Enis_Mert_Kuzu_theoretical1

February 3, 2023

1 Tweet classification with naive bayes

For this notebook we are going to implement a naive bayes classifier for classifying positive or negative based on the words in the tweet. Recall that for two events A and B the bayes theorem says

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where P(A) and P(B) is the ***class probabilities*** and P(B|A) is called ***conditional probabilities***. this gives us the probability of A happening, given that B has occurred. So as an example if we want to find the probability of "is this a positive tweet given that it contains the word"good" " we will obtain the following

$$P(\text{"positive"}|\text{"good" in tweet}) = \frac{P(\texttt{"}\text{"good" in tweet}|\text{"positive"})P(\text{"positive"})}{P(\texttt{"}\text{"good" in tweet})}$$

This means that to find the probability of "is this a positive tweet given that it contains the word"good" " we need the probability of "good" being in a positive tweet, the probability of a tweet being positive and the probability of "good" being in a tweet.

Similarly if we want to obtain the opposite "is this a negative tweet given that it contains the word"boring" " we get

$$P(\text{"negative"}|\text{"boring" in tweet}) = \frac{P(\text{"boring" in tweet}|\text{"negative"})P(\text{"negative"})}{P(\text{"boring" in tweet})}$$

where we need the probability of "boring" being in a negative tweet, the probability of a tweet negative being and the probability of "boring" being in a tweet.

We can now build a classifier where we compare those two probabilities and whichever is the larger one it's classified as

if P("positive"|"good" in tweet) > P("negative"|"boring" in tweet)

Tweet is positive

else

Tweet is negative

Now let's expand this to handle multiple features and put the Naive assumption into bayes theroem. This means that if features are independent we have

$$P(A, B) = P(A)P(B)$$

This gives us:

$$P(A|b_1, b_2, ..., b_n) = \frac{P(b_1|A)P(b_2|A)...P(b_n|A)P(A)}{P(b_1)P(b_2)...P(b_n)}$$

or

$$P(A|b_1, b_2, ..., b_n) = \frac{\prod_i^n P(b_i|A)P(A)}{P(b_1)P(b_2)...P(b_n)}$$

So with our previous example expanded with more words "is this a positive tweet given that it contains the word"good" and "interesting" " gives us

$$P(\text{"positive"}|\text{"good", "interesting" in tweet}) = \frac{P(\text{"good" in tweet}|\text{"positive"})P(\text{"interesting" in tweet}|\text{"positive"})}{P(\text{"good" in tweet})P(\text{"interesting" in tweet})}$$

As you can see the denominator remains constant which means we can remove it and the final classifier end up

$$y = argmax_A P(A) \prod_i^n P(b_i|A)$$

The dataset that you will be working with can be downloaded from the following link: https://uppsala.instructure.com/courses/66466/files

[1]:
```python
#stuff to import
import pandas as pd
import numpy as np
import random
import sklearn
from sklearn.model_selection import train_test_split
```

Load the data, explore and pre-processing

[2]:
```python
tweets=pd.read_csv('twitter_sentiment_analysis.csv',encoding='latin',
                   names = ['sentiment','id','date','query','user','tweet'])
tweets
```

[2]:
```
         sentiment          id                          date     query  \
0                0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1                0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
2                0  1467810917  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY
```

```
3               0  1467811184  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
4               0  1467811193  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
…             …         …                            …       …
1599995         4  2193601966  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599996         4  2193601969  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599997         4  2193601991  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599998         4  2193602064  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599999         4  2193602129  Tue Jun 16 08:40:50 PDT 2009  NO_QUERY

                   user                                              tweet
0        _TheSpecialOne_  @switchfoot http://twitpic.com/2y1zl - Awww, t…
1          scotthamilton  is upset that he can't update his Facebook by …
2               mattycus  @Kenichan I dived many times for the ball. Man…
3                ElleCTF    my whole body feels itchy and like its on fire
4                 Karoli  @nationwideclass no, it's not behaving at all…
…                    …                                                …
1599995  AmandaMarie1028  Just woke up. Having no school is the best fee…
1599996      TheWDBoards  TheWDB.com - Very cool to hear old Walt interv…
1599997           bpbabe  Are you ready for your MoJo Makeover? Ask me f…
1599998     tinydiamondz  Happy 38th Birthday to my boo of alll time!!! …
1599999   RyanTrevMorris  happy #charitytuesday @theNSPCC @SparksCharity…

[1600000 rows x 6 columns]
```

[3]:
```
tweets = tweets.sample(frac=1)
tweets = tweets[:200000]
print("Dataset shape:", tweets.shape)
```

```
Dataset shape: (200000, 6)
```

