Section A: Sentimental Analysis

1&2. I have used tweepy to extract the tweets, and in that, I have taken only the text field. At the time of extraction itself, I had filtered all the data. And stored the data into a text file named as “tweettext.txt”. I have used the below python script named as “tweet.py” for the process. I have enclosed below script and the output file in the **yaswanthchiruvella_B00849892/sentimental** folder. Below is the screenshot for reference.

```
# Tweet.py used for data extraction from twitter.
import tweepy
import re

consumer_key = "gfUORUQGGFsuKvGgx6Z9x9Duk"
consumer_secret = "xYoAmf91wtTj0FYybSnrfF14nLb4szt0hZ19TrBRVijuHWEhKT"
access_token = "1829926238-MsFpsArEeAexMmfieTOmbdDgcWkmVFxJbk6Bc0b"
access_token_secret = "6II006wphfSYxWoU1Gsnh1cCuy9eEgUfkqMSgHtGGUAHB"
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth, wait_on_rate_limit=True)
file = open("tweettext.txt","w")

search_words = 'Canada OR University OR Dalhousie University OR Halifax OR Canada Education'


for tweet in tweepy.Cursor(api.search,
                           q=search_words,
                           lang="en").items(844):
  remove_specialchars = re.compile(r'http\S+|([^a-zA-Z\s]+?)|//?|<.*?>|\\|RT?')
  without_emojios = tweet._json["text"].encode('ascii', 'ignore').decode('ascii')
  json_str = remove_specialchars.sub(r'', without_emojios)
  file.write(json_str+"\n")
```

4 &5. I created a bag of words for each tweet and compared the same with a list of positive and negative words which I extracted from this website [1]. I have stored the matched words in a separate column in a csv file named “sentiment.csv”. I kept duplicate words too, means if a text message tweet has two "like" then I kept that two likes in the "Match" column its because to show why I measured that tweet as a positive even though a negative word is present? Since it has two positive matches. I kept that because to show Below is the python script named as “Sentiment Analysis.py” that I have used to accomplish this process. I have enclosed below python script and csv file in the **yaswanthchiruvella_B00849892/sentimental** folder. Below is the screenshot for your reference.

Note: In case if the message column in “sentiment.csv” is not visible in excel please format the column as a “text”.

```python
# Python script used to create bag of words from tweet and compare the same with postive/negative wordlis
import csv

positive_count = 0
negative_count = 0
positive_list = [...]
negative_list = [...]

def polarity(pos, neg):
    if pos > neg:
        return "Positive"
    elif pos == neg:
        return "Neutral"
    else:
        return "Negative"
i = 0
list = []
match = []
headers = ['Tweet','Message/tweets','Match','Polarity']

with open('sentiment.csv', 'w', newline='') as new_file:
    csv_writer = csv.writer(new_file)
    csv_writer.writerow(headers)
    with open('tweettext.txt', 'r') as f:
        for lines in f:
            list.append(lines.replace('\n', ""))
            # print(list)
            length = len(list)
            while i < length:
                id = i
                words = list[i].split()
                # print(words)
```

