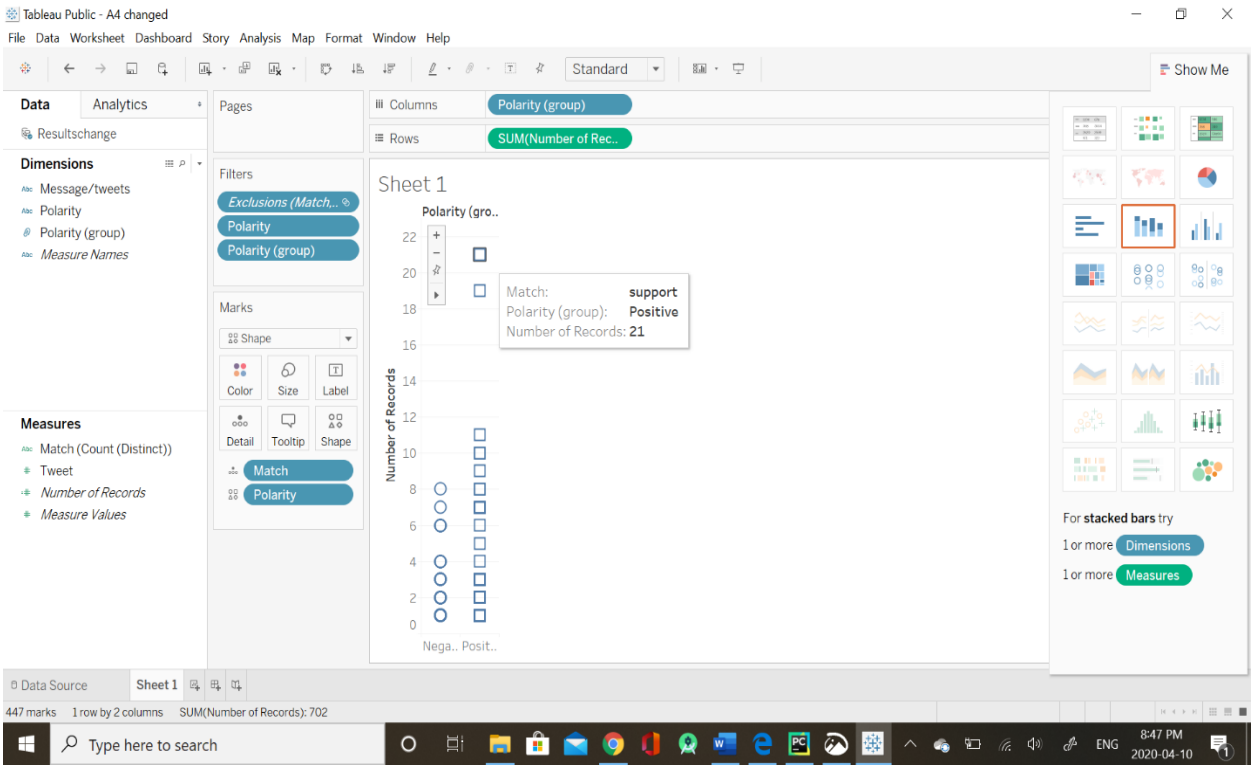


Graph which represents positive and negative words separately as two lines: (Figure 9)

