Report for the Programming for Data Analytics Project B

Stage 1 –Vocabulary Composition and Word Frequency Calculations Develop code for reading all tweets from both the positive and negative files.

For simplicity and quick run, I have shorten the file size during the implementation phase.

For the stage 1 task first, I read from the train folder both the trainPos.txt and trainNeg.txt files.

'''

: Here I have set the path directory for the train folder for the both files.

'''

pathPos = "C:/Users/Mohammed/Desktop/ProjectB/dataFiles/train/trainPos.txt"

pathNeg = "C:/Users/Mohammed/Desktop/ProjectB/dataFiles/train/trainNeg.txt"

#pathPos = "C:/Users/Mohammed/Desktop/ProjectB/dataFiles/test/testPos.txt"

#pathNeg = "C:/Users/Mohammed/Desktop/ProjectB/dataFiles/test/testNeg.txt"

```
In [8]: runfile('C:/Users/Mohammed/Desktop/ProjectB/project2.py', wdir='C:/Users/Mohammed/Desktop/
ProjectB')
<_io.TextIOWrapper name='C:/Users/Mohammed/Desktop/ProjectB/trainPos1.txt' mode='r' encoding='ISO-8859-1'>
@iShustoff ? ? 119 ??? ????? ????????? ??????  ??? ??? ???????????????? ? ??? ????? ?????????
@Su2ieQ13 But you're IMing with meeeeee.
"@apogeum whoooaa, thats soo awesome  my eyes look like black.. except if you have a yellow light bulb
close to my eyes then u can"
The shop of the day  http://bit.ly/RJbgs
```

Fig – 1: Opening and Reading the file screenshot

'''

: Here I opened and reading the file

: After reading the file I have tokenize the words

: After tokenize I have set unique words.

: param pathPos : is the variable for the trainPos file path directory

'''

filePositive = open(pathPos,'r', encoding = 'ISO-8859-1',newline='')

#print(filePositive)

resultPos = filePositive.read()

#print(resultPos)

```
beofreTokenizePos = len(resultPos)


posWords = word_tokenize(resultPos)

#print('After tokenize======\n',posWords)


afterTokenizePos = len(posWords)


#posWords = (stopwords.words('english'))

#print('After stop words ====\n',afterStopWordsPos)


posUniqueWords = set(posWords)

#print(posUniqueWords)

'''

   : Here I set the unique words as a key for the dictionary

'''

posDic = dict.fromkeys(posUniqueWords,0)

#print(posDic)
```

After open and reading the file and I have tokenize the words which is also part 4 of the project to clean the tweets.

```
'except', 'if', 'you', 'have', 'a', 'yellow', 'light', 'bulb', 'close', 'to', 'my', 'eyes', 'then', 'u',
'can', "'", 'The', 'shop', 'of', 'the', 'day', 'http', ':', '//bit.ly/RJbgs', "'", 'i', 'could', "n't",
'sleep', 'so', 'i', 'stayed', 'awake', 'watching', '@', 'lilbsuremusic', 'on', 'this', 'live', 'stream',
'thingy', 'and', 'now', 'i', "'m", 'taking', 'my', 'butt', 'to', 'bed', ',', 'so', 'sweet', 'dreams',
'``', '@', 'Lee_Knight', 'ok', 'haha', 'thanks', 'i', 'will', 'try', 'that', 'lol', "'", 'I', 'might',
'go', 'to', 'see', 'Angles', 'And', 'Demons', 'tonight', ',', 'to', 'see', 'what', 'all', 'the', 'hype',
'is', 'about', '.', 'I', 'will', 'let', 'ye', 'know', 'what', 'I', 'think', '.', '.', "'", '@',
```

Fig – 2: After Tokenize the words

```
'''

   : Here I added all the unique words to the dictionary which is 'posDic'

'''

for word in posWords:

   posDic[word] = posDic.get(word,0) + 1
```

#pprint.pprint(posDic)# this will show the word and number of frequency side by side

#print(word, '=======>', posDic[word]) # this will show the word and number of frequency side by side

##print("POSITIVE DICT")

#print(posDic)

'''

: Here I added all the unique words from the positive file.