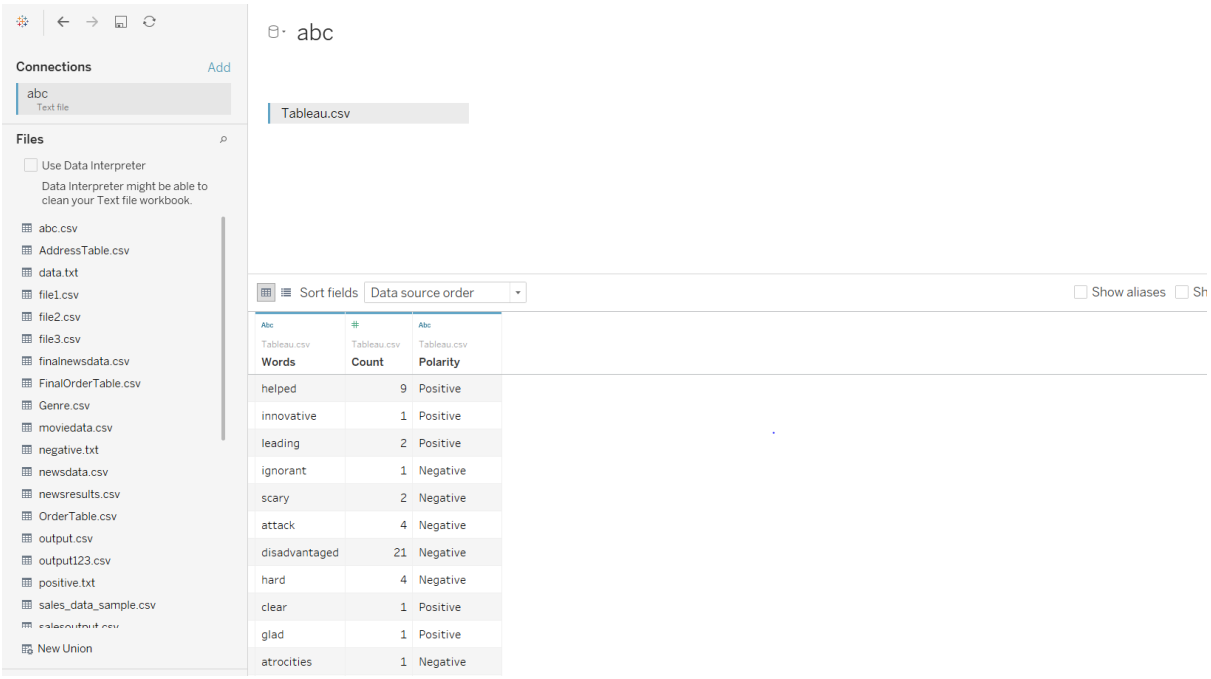
6. To create this visualization in tableau. First, I have created a new CSV file named “tableau.csv”. In that, I kept the Matched words with their respective total count. And use the same to visualize data in tableau. Below is the table for your reference.

	Words	Count	Polarity
2	helped	9	Positive
3	innovative	1	Positive
4	leading	2	Positive
5	ignorant	1	Negative
6	scary	2	Negative
7	attack	4	Negative
8	disadvants	21	Negative

To create this CSV file, I have used the below python script named as “tableau.py”. In this script, I have used the set function to take unique words under “Match” column of “sentiment.csv” file [2]. Below is the screenshot of it for your reference.