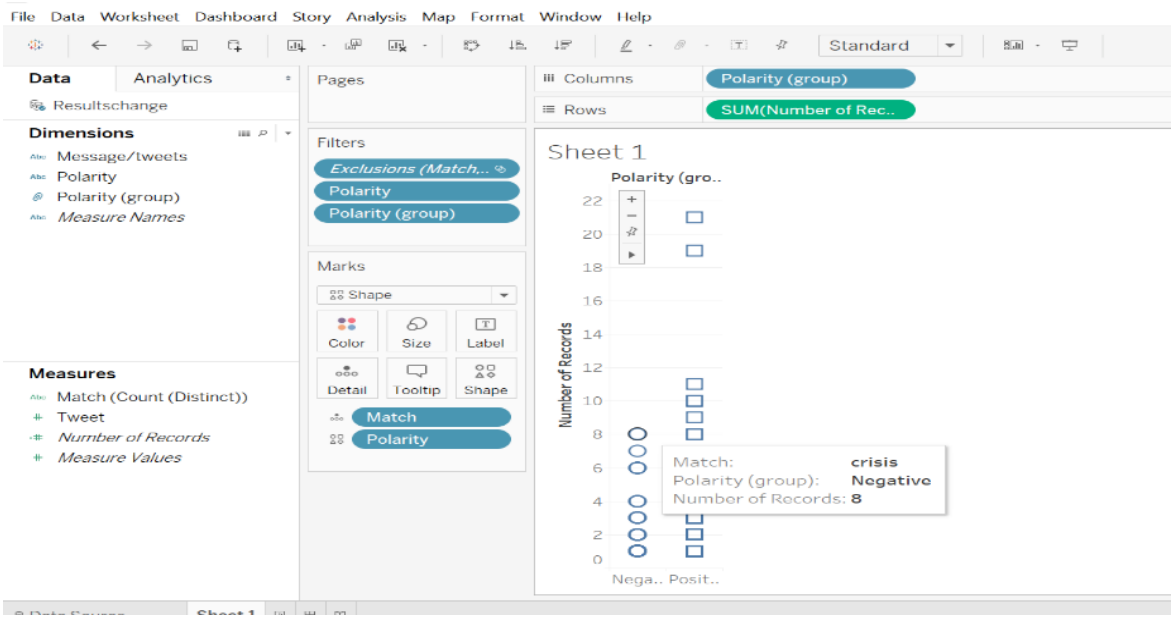
The below figure represents the graph for the positive and negative words. In the diagrams followed by the below picture, I have also highlighted the most frequent positive and negative words.