[4]:
```
tweets['sentiment'].unique()
```

[4]:
```
array([0, 4], dtype=int64)
```

Currently (0 = negative and 4 = positive) changing the notation to (0 = negative and 1 = positive)

[5]:
```
tweets['sentiment']=tweets['sentiment'].replace(4,1)
tweets
```

[5]:
```
         sentiment          id                          date     query  \
400068           0  2057315774  Sat Jun 06 12:46:28 PDT 2009  NO_QUERY
949261           1  1823685288  Sat May 16 22:40:12 PDT 2009  NO_QUERY
969256           1  1827935321  Sun May 17 11:45:08 PDT 2009  NO_QUERY
504087           0  2188029612  Mon Jun 15 21:11:07 PDT 2009  NO_QUERY
1543485          1  2181375516  Mon Jun 15 11:44:36 PDT 2009  NO_QUERY
…             …         …                            …       …
893186           1  1691783137  Sun May 03 18:36:00 PDT 2009  NO_QUERY
```

```
805761           1  1468712754  Tue Apr 07 03:19:02 PDT 2009  NO_QUERY
1210674          1  1989073863  Mon Jun 01 00:15:36 PDT 2009  NO_QUERY
862490           1  1676910477  Fri May 01 23:40:37 PDT 2009  NO_QUERY
730945           0  2263700670  Sun Jun 21 01:33:29 PDT 2009  NO_QUERY

                    user                                              tweet
400068   LilMissDayDream  I officially hate JB haters!!!The brothers you…
949261   swollenabattoir  omg gonna get new shirts from the US!  green d…
969256      crucesignati  @jordancrockett1 just the verse I was reading …
504087         tonyvisme  @kellbell68 damn I thought you were talking to…
1543485     typicalrouse               @spacebetween41 ha same to you dear
…                    …                                                 …
893186           e_wills    @LeAnnBu lol nahh but wen i do i kan handle it
805761           niknakx  i got the twilight 2-disc special edition yest…
1210674         KellyKamp  Pinkpopdag 2 was vette shit: Volbeat, Placebo …
862490      coconoirgifts  Thank you to everyone who has been following m…
730945       erinebreslin                I just ate so much Mac and cheese!

[200000 rows x 6 columns]
```

Removing the unnecessary columns.

[6]:
```
tweets.drop(['date','query','user'], axis=1, inplace=True)
tweets.drop('id', axis=1, inplace=True)
tweets.head(10)
```

[6]:
```
         sentiment                                              tweet
400068           0  I officially hate JB haters!!!The brothers you…
949261           1  omg gonna get new shirts from the US!  green d…
969256           1  @jordancrockett1 just the verse I was reading …
504087           0  @kellbell68 damn I thought you were talking to…
1543485          1              @spacebetween41 ha same to you dear
1033221          1                    birthday tomorrow cant`t wait
506399           0  @madisonlee13 promiswe you'll never leave? bec…
506522           0         @1Aprella1 Sorry to hear that  That blows!
740809           0  Just missed 'No Country for old men&quot; on H…
832429           1  @ddlovato dont sleep!  You shouldnt miss ANYTH…
```

Checking if any null values present

[7]:
```
(tweets.isnull().sum() / len(tweets))*100
```

[7]:
```
sentiment    0.0
tweet        0.0
dtype: float64
```

Now make a new column for side by side comparison of new tweets vs old tweets

[8]:
```
#converting pandas object to a string type
tweets['tweet'] = tweets['tweet'].astype('str')
```