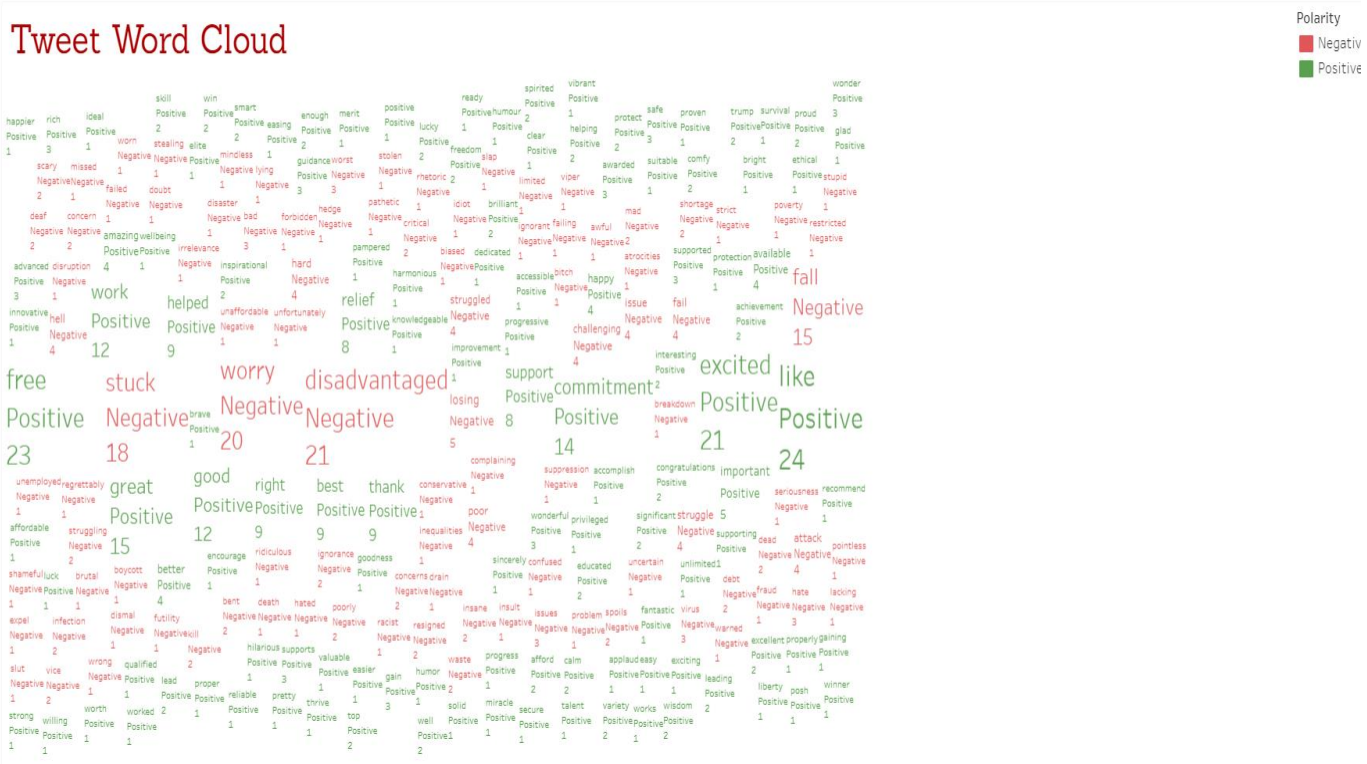
```
1    import csv
2
3    positive_list = [...]
259  negative_list = [...]
863  words=[]
864  words1 = []
865  headers = ["Words","Count","Polarity"]
866  with open('sentiment.csv', 'r') as csvfile:
867      reader = csv.reader(csvfile)
868      next(reader)
869      for row in reader:
870        csv_words = row[2].split(",")
871        for i in csv_words:
872          words.append(i.replace(" ",""))
873      test_list = list(filter(None, words))
874      for i in test_list:
875          x = test_list.count(i)
876          words1.append((i,x))
877          finalset = set(words1)
878  with open('Tableau.csv', 'w', newline='') as new_file:
879      csv_writer = csv.writer(new_file)
880      csv_writer.writerow(headers)
881      for key, value in finalset:
882          if key in positive_list:
883              polarity = "Positive"
884          elif key in negative_list:
885              polarity = "Negative"
886          else:
887              polarity = "Neutral"
888          csv_writer.writerow([key, value,polarity])
889          print(key,value,polarity)
890
```

After generating the CSV file, I downloaded the tableau desktop application [3]. Then I imported the CSV file and created a word cloud and bubble cloud [4]. I have enclosed all the python scripts, CSV files and screenshots in the **yaswanthchiruvella_B00849892/sentimental** folder. Below are the screenshots for your reference.

