FIT5196 Task 2 in Assessment 1

Student Name: Wei Chen

Student ID: 24719927

Date: 7/9/2020

Version: 1.0

Environment: Python 3 and Jupyter notebook

1. Import libraries

In [ ]:

```
import pandas as pd
import nltk
import langid
import re
from nltk.probability import *
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
```

2. Read excel file

In [ ]:

```
xlsx = pd.ExcelFile('24719927.xlsx')
```

In [ ]:

```
# Find out the total number of excel sheets
sheets_num = len(xlsx.sheet_names)
sheets_num
```

In [ ]:

```
# Find out the dates of tweets
xlsx.sheet_names[0:5]
```

In [ ]:

```
# Read the first worksheet in excel for testing
xlsx.parse(0)
```

3. Pre-process the tweets from excel

3.1 Extract tweet content

In [ ]:

```
# Create an empty list to store the content of the tweets
tweet_list =[]
for num in range(81):
    #read the content of tweets in each worksheets and get rid of all the NA value
    limitPer = len(sheet1)*0.9
    tweet_raw= xlsx.parse(num).dropna(how="all").dropna(thresh=limitPer,axis=1)# grab columns with
    tweet_new = tweet_raw.iloc[:,0]
    tweet_new = tweet_new.values.tolist()
    tweet_list.append(tweet_new)

#Remove "text" in the list
for tweets in tweet_list:
        if tweets[0] == "text":
            tweets.remove(tweets[0])
```

3.2 Remove non-English tweet

In [ ]:

```
for num1 in range(81):
    for num2 in range(len(tweet_list[num1])):
        # Find contents only contains numeric value and replace them by ""
        if type(tweet_list[num1][num2]) == type(0):
            tweet_list[num1][num2] = ""
        # Find contents that are not English and replace them by ""
        elif langid.classify(tweet_list[num1][num2])[0] != "en":
            tweet_list[num1][num2] = ""
```

In [ ]:

```
tweet_list[0]
```

In [ ]:

```
# Rmove all the "" value in each tweets for each day
for num1 in range(81):
    while("" in tweet_list[num1]) :
        tweet_list[num1].remove("")
```

In [ ]:

```
tweet_list[0]
```

3.3 Create doc-term matrix

In [ ]:

```
# I dont know how to generate sparsen representation like the sample example, this is all I know abo

array=[]
shape=[]
vectorizer = CountVectorizer()
# tokenize and build vocab
for num in range(81):
    for content in tweet_list[num]:
        vectorizer.fit([content])
        vector = vectorizer.transform([content])
        array.append(vector.toarray())
        shape.append(vector.shape)
```

3.4 Tokenize the processed tweets

In [ ]:

```
for num1 in range(81):
    # Join the content of the tweets for each day in the list
    tweet_list[num1] = ''.join(map(str, tweet_list[num1]))
    # Covert to lowercase
    tweet_list[num1] = tweet_list[num1].lower()
```

In [ ]:

```
from nltk.tokenize import RegexpTokenizer
#Tokenize the content of tweets
tokenizer = RegexpTokenizer(r"[a-zA-Z]+(?:[-'][a-zA-Z]+)?")
for num1 in range(81):
        tweet_list[num1] = tokenizer.tokenize(tweet_list[num1])
```

In [ ]:

```
# display tokens for first day
tweet_list[0]
```

3.5 Remove stopwords from tweets tokens

In [ ]:

```
# Create stopwords list
stopwords = []
with open('./stopwords_en.txt') as f:
    stopwords = f.read().splitlines()
stopwords = set(stopwords)
```

In [ ]:

```
# Remove stopwords
for num1 in range(81):
    tweet_list[num1]= [word for word in tweet_list[num1] if not word in list(stopwords)]
```

3.6 Remove tokens with length less than 3

In [ ]:

```
# Remove tokens with the length less than 3
for num1 in range(81):
    tweet_list[num1]=[word for word in tweet_list[num1] if len(word)>2]
```