'''

allPosWords = sum(posDic[item] for item in posDic)

#print(allPosWords)

Then I set the unique words for both file and I used set for this

```
In [10]: runfile('C:/Users/Mohammed/Desktop/ProjectB/project2.py', wdir='C:/Users/Mohammed/Desktop/
ProjectB')
{':', 'far', "'re", 'event', 'Su2ieQ13', 'I', 'might', 'thanks', 'you', 'sweet', 'brief', 'know', 'name',
'@', 'i', 'honest', 'dreams', 'awake', 'so', 'awesome', 'except', ',', 'safe', 'hour', 'Angles', 'b',
'rain', '```', 'out', 'then', 'yellow', 'bed', '?', 'an', 'could', 'http', 'Morning', 'half', 'to',
'ComedyQueen', 'riskin', 'Went', 'noticed', 'ye', '`', 'EmmaK67', 'thats', 'this', 's', 'at', 'light',
'on', 'watching', 'as', '119', 'killer', 'lilbsuremusic', 'u', 'Very', 'trip', 'today', 'shellistevens',
'just', 'shop', 'and', 'Ibiza', 'shower', 'be', 'jeffsure', 'live', 'Time', "'thats", 'course', 'all',
'now', 'SeagirlX', 'Lee_Knight', 'whoooaa', '23rd', 'eyes', 'hype', 'bulb', 'turned', 'iShustoff',
'Tings', "'ve", 'down', 'tonight', 'BLT', 'if', 'about', 'for', 'let', 'close', 'can', 'And', 'it', 'try',
'dang', 'though.lol', 'IMing', 'what', 'goin', 'with', 'taking', 'the', 'good', 'but', 'See', "'s",
```

Fig – 3 : Unique words train positive file

After setting the unique words I used unique words as a key for the dictionary and also got the number of frequency

```
{':': 1, 'far': 1, "'re": 1, 'event': 1, 'Su2ieQ13': 1, 'I': 3, 'might': 1, 'thanks': 1, 'you': 4,
'sweet': 1, 'brief': 1, 'know': 1, 'name': 1, '@': 10, 'i': 5, 'honest': 1, 'dreams': 1, 'awake': 1, 'so':
4, 'awesome': 1, 'except': 1, ',': 9, 'safe': 1, 'hour': 1, 'Angles': 1, 'b': 1, 'rain': 1, '```': 9,
'out': 2, 'then': 1, 'yellow': 1, 'bed': 1, '?': 66, 'an': 1, 'could': 1, 'http': 1, 'Morning': 1, 'half':
1, 'to': 4, 'ComedyQueen': 1, 'riskin': 1, 'Went': 1, 'noticed': 1, 'ye': 1, '`': 1, 'EmmaK67': 1,
'thats': 1, 'this': 1, 's': 1, 'at': 1, 'light': 1, 'on': 2, 'watching': 1, 'as': 1, '119': 1, 'killer':
1, 'lilbsuremusic': 1, 'u': 1, 'Very': 1, 'trip': 1, 'today': 1, 'shellistevens': 1, 'just': 1, 'shop': 1,
```

Fig – 4: Unique words and number of frequency

I did the same operation with the negative file in the train folder.

Stage 2 – Calculating Word Probability Calculations

To get the conditional probability as well as positive and negative file probability I have done couple of operation. I have add the both positive and negative file all words then I took out the unique words from the both file and their frequency

''' : Here will show the positive word probility '''

```
positive_word_probability ={}

for word, frequency in posDic.items():

    prob = (frequency+1)/(allPosWords + allWords)

    positive_word_probability[word] = prob

#print('Positive word probability\n', positive_word_probability)

result_positive_probability = len(positive_word_probability)
```