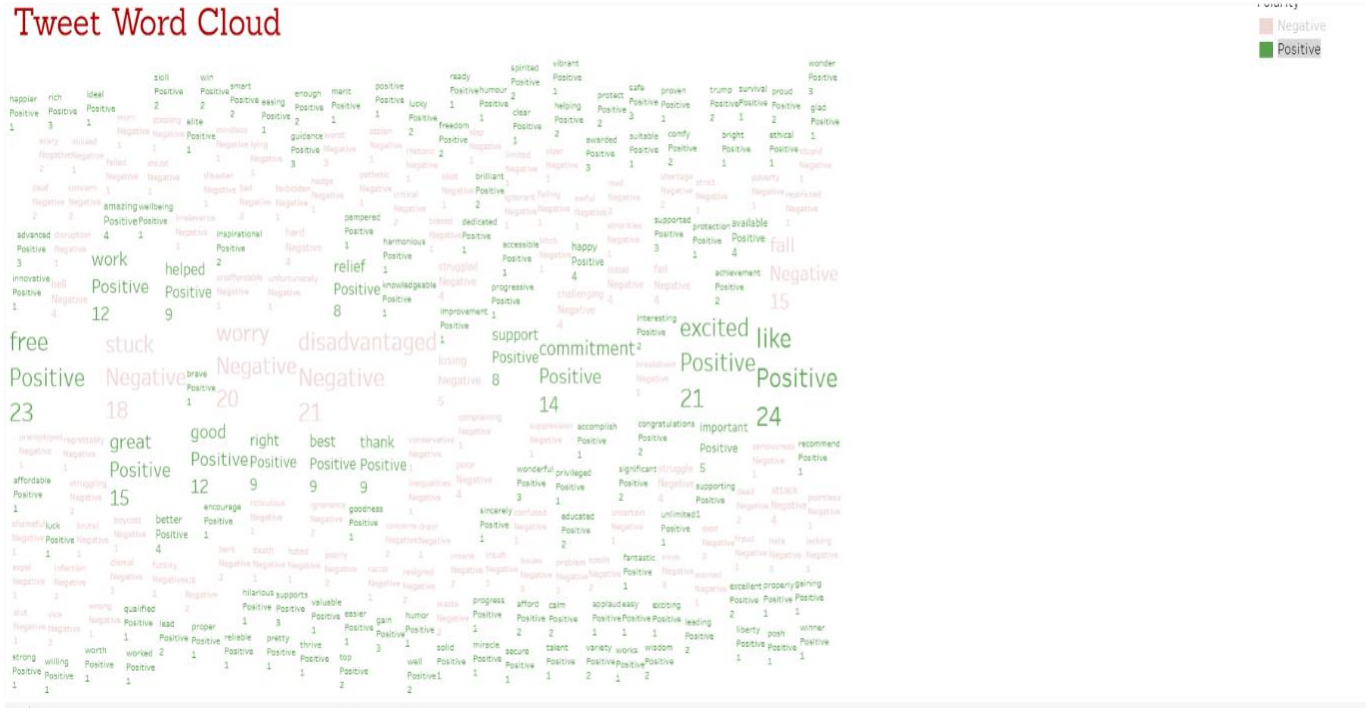


Words	Count	Polarity
helped	9	Positive
innovative	1	Positive
leading	2	Positive
ignorant	1	Negative
scary	2	Negative
attack	4	Negative
disadvantaged	21	Negative
hard	4	Negative
clear	1	Positive
glad	1	Positive
atrocities	1	Negative

The below screenshot shows both postive and negative words along with polarity and count.