Check the number of positive vs. negative tagged sentences

```
[9]: positives = tweets['sentiment'][tweets.sentiment == 1 ]
     negatives = tweets['sentiment'][tweets.sentiment == 0 ]

     print('Total length of the data is:         {}'.format(tweets.shape[0]))
     print('No. of positve tagged sentences is:  {}'.format(len(positives)))
     print('No. of negative tagged sentences is: {}'.format(len(negatives)))
```

```
Total length of the data is:         200000
No. of positve tagged sentences is:  100091
No. of negative tagged sentences is: 99909
```

```
[11]: # nltk
      import nltk
      from nltk.stem import WordNetLemmatizer
      from nltk.corpus import stopwords
      from nltk.tokenize import word_tokenize
      #Stop Words: A stop word is a commonly used word (such as "the", "a", "an",␣
       ↪"in")
      #that a search engine has been programmed to ignore,
      #both when indexing entries for searching and when retrieving them as the␣
       ↪result of a search query.
      nltk.download('stopwords')
      nltk.download('punkt')
      nltk.download('omw-1.4')
      nltk.download('wordnet')
      stopword = set(stopwords.words('english'))
      print(stopword)
```

```
{'all', 'couldn', 'most', 'being', 've', 'ourselves', 'with', 'this', 'no',
'but', 'at', 'his', 'do', 'we', 'had', 'against', 'above', 'for', 'there',
'himself', "wasn't", 'more', 'here', 'wouldn', 'yours', "you'll", 'm',
"needn't", 'me', 'off', 'doesn', 'am', 'until', 'is', 'any', 'didn', 'he',
'and', 'haven', 'myself', 'should', "mustn't", 'between', 're', "she's", 'your',
"couldn't", 'an', "isn't", 'their', "shouldn't", "shan't", 'don', 'whom', 'so',
"aren't", 'if', 'y', 'than', 'of', 'very', 'few', 'hadn', 'why', 'the', 'was',
'because', 'has', 'it', "should've", 'own', "that'll", "hadn't", 'did',
'further', 'as', 'in', 'does', 'our', 'then', 'such', 'mightn', "you're", 'isn',
'over', 'wasn', 'will', 'same', 'shouldn', "wouldn't", 'down', 'by', 'been',
'before', "won't", 'just', 'yourselves', 's', "you've", 'were', 'both', 'from',
'weren', 'under', 'theirs', 'on', 'through', 'again', 'up', 'these', 'each',
'him', 'which', 'my', 't', "mightn't", 'having', 'll', 'them', 'mustn', 'not',
'that', 'yourself', 'are', 'now', 'how', 'its', 'to', 'd', 'once', 'other',
'only', "don't", 'where', 'shan', "weren't", 'you', 'o', 'into', 'aren',
"didn't", 'ma', 'about', "you'd", 'be', 'during', "hasn't", 'itself', 'won',
'out', 'a', 'when', 'doing', 'below', 'or', "doesn't", 'ours', 'who', 'have',
'herself', 'after', 'ain', 'her', 'what', 'needn', 'hasn', 'she', "it's",
"haven't", 'i', 'too', 'while', 'hers', 'those', 'some', 'themselves', 'they',
'nor', 'can'}

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\emert\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\emert\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\emert\AppData\Roaming\nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\emert\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

Data Cleaning

[17]:
```
import warnings
warnings.filterwarnings('ignore')
import re
import string
import pickle
urlPattern = r"((http://)[^ ]*|(https://)[^ ]*|( www\.)[^ ]*)"
userPattern = '@[^\s]+'
some = 'amp,today,tomorrow,going,girl'
def process_tweets(tweet):
  # Lower Casing
    tweet = re.sub(r"he's", "he is", tweet)
    tweet = re.sub(r"there's", "there is", tweet)
    tweet = re.sub(r"We're", "We are", tweet)
    tweet = re.sub(r"That's", "That is", tweet)
    tweet = re.sub(r"won't", "will not", tweet)
    tweet = re.sub(r"they're", "they are", tweet)
    tweet = re.sub(r"Can't", "Cannot", tweet)
    tweet = re.sub(r"wasn't", "was not", tweet)
    tweet = re.sub(r"don\x89Ûªt", "do not", tweet)
    tweet = re.sub(r"aren't", "are not", tweet)
    tweet = re.sub(r"isn't", "is not", tweet)
    tweet = re.sub(r"What's", "What is", tweet)
    tweet = re.sub(r"haven't", "have not", tweet)
    tweet = re.sub(r"hasn't", "has not", tweet)
    tweet = re.sub(r"There's", "There is", tweet)
    tweet = re.sub(r"He's", "He is", tweet)
    tweet = re.sub(r"It's", "It is", tweet)
    tweet = re.sub(r"You're", "You are", tweet)
    tweet = re.sub(r"I'M", "I am", tweet)
    tweet = re.sub(r"shouldn't", "should not", tweet)
    tweet = re.sub(r"wouldn't", "would not", tweet)
    tweet = re.sub(r"i'm", "I am", tweet)
    tweet = re.sub(r"I\x89Ûªm", "I am", tweet)
    tweet = re.sub(r"I'm", "I am", tweet)
    tweet = re.sub(r"Isn't", "is not", tweet)
    tweet = re.sub(r"Here's", "Here is", tweet)
    tweet = re.sub(r"you've", "you have", tweet)
    tweet = re.sub(r"you\x89Ûªve", "you have", tweet)
    tweet = re.sub(r"we're", "we are", tweet)
    tweet = re.sub(r"what's", "what is", tweet)
    tweet = re.sub(r"couldn't", "could not", tweet)
    tweet = re.sub(r"we've", "we have", tweet)
    tweet = re.sub(r"it\x89Ûªs", "it is", tweet)
    tweet = re.sub(r"doesn\x89Ûªt", "does not", tweet)
    tweet = re.sub(r"It\x89Ûªs", "It is", tweet)
    tweet = re.sub(r"Here\x89Ûªs", "Here is", tweet)
    tweet = re.sub(r"who's", "who is", tweet)
    tweet = re.sub(r"I\x89Ûªve", "I have", tweet)
    tweet = re.sub(r"y'all", "you all", tweet)
    tweet = re.sub(r"can\x89Ûªt", "cannot", tweet)
    tweet = re.sub(r"would've", "would have", tweet)
    tweet = re.sub(r"it'll", "it will", tweet)
    tweet = re.sub(r"we'll", "we will", tweet)
    tweet = re.sub(r"wouldn\x89Ûªt", "would not", tweet)
    tweet = re.sub(r"We've", "We have", tweet)
    tweet = re.sub(r"he'll", "he will", tweet)
    tweet = re.sub(r"Y'all", "You all", tweet)
    tweet = re.sub(r"Weren't", "Were not", tweet)
    tweet = re.sub(r"Didn't", "Did not", tweet)
    tweet = re.sub(r"they'll", "they will", tweet)
    tweet = re.sub(r"they'd", "they would", tweet)
    tweet = re.sub(r"DON'T", "DO NOT", tweet)
    tweet = re.sub(r"That\x89Ûªs", "That is", tweet)
    tweet = re.sub(r"they've", "they have", tweet)
    tweet = re.sub(r"i'd", "I would", tweet)
    tweet = re.sub(r"should've", "should have", tweet)
    tweet = re.sub(r"You\x89Ûªre", "You are", tweet)
    tweet = re.sub(r"where's", "where is", tweet)
    tweet = re.sub(r"Don\x89Ûªt", "Do not", tweet)
    tweet = re.sub(r"we'd", "we would", tweet)
    tweet = re.sub(r"i'll", "I will", tweet)
    tweet = re.sub(r"weren't", "were not", tweet)
    tweet = re.sub(r"They're", "They are", tweet)
    tweet = re.sub(r"Can\x89Ûªt", "Cannot", tweet)
    tweet = re.sub(r"you\x89Ûªll", "you will", tweet)
    tweet = re.sub(r"I\x89Ûªd", "I would", tweet)
    tweet = re.sub(r"let's", "let us", tweet)
    tweet = re.sub(r"it's", "it is", tweet)
    tweet = re.sub(r"can't", "cannot", tweet)
```