''' : Here will show the negative word probability '''

```
negative_word_probability ={}

for word, frequency in negDic.items():

    prob = (frequency+1)/(allNegWords + allWords)

    negative_word_probability[word] = prob

#print('Negative word probability\n', negative_word_probability)

result_negative_probability = len(negative_word_probability)
```

```
Positive word probability
 {':': 0.003189792663476874, 'far': 0.003189792663476874, "'re": 0.003189792663476874, 'event':
0.003189792663476874, 'Su2ieQ13': 0.003189792663476874, 'I': 0.006379585326953748, 'might':
0.003189792663476874, 'thanks': 0.003189792663476874, 'you': 0.007974481658692184, 'sweet':
0.003189792663476874, 'brief': 0.003189792663476874, 'know': 0.003189792663476874, 'name':
0.003189792663476874, '@': 0.017543859649122806, 'i': 0.009569377990430622, 'honest':
0.003189792663476874, 'dreams': 0.003189792663476874, 'awake': 0.003189792663476874, 'so':
0.007974481658692184, 'awesome': 0.003189792663476874, 'except': 0.003189792663476874, ',':
```

Fig – 5: Positive file words probability

```
Negative word probability
 {'interst': 0.0033222591362126247, ':': 0.0049833887043189366, 'are': 0.0049833887043189366, 'home':
0.0033222591362126247, 'bill': 0.0033222591362126247, 'their': 0.006644518272425249, 'thinking':
0.0033222591362126247, '//myloc.me/5dMB': 0.0033222591362126247, 'sh': 0.0033222591362126247, 'its':
0.0049833887043189366, 'Is': 0.0033222591362126247, 'text': 0.0033222591362126247, 'gone':
0.0033222591362126247, 'I': 0.0033222591362126247, 'cancelling': 0.0033222591362126247, 'got':
0.008305647840531562, 'would': 0.0033222591362126247, 'side': 0.0049833887043189366, 'over':
```

Fig – 6: Negative words Probability

Stage 3 – Classifying Unseen Tweets and Performing Basic Evaluation

After building the model I passed new tweet and based on the model it will show the output result whether tweet was positive or negative.

''' : Here I would take new tweet to test : the level of accuracy and its probability '''

```
negative = 1

positive = 1

sentence = input('Enter a new tweet:')

sentence = sentence.lower()

newWords = sentence.split(' ')

for word in newWords:

    if word in negative_word_probability:

        negative = negative * negative_word_probability[word]

    if word in positive_word_probability:

        positive = positive * positive_word_probability[word]

print("\nAccuracy Level :\n", 'Positive: ', positive, '\nNegative: ', negative, '\n')

if positive > negative:

    print("Tweet was Positive")

else:

    print("Tweet was Negative")
```

```
Enter a new tweet:How are you today?

Accuracy Level :
 Positive:  0.007974481658692184
Negative:  1.655610865222238e-05

Tweet was Positive
```

Fig – 7: Positive tweet result from training file

```
Enter a new tweet:How are you today?

Accuracy Level :
 Positive:  4.2042086731841285e-08
Negative:  1.6378499045169084e-08

Tweet was Positive
```

Fig – 7.1: Positive tweet result from full running training file

```
Enter a new tweet:I do not like hot.

Accuracy Level :
 Positive:  1.4604943400915153e-07
Negative:  3.2085481884151905e-07

Tweet was Negative
```

Fig – 8: Negative tweet result from training file

```
Enter a new tweet:i do not like hot.

Accuracy Level :
 Positive:  8.088299542237629e-17
Negative:  8.782507652384451e-16

Tweet was Negative
```

Fig – 8.1: Negative tweet result from full running training file

Now running the test folder files both positive and negative files with new tweet and the result and accuracy level if completely different.

```
Enter a new tweet:How are you today?

Accuracy Level :
 Positive:  1.5364337575979396e-08
Negative:  1.0954117230298596e-08

Tweet was Positive
```

Fig – 9: Positive tweet result from test file

```
Enter a new tweet:I do not like hot.

Accuracy Level :
 Positive:  3.9971126889397315e-11
Negative:  2.3850353887595446e-10

Tweet was Negative
```

Fig – 9: Negative tweet result from test file

Stage 4 - Additional Elements This section is for the investigations of the impact of some pre-processing techniques on the accuracy. Common techniques include lowering the case of all words, punctuations removal, stop-word removal, n-grams, etc.