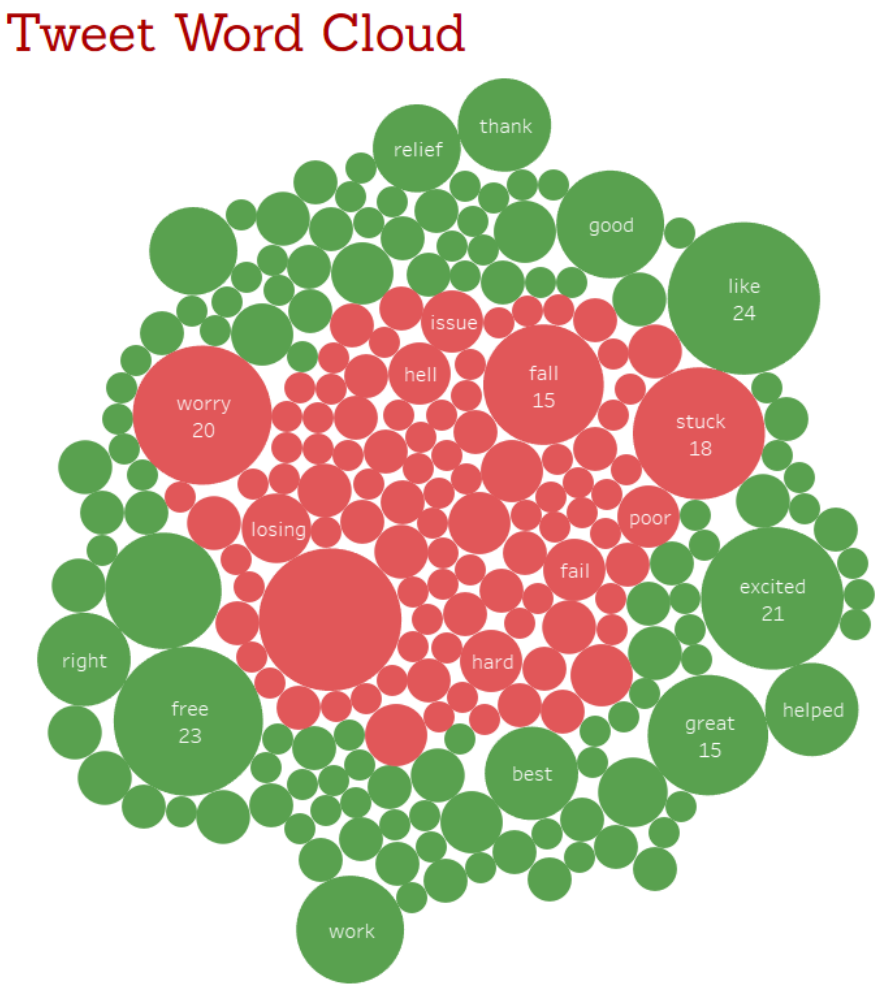


The below screenshot shows only postive words along with their count.



The below screenshot shows only Negative words along with their count.

The below screenshot visualizes the data in bubble cloud.



Section B: Semantic Analysis

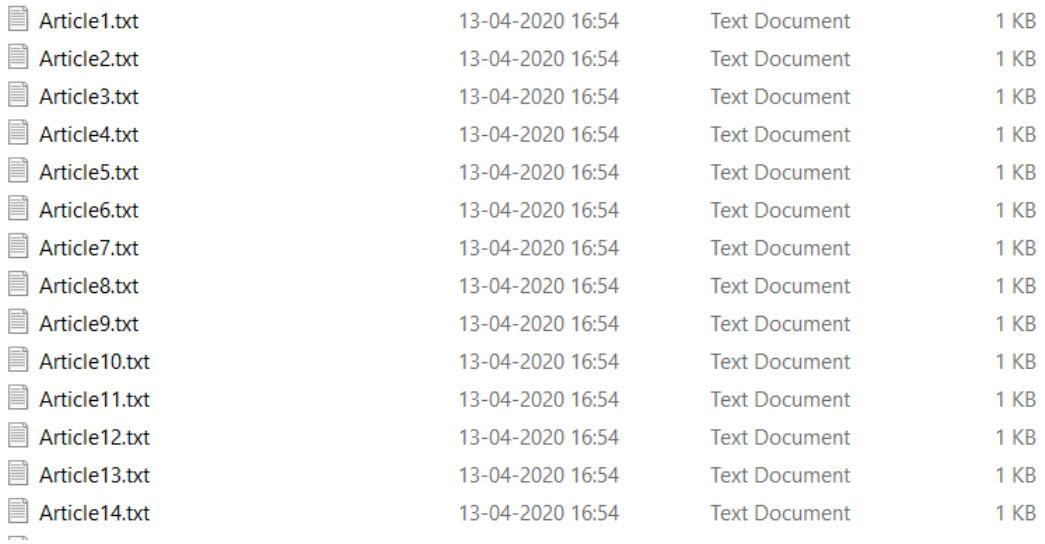
7 & 9. I have used the below python script named as “newsapi.py” to extract the news articles [5] and filtered the same at the time of extraction. I have rendered only title, content and description from the news article as mentioned in the assignment, and stored the three fields in a CSV file named newsdata.csv. I have enclosed the python script and CSV file in the **yaswanthchiruvella_B00849892/semantic** folder. Below are the screenshots for your reference.

```
import requests
import csv
import re

members = ['Canada', 'University', 'Dalhousie University', 'Halifax', 'Canada Education', 'Moncton', 'Toronto']

def filter(a):
    if (a is None):
        pass
    else:
        final_str1 = a.replace('\r\n', '')
        final_str2 = final_str1.encode('ascii', 'ignore').decode('ascii')
        final_str3 = remove_specialtags.sub(r'', final_str2)
        return final_str3


remove_specialtags = re.compile(r'https\S+|([^a-zA-Z\s]+?)')

with open('newsdata.csv', 'w', newline="") as f:
    csv_writer = csv.writer(f)
    for value in members:
        URL = 'http://newsapi.org/v2/everything?apiKey=77c95b9979754ec28605b5ed059439ef&page=1&q=' + str(value)

        # sending get request and saving the response as response object
        r = requests.get(url=URL)
        data = r.json()
        if ("articles" in data.keys()):
            for json in data['articles']:
                csv_writer.writerow([filter(json['title']), filter(json['description']), filter(json['content'])])