```python
    tweet = re.sub(r"don't", "do not", tweet)
    tweet = re.sub(r"you're", "you are", tweet)
    tweet = re.sub(r"i've", "I have", tweet)
    tweet = re.sub(r"that's", "that is", tweet)
    tweet = re.sub(r"i'll", "I will", tweet)
    tweet = re.sub(r"doesn't", "does not", tweet)
    tweet = re.sub(r"i'd", "I would", tweet)
    tweet = re.sub(r"didn't", "did not", tweet)
    tweet = re.sub(r"ain't", "am not", tweet)
    tweet = re.sub(r"you'll", "you will", tweet)
    tweet = re.sub(r"I've", "I have", tweet)
    tweet = re.sub(r"Don't", "do not", tweet)
    tweet = re.sub(r"I'll", "I will", tweet)
    tweet = re.sub(r"I'd", "I would", tweet)
    tweet = re.sub(r"Let's", "Let us", tweet)
    tweet = re.sub(r"you'd", "You would", tweet)
    tweet = re.sub(r"It's", "It is", tweet)
    tweet = re.sub(r"Ain't", "am not", tweet)
    tweet = re.sub(r"Haven't", "Have not", tweet)
    tweet = re.sub(r"Could've", "Could have", tweet)
    tweet = re.sub(r"youve", "you have", tweet)
    tweet = re.sub(r"donå«t", "do not", tweet)