Positive word: **(Figure 10)**

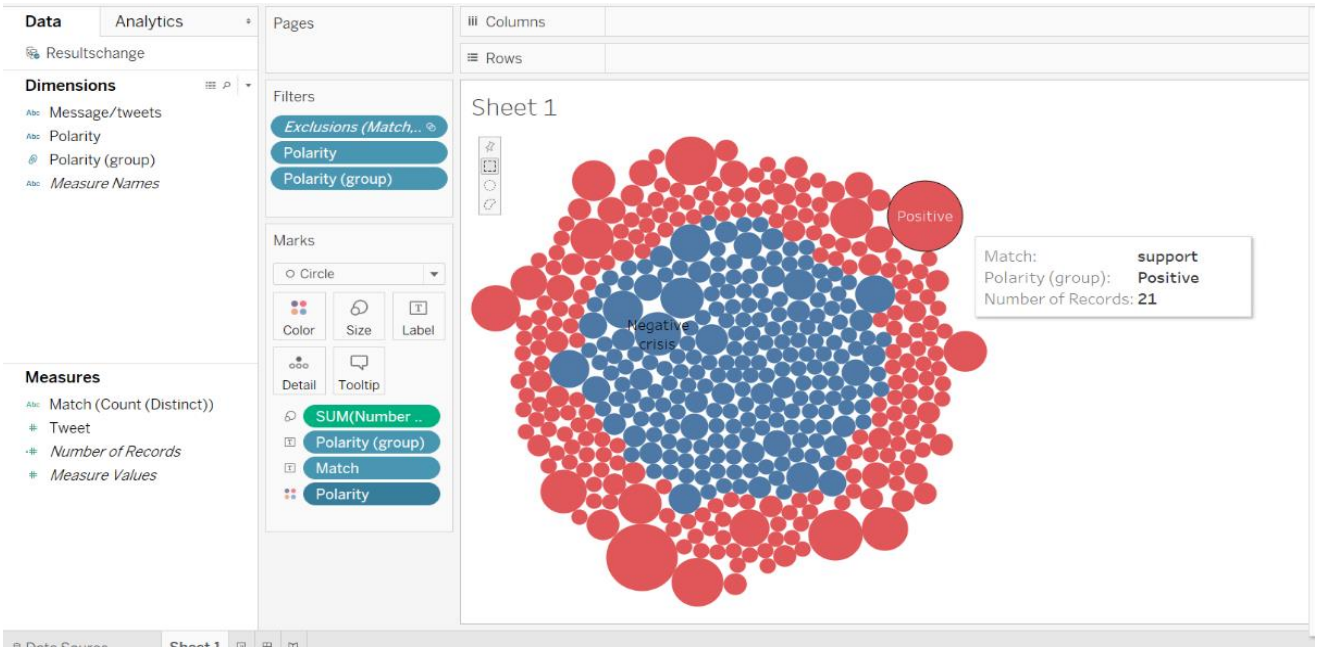


Negative word: **(Figure 11)**



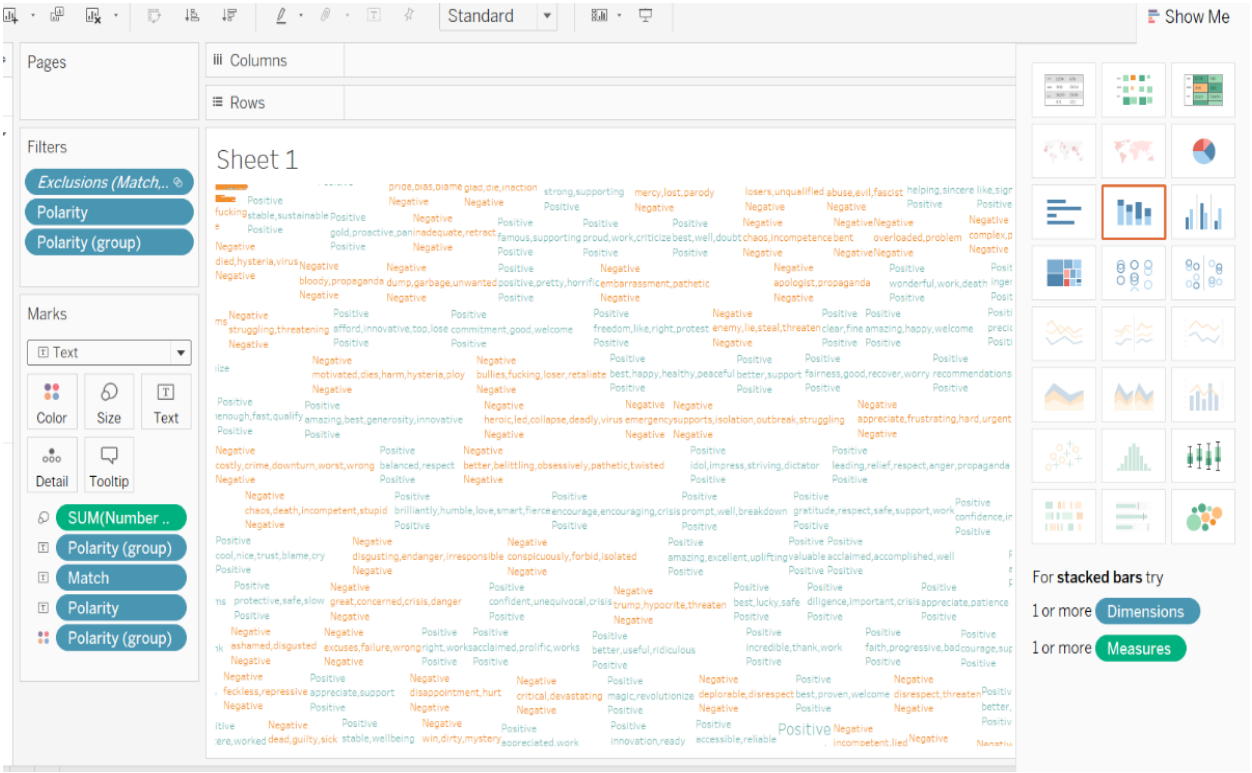
Packed bubbles: (Figure 12)

To visualize the bunch of positive and negative words, I have used packed bubbles. The red bubbles represent the positive words and the blue bubbles represent the negative words. The size of the bubble directly proportional to the frequency of the word if the frequency is more then the size is high. I have also highlighted the biggest bubble of positive words and also the biggest of negative words as positive and negative respectively.



Word cloud which represents all the words: (Figure 13)

The words which are in orange color represent negative words and the blue words represent positive words. I have also mentioned polarity along with the word. As I have more unique values so the word cloud is not clear, this is the reason that made me choose packed bubbles to represent the data.