```

8. I have considered each news article as a document and stored the same in a text file. I have used the below script python script named as “Semantic.py” to accomplish the task. I have enclosed the python script in the **yaswanthchiruvella_B00849892/semantic** folder. Below is the screenshot for your reference.

```
# This script will consider each article as a document and stored the same in a text file
with open('newsdata.csv', 'r') as csvfile:
    line = csvfile.readlines()
    for row in range(len(line)):
        rowcount = rowcount+1
        file_name = "Article"+str(rowcount)+".txt"
        article = line[row].replace(',','\n')
        with open(os.path.join('C:\\Users\\rocki\PycharmProjects\csv\Semantic',file_name), 'w') as output:
            output.write(article)
```

The above script will populate the files in the “semantic” folder.



Article1.txt	13-04-2020 16:54	Text Document	1 KB
Article2.txt	13-04-2020 16:54	Text Document	1 KB
Article3.txt	13-04-2020 16:54	Text Document	1 KB
Article4.txt	13-04-2020 16:54	Text Document	1 KB
Article5.txt	13-04-2020 16:54	Text Document	1 KB
Article6.txt	13-04-2020 16:54	Text Document	1 KB
Article7.txt	13-04-2020 16:54	Text Document	1 KB
Article8.txt	13-04-2020 16:54	Text Document	1 KB
Article9.txt	13-04-2020 16:54	Text Document	1 KB
Article10.txt	13-04-2020 16:54	Text Document	1 KB
Article11.txt	13-04-2020 16:54	Text Document	1 KB
Article12.txt	13-04-2020 16:54	Text Document	1 KB
Article13.txt	13-04-2020 16:54	Text Document	1 KB
Article14.txt	13-04-2020 16:54	Text Document	1 KB

10a. To calculate TF-IDF, I have used the below python script named as "Semantic.py" which iterates through all documents in the folder and calculates how many documents containing a particular term that is 'Canada', 'University', 'Dalhousie University', 'Halifax', 'Business'. I have stored the same in a CSV file named "semanticA.csv". I have enclosed the python script and CSV file in the **yaswanthchiruvella_B00849892/semantic** folder. Below are the screenshots for your reference.

```
27  with open('semanticA.csv', 'w', newline='') as new_file:
28      csv_writer = csv.writer(new_file)
29      csv_writer.writerow(headers1)
30      for root, dirs, files in os.walk('C:\\Users\\rocki\\PycharmProjects\\csv\\Semantic'):
31          if files:
32              for filename in files:
33                  Total = len(files)
34                  with open(os.path.join('C:\\Users\\rocki\PycharmProjects\csv\Semantic', filename), 'r') as f:
35                      contents = f.read()
36                      print(contents)
37                      if words[0] in contents:
38                          print(words[0] +'found')
39                          CWordCount += 1
40                      if words[1] in contents:
41                          print(words[1] +'found')
42                          Uwordcount += 1
43                      if words[2] in contents:
44                          print(words[2] +'found')
45                          DuWordCount += 1
46                      if words[3] in contents:
47                          print(words[3] +'found')
48                          HWordCount += 1
49                      if words[4] in contents:
50                          print(words[4] +'found')
51                          Bwordcount += 1
52              csv_writer.writerow([words[0], CWordCount, str(Total)+'/'+str(CWordCount), math.log10(Total/CWordCount)])
53              csv_writer.writerow([words[1], Uwordcount, str(Total)+'/'+str(Uwordcount), math.log10(Total/Uwordcount)])
54              csv_writer.writerow([words[2], DuWordCount, str(Total)+'/'+str(DuWordCount), math.log10(Total/DuWordCount)])
55              csv_writer.writerow([words[3], HWordCount, str(Total)+'/'+str(HWordCount), math.log10(Total/HWordCount)])
56              csv_writer.writerow([words[4], Bwordcount, str(Total)+'/'+str(Bwordcount), math.log10(Total/Bwordcount)])
57          else:
58              print("Empty Directory")
```