For this I have used nltk tokenizer and corpus to remove the stop words to clean the tweet before the get the unique words and number of frequency as well as probability.


```
{'are', 'd', 'their', 'hadn', 'ma', 'm', 'against', 'its', "mustn't", "needn't", 'off', 'you', 'her',
'whom', 'over', 'these', 'he', 'i', "hasn't", 'during', 'am', 'isn', 'again', 'themselves', 'so',
"that'll", 'shouldn', 'which', "shan't", 'hasn', "should've", 'through', 'those', 'after', 'having',
"won't", 'll', 'himself', 'out', 'then', 'very', 'above', 'few', 'aren', 'being', 'an', 'further',
'herself', 'does', 'to', 'myself', 'y', 'do', 'this', 's', "didn't", 'at', 'as', 'theirs', 'on', 'we',
"you're", 'just', "haven't", 'had', 'and', 'couldn', 'ourselves', 'be', 'by', 'below', 'wasn', 'ain',
'all', "weren't", 'weren', 'now', 'nor', 'been', 'doing', 'was', 'down', "couldn't", 'wouldn', "you've",
'if', 'about', "she's", 'for', "don't", 'yourselves', 'can', 'too', "mightn't", 'it', "you'd", 'other',
'both', 'shan', 'from', 'most', 'some', 'than', 'they', 'won', "doesn't", 'under', 'until', "you'll", 'o',
"wouldn't", 'should', 'what', 'mightn', 'because', 'once', 't', 'with', 'such', 'the', 'why', 'were',
```

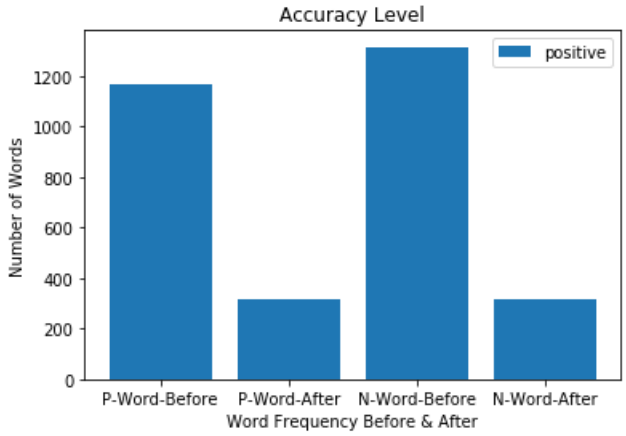

Fig – 10: After stop words remove

Stage 5 – Visualization

Here in the bar chart you can see from the training file the accuracy level for the positive and negative words.

```
''' : Here am showing the bar chart for the word before and after tokenize and their

   : accuracy level. '''

x_axis = ['P-Word-Before', 'P-Word-After', 'N-Word-Before', 'N-Word-After']

y_axis = [beofreTokenizePos,afterTokenizePos,beofreTokenizeNeg,afterTokenizePos]

ind = np.arange(len(x_axis))

plt.bar(ind, y_axis)

plt.xticks(ind , x_axis)

plt.title('Accuracy Level')

plt.xlabel('Word Frequency Before & After')

plt.ylabel('Number of Words')

plt.legend(['positive', 'negative'], loc='best')

plt.show()


#============= New tweet accuracy and probability=============================

'''

   : Here am showing the bar chart of the new tweet accuracy and probability

'''

x_bar = ['Positive-Tweet-Accuracy', 'Negative-Tweet-Accuracy']

y_bar = [positive, negative]

level = np.arange(len(x_bar))


plt.bar(level, y_bar)

plt.xticks(level, x_bar)

plt.title('Probability New Tweet Accuracy Level')

plt.xlabel('Positive or Negative')

plt.ylabel('Probability Level')

plt.show()
```

```
#==============================Line Graph ====================================
''': Here am showing the line graph of the new tweet accuracy and probability '''
wordsBeforePos = ([beofreTokenizePos, beofreTokenizeNeg])
wordsAfterPos = ([afterTokenizePos, afterTokenizePos])
plt.plot(wordsBeforePos)
plt.plot(wordsAfterPos)
plt.xlabel('Before word Tokenize')
plt.ylabel('After word tokenize')
plt.title('Before Accuracy')
plt.legend(['positive', 'negative'], loc='best')
plt.show()
```