Section-B: (Semantic Analysis)

For performing semantic analysis [5], I have extracted and cleaned news data using the code provided in figure 14. I have also provided the python file in the folder “Kuramsetti, Pavansai, B00846500_A4\Semantic Analysis” under the name “news.py”. I have loaded all the data to CSV file which is provided in the same folder under the name “datanewsss”. As per the instructions, I have extracted each article with attributes(title, description, content) to a new document that allowed me to form 140 documents. I have provided all the documents in the folder “Kuramsetti, Pavansai, B00846500_A4\Semantic Analysis\All Documents.” I have also provided the code to divide the data into 140 documents under the name “Semantic.py” in the assignment folder “Kuramsetti, Pavansai, B00846500_A4\Semantic Analysis”.

News data cleaning code: (Figure 14)

```
def news(key):  # passing each key as a parameter to the function
    all_articles = newsapi.get_everything(q=key,
                                          language='en')
    print(all_articles)
    for x in all_articles["articles"]:
        data = []
        regex = re.sub(r'[^\x00-\x7F]+|http\S+', '', str(x["title"]))
        regex = [re.sub(r"[^a-zA-Z0-9]+", ' ', j) for j in regex.split(" ")]
        regex = " ".join(regex)
        data.append(regex)
        # cleaning each required attribute
        regex = re.sub(r'[^\x00-\x7F]+|http\S+', '', str(x["description"]))
        regex = [re.sub(r"[^a-zA-Z0-9]+", ' ', j) for j in regex.split(" ")]
        regex = " ".join(regex)
        data.append(regex)
        # appending attribute after cleaning
        regex = re.sub(r'[^\x00-\x7F]+|http\S+', '', str(x["content"]))
        regex = [re.sub(r"[^a-zA-Z0-9]+", ' ', j) for j in regex.split(" ")]
        regex = " ".join(regex)
        data.append(regex)
        p2.writerow(data)
```

Code to divide data into documents: (Figure 15)

```
import csv

inputfile = open(r"C:\Users\pavan\OneDrive\Documents\datanewsss.csv", 'r', encoding="utf-8-sig")
csv_reader = csv.DictReader(inputfile)
linecount = 0
for line in csv_reader:
    #print(line)
    linecount = linecount + 1
    outputfile = open(r'Semenatic' + str(linecount) + '.txt', 'w')
    # csv_writer = csv.writer(outputfile)
    # data = ["Title", "Description", "Content"]
    # csv_writer.writerow(data)
    data = []
    # print(line.keys())
    data.append(line["Title"])
    data.append(line["Description"])
    data.append(line["Content"])
    data1 = ' '.join(data)
    outputfile.write(data1)
```

Term Frequency-Inverse Document Frequency (TF-IDF):

a. By following the instruction, I have applied TF-IDF on 140 documents (N=140). Below I'm attaching the screenshot of obtained output and the code for performing log(N/df). I have also provided the python and CSV files in the folder "Kuramsetti, Pavansai, B00846500_A4\Semantic Analysis" under the names "testseman.py" and "Semanticresults.csv" respectively.

Obtained output: (Figure 16)

A	B	C	D
Total Documents	140		
Search Query	Document	Total Docu	Log(N/df)
Canada	50	2.8	0.45
University	23	6.09	0.78
Dalhousie University	6	23.33	1.37
Halifax	12	11.67	1.07
Business	3	46.67	1.67

Code: (Figure 17)

I have written all the code for TF-IDF (a and b) in a single python file. I'm attaching screenshots by snipping the code separately to make it understandable.

```
import csv
import math

if __name__ == '__main__':
    import Semantic
    linecoun = Semantic.linecount
    print(linecoun)
outputfile = open(r"C:\Users\pavan\OneDrive\Documents\Semanticresults.csv", 'w', newline='')
csv_writer = csv.writer(outputfile)
adddata=["Total Documents",linecoun]
csv_writer.writerow(adddata)
adddata=["Search Query","Document containing term(df)","Total Documents(N)/number of documents term appeared(df)","Log(N/df)"]
csv_writer.writerow(adddata)

data = ["Canada", "University", "Dalhousie University", "Halifax", "Business"]
type(data)
listofart=[]
for list in data:
    adddata=[]
    adddata.append(list)
    print(list)
    doccount = 0
    z=0
    logvalue = 0
```