    tweet = re.sub(r"some1", "someone", tweet)
    tweet = re.sub(r"yrs", "years", tweet)
    tweet = re.sub(r"hrs", "hours", tweet)
    tweet = re.sub(r"2morow|2moro", "tomorrow", tweet)
    tweet = re.sub(r"2day", "today", tweet)
    tweet = re.sub(r"4got|4gotten", "forget", tweet)
    tweet = re.sub(r"b-day|bday", "b-day", tweet)
    tweet = re.sub(r"mother's", "mother", tweet)
    tweet = re.sub(r"mom's", "mom", tweet)
    tweet = re.sub(r"dad's", "dad", tweet)
    tweet = re.sub(r"hahah|hahaha|hahahaha", "haha", tweet)
    tweet = re.sub(r"lmao|lolz|rofl", "lol", tweet)
    tweet = re.sub(r"thanx|thnx", "thanks", tweet)
    tweet = re.sub(r"goood", "good", tweet)
    tweet = re.sub(r"some1", "someone", tweet)
    tweet = re.sub(r"some1", "someone", tweet)
    tweet = tweet.lower()
    tweet=tweet[0:]
    # Removing all URls
    tweet = re.sub(urlPattern,'',tweet)
    # Removing all @username.
    tweet = re.sub(userPattern,'', tweet)
    #remove some words
    tweet= re.sub(some,'',tweet)
    #Remove punctuations
    tweet = tweet.translate(str.maketrans("","",string.punctuation))
    #tokenizing words
    tokens = word_tokenize(tweet)
    #tokens = [w for w in tokens if len(w)>2]
    #Removing Stop Words
    final_tokens = [w for w in tokens if w not in stopword]
    #reducing a word to its word stem
    wordLemm = WordNetLemmatizer()
    finalwords=[]
    for w in final_tokens:
      if len(w)>1:
        word = wordLemm.lemmatize(w)
        finalwords.append(word)
    return ' '.join(finalwords)
```

[18]:
```
abbreviations = {
    "$" : " dollar ",
    "€" : " euro ",
    "4ao" : "for adults only",
    "a.m" : "before midday",
    "a3" : "anytime anywhere anyplace",
    "aamof" : "as a matter of fact",
    "acct" : "account",
    "adih" : "another day in hell",
    "afaic" : "as far as i am concerned",
    "afaict" : "as far as i can tell",
    "afaik" : "as far as i know",
    "afair" : "as far as i remember",
    "afk" : "away from keyboard",
    "app" : "application",
    "approx" : "approximately",
    "apps" : "applications",
    "asap" : "as soon as possible",
    "asl" : "age, sex, location",
    "atk" : "at the keyboard",
    "ave." : "avenue",
    "aymm" : "are you my mother",
    "ayor" : "at your own risk",
    "b&b" : "bed and breakfast",
    "b+b" : "bed and breakfast",
    "b.c" : "before christ",
    "b2b" : "business to business",
    "b2c" : "business to customer",
    "b4" : "before",
    "b4n" : "bye for now",
    "b@u" : "back at you",
    "bae" : "before anyone else",
    "bak" : "back at keyboard",
    "bbbg" : "bye bye be good",
    "bbc" : "british broadcasting corporation",
    "bbias" : "be back in a second",
    "bbl" : "be back later",
    "bbs" : "be back soon",
    "be4" : "before",
    "bfn" : "bye for now",
    "blvd" : "boulevard",
    "bout" : "about",
    "brb" : "be right back",
    "bros" : "brothers",
    "brt" : "be right there",
    "bsaaw" : "big smile and a wink",
    "btw" : "by the way",
    "bwl" : "bursting with laughter",
    "c/o" : "care of",
    "cet" : "central european time",
    "cf" : "compare",
    "cia" : "central intelligence agency",
    "csl" : "can not stop laughing",
    "cu" : "see you",
    "cul8r" : "see you later",
    "cv" : "curriculum vitae",
    "cwot" : "complete waste of time",
    "cya" : "see you",
    "cyt" : "see you tomorrow",
    "dae" : "does anyone else",
    "dbmib" : "do not bother me i am busy",
    "diy" : "do it yourself",
    "dm" : "direct message",
    "dwh" : "during work hours",
    "e123" : "easy as one two three",
    "eet" : "eastern european time",
    "eg" : "example",
    "embm" : "early morning business meeting",
    "encl" : "enclosed",
    "encl." : "enclosed",
    "etc" : "and so on",
    "faq" : "frequently asked questions",
    "fawc" : "for anyone who cares",
    "fb" : "facebook",
    "fc" : "fingers crossed",
    "fig" : "figure",
    "fimh" : "forever in my heart",
    "ft." : "feet",
    "ft" : "featuring",
    "ftl" : "for the loss",
    "ftw" : "for the win",
    "fwiw" : "for what it is worth",
    "fyi" : "for your information",
    "g9" : "genius",
    "gahoy" : "get a hold of yourself",
    "gal" : "get a life",
    "gcse" : "general certificate of secondary education",
    "gfn" : "gone for now",
    "gg" : "good game",
    "gl" : "good luck",
    "glhf" : "good luck have fun",
    "gmt" : "greenwich mean time",
    "gmta" : "great minds think alike",
    "gn" : "good night",
    "g.o.a.t" : "greatest of all time",
    "goat" : "greatest of all