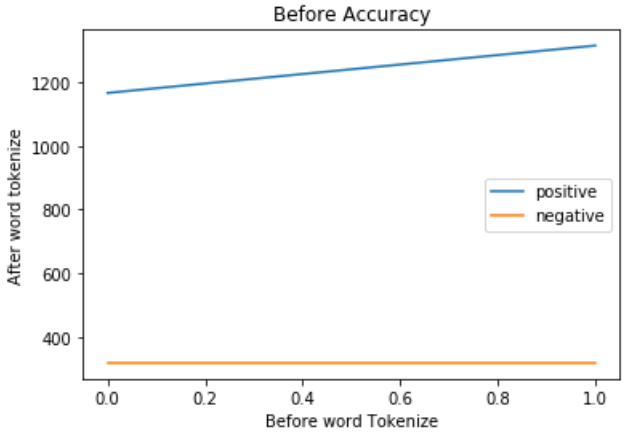
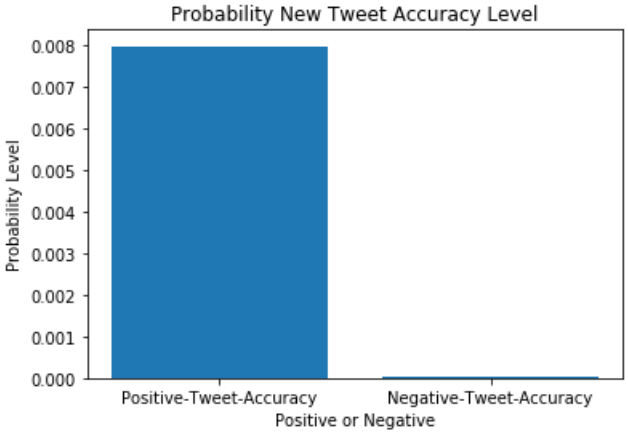


Fig – 11 Words Accuracy Level

Here am showing the probability level of accuracy from the training file positive and negative for new tweet.

```
Enter a new tweet:How are you today?

Accuracy Level :
 Positive:  0.007974481658692184
Negative:  1.655610865222238e-05

Tweet was Positive
```




```
Enter a new tweet:I do not like hot.

Accuracy Level :
 Positive:  1.4604943400915153e-07
Negative:  3.2085481884151905e-07

Tweet was Negative
```

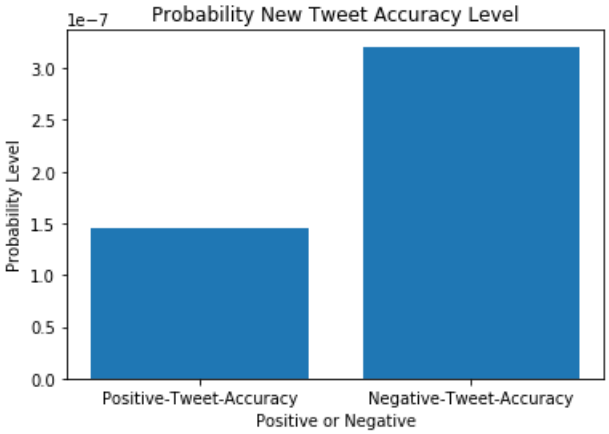


Fig – 12 New Tweet Accuracy Level for Both Positive & Negative

With this project certainly there, lots can be improve