(Figure 18)

```
    for i in range(1, linecoun):
        with open("Semenatic" + str(i) + ".txt") as openfile:
            for line in openfile:
                if list in line:
                    doccount = doccount + 1
                    print(i)
                    if list == "Canada":
                        k = doccount
                        listofart.append(i)
    #print(doccount)
    #print(listofart)
    adddata.append(doccount)
    print("only canada count" + str(k))
    if doccount != 0:
        z = round((linecoun / doccount),2)
        #print(z)
        adddata.append(z)
        logvalue=round(math.log(z, 10),2)
        adddata.append(logvalue)
        #print(logvalue)
    csv_writer.writerow(adddata)
    print(adddata)
```

b. By continuing with the above data, I have generated all the articles which contain the term "Canada". The python and CSV files are provided in the folder "Kuramsetti, Pavansai, B00846500_A4\Semantic Analysis" under the names "testseman.py" and "Semanticres.csv" respectively.

Obtained output: (Figure 19)

	A	B	C
1	Term	Canada	
2	Canada appeared in 50 Documents	Total word(m)	Frequency(f)
3	Article #1	103	5
4	Article #2	99	2
5	Article #3	82	3
6	Article #4	66	3
7	Article #5	113	3
8	Article #6	104	3
9	Article #7	109	1
10	Article #8	67	2
11	Article #9	68	2
12	Article #10	97	3

Code: (Figure 20)

```
outputfile1 = open(r"C:\Users\pavan\OneDrive\Documents\Semanticres.csv", 'w', newline='')
csvwriter = csv.writer(outputfile1)
addrow=["Term","Canada"]
csvwriter.writerow(addrow)
addrow=["Canada appeared in "+str(k)+" Documents","Total word(m)","Frequency(f)"]
csvwriter.writerow(addrow)
hrf = 0
for i in listofart:
    addrow=[]
    with open("Semenatic" + str(i) + ".txt") as openfile:
        article=0
        canadacount = 0
        wordcount = 0
        for line in openfile:
            for part in line.split():
                wordcount = wordcount + 1
                if "Canada" in part:
                    article="Article #" + str(i)
                    canadacount = canadacount + 1
    addrow.append(article)
    addrow.append(wordcount)
    addrow.append(canadacount)
    csvwriter.writerow(addrow)
    print(addrow)
```

(Figure 21)

```
    if i == 1:
        print()
        hrf = (canadacount / wordcount)
        p=i
    else:
        if hrf < (canadacount / wordcount):
            hrf = (canadacount / wordcount)
            print("look here"+str(hrf))
            p=i
print("Look here" + str(hrf))
highest="Article with highest relative frequency is Article #" + str(p)
addhighest=[]
addhighest.append(highest)
csvwriter.writerow(addhighest)
```

c. f/m value:

I have printed the highest f/m value in the CSV file “semanticres.csv”. The code for f/m value can be found in “testseman.py” file which is also provided in the above figure 21.

49	Article #119	64	2
50	Article #120	95	2
51	Article #131	99	1
52	Article #134	99	2
53	Article with highest relative frequency is Article #18		

REFERENCES:

[1]"Sentiment analysis", En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Sentiment_analysis. [Accessed: 07- Apr- 2020].

[2]"jeffreybreen/twitter-sentiment-analysis-tutorial-201107", GitHub. [Online]. Available: https://github.com/jeffreybreen/twitter-sentiment-analysis-tutorial-201107/blob/master/data/opinion-lexicon-English/negative-words.txt. [Accessed: 08- Apr- 2020].

[3]"positive-words.txt", Gist. [Online]. Available: https://gist.github.com/mkulakowski2/4289437. [Accessed: 08- Apr- 2020].

[4]"Get Started - Tableau", Help.tableau.com. [Online]. Available: https://help.tableau.com/current/pro/desktop/en-us/gettingstarted_overview.htm. [Accessed: 09- Apr- 2020].

[5]"Semantic analysis (linguistics)", En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Semantic_analysis_(linguistics). [Accessed: 09- Apr- 2020].