3.7 Remove rare tokens and context-dependent tokens

In [ ]:

```
# This cell took 10 hours to run, so I have saved the results into txt files, after that I turned o
# The lists of raretoken and context-dependent token are read from the txt files, but the results a

# Create list to store rare words and context dependent words
raretoken=[]
contextdep=[]
# Set initial frequence to 0
freq=0
# Create for loops to check the frequency of each word in all worksheets
for num1 in range(81):
    # For loop to get each word from each worksheet
    for word in set(tweet_list[num1]):
        freq=0
        # For loop to check the frequency of each word
        for num2 in range(81):
            if word in set(tweet_list[num2]):
                freq+=1
        if freq <5 and not in raretoken:
            raretoken.append(word)
        if freq>60 and not in contextdep:
            contextdep.append(word)
```

In [ ]:

```
# Find out total number of rare tokens
len(raretoken)
```

In [ ]:

```
# Display top 20 rare tokens
raretoken[0:20]
```

In [ ]:

```
# Find out total number of context dependent tokens
len(contextdep)
```

In [ ]:

```
# Display top 20 tokens
contextdep[0:20]
```

In [ ]:

```
# Combine rare token and contex-dependent lists
token=raretoken+contextdep
```

In [ ]:

```
# This cell took 4 hours to run
# Remove rare tokens and context-dependent tokens
for num1 in range(81):
    tweet_list[num1]= [word for word in tweet_list[num1] if not word in token]
```

In [ ]:

```
# copy a new list in case of an accident
tweet_list2 = tweet_list.copy()
```

3.8 Stem the tokens with porter stemmer

In [ ]:

```
#Poterstemmer to stem the tokens
ps = PorterStemmer()
for num in range(81):
     tweet_list2[num]=[ps.stem(tweet) for tweet in tweet_list2[num]]
```

4. Generate required ouput

4.1 Generate first 200 meaningful bigrams

In [ ]:

```
# Find out first 200 meaningful bigrams with PMI measure
meaningful_big=[]
bigram_measures = nltk.collocations.BigramAssocMeasures()
for num in range(81):
    finder = nltk.collocations.BigramCollocationFinder.from_words(tweet_list2[num])
    meaningful_big.append(finder.nbest(bigram_measures.pmi, 200))
```

In [ ]:

```
# Display the result
meaningful_big
```

4.2 Generate 100 most frequent bigrams

In [ ]:

```
# Create 100 most frequent bigrams
blist=[]
from nltk.util import ngrams
for num in range(81):
    bigrams = ngrams(tweet_list2[num], n = 2)
    fdbigram = FreqDist(bigrams)
    blist.append(fdbigram.most_common(100))
```

In [ ]:

```
# Write bigrams to txt file
bigram_txt = open("24719927_100bi.txt",'w')
for num in range(81):
    bigram_txt.write(str(date[num]) + ':' + str(blist[num]) + '\n')
bigram_txt.close
```

4.3 Generate 100 most frequent unigrams

In [ ]:

```
# Create 100 most frequent unigrams
ulist=[]
for num in range(81):
    unigrams = ngrams(tweet_list2[num], n =1)
    fdbigram = FreqDist(unigrams)
    ulist.append(fdbigram.most_common(100))
```

In [ ]:

```
# Write unigrams to text file
unigram_txt = open("24719927_100uni.txt",'w')
for num in range(81):
    unigram_txt.write(str(date[num]) + ':' + str(ulist[num]) + '\n')
unigram_txt.close
```

4.4 Gnerate word frequency count

In [ ]:

```
# Count word frequence for all the words in the tweets
wordfreq = {}
for content in tweet_list2:
    for token1 in content:
        if token1 not in wordfreq.keys():
            wordfreq[token1] = 1
        else:
            wordfreq[token1] += 1
```

In [ ]:

```
# Sort dictionary alphabetically
count=[]
sortedDict = dict( sorted(wordfreq.items(), key=lambda x: x[0].lower()) )

for k,v in sortedDict.items():
    count.append('{}:{}'.format(k,v))

wordcount_txt = open("24719927_vocab.txt",'w')
# Write word count to text file
for word in count:
    wordcount_txt.write(str(word)+'\n')
    wordcount_txt.close
```

In [ ]:

```
# Diplay top 10 word counts
count[0:10]
```