The above script will generate csv file like this

	A	B	C	D	E
1	SearchQuery	Documents Containing Term (df)	Total Documents (N)/df	log10(N/df)	
2	Canada	157	510/157	0.511670524	
3	University	63	510/63	0.908229627	
4	Dalhousie University	10	510/10	1.707570176	
5	Halifax	45	510/45	1.054357662	
6	Business	9	510/9	1.753327667	

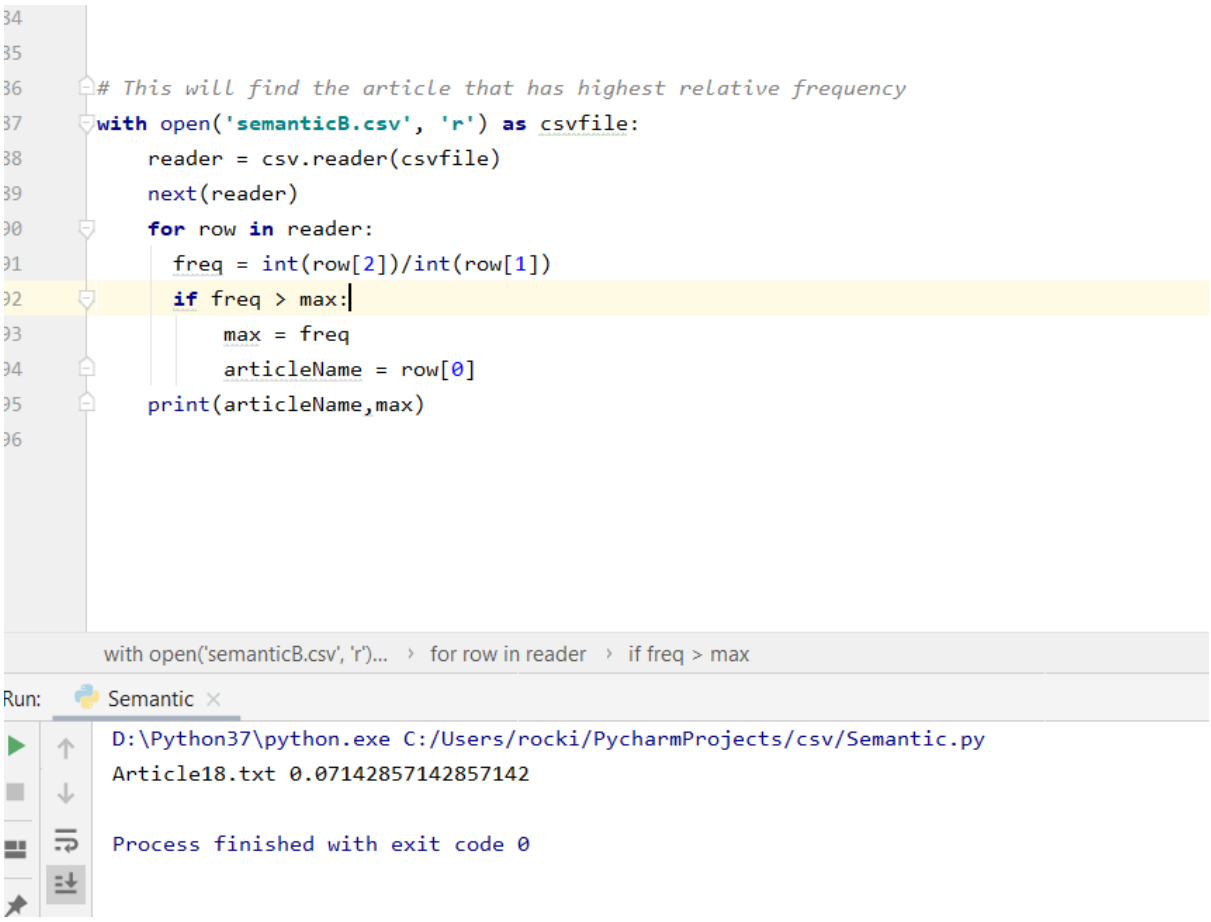
10b. To find the frequency of the word “Canada” and to calculate total word count in a document, I have used the below python script named as “Semantic.py”. I have stored the same in a CSV file named “semanticB.csv”. I have enclosed the python script and CSV file in the **yaswanthchiruvella_B00849892/semantic** folder. Below are the screenshots for your reference.

```
60  #This will calculate the total count of words in a document along with number of occurences of word "Canada"
61  with open('semanticB.csv', 'w', newline='') as new_file:
62      csv_writer = csv.writer(new_file)
63      csv_writer.writerow(headers2)
64      for root, dirs, files in os.walk('C:\\Users\\rocki\\PycharmProjects\\csv\\Semantic'):
65          if files:
66              for filename in files:
67                  CWordCount = 0
68                  with open(os.path.join('C:\\Users\\rocki\PycharmProjects\csv\Semantic',filename),'r') as f:
69                      contents = f.read()
70                      wordcount = len(contents.split())
71                  for m in pattern.finditer(contents):
72                      if m.group()==words[0]:
73                          CWordCount += 1
74                  if CWordCount:
75                      csv_writer.writerow([filename, wordcount, CWordCount])
76                      print(filename, CWordCount)
77          else:
78              print("Empty Directory")
```

The above will generate csv file like this

	A	B	C	D
1	Documents	Total Words (m)	Frequency (f)	
2	Article1.txt	100	5	
3	Article10.txt	62	2	
4	Article100.txt	91	1	
5	Article103.txt	91	2	
6	Article107.txt	90	1	
7	Article11.txt	93	3	
8	Article110.txt	94	4	
9	Article117.txt	53	2	
10	Article12.txt	75	2	
11	Article132.txt	93	2	
12	Article14.txt	89	2	
13	Article141.txt	102	1	
14	Article142.txt	89	6	
15	Article143.txt	95	4	
16	Article144.txt	94	2	
17	Article145.txt	93	1	
18	Article146.txt	87	6	
19	Article147.txt	92	5	
20	Article148.txt	97	1	
21	Article149.txt	102	2	
22	Article15.txt	96	2	
23	Article150.txt	101	7	
24	Article152.txt	98	6	
25	Article153.txt	82	4	
26	Article154.txt	99	6	
27	Article155.txt	94	3	
28	Article156.txt	90	3	

10C. To find the document that has the highest relative frequency (f/m) I have used the below python script named as “Semantic.py” which will calculate the f/m and display the file that has highest f/m value. I have enclosed the python script in the **yaswanthchiruvella_B00849892/semantic** folder. Below is the screenshot for your reference.



References

[1] Bing Liu, “Opinion Mining, Sentiment Analysis, Opinion Extraction,” *Uic.edu*, 2015. [Online]. Available: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon. [Accessed: 12-Apr-2020]

[2] “5. Data Structures — Python 3.8.2 documentation,” *Python.org*, 2020. [Online]. Available: https://docs.python.org/3/tutorial/datastructures.html#sets. [Accessed: 12-Apr-2020]

[3] “Tableau Public,” *Tableau Public*, 2020. [Online]. Available: https://public.tableau.com/en-us/s/. [Accessed: 12-Apr-2020]

[4] “How to Make a Word Cloud in Tableau - AbsentData,” *AbsentData*, 2018. [Online]. Available: https://www.absentdata.com/make-word-cloud-tableau/. [Accessed: 12-Apr-2020]

[5] News API - A JSON API for live news and blog articles, “News API - A JSON API for live news and blog articles,” *Newsapi.org*, 2020. [Online]. Available: https://newsapi.org/. [Accessed: 12-Apr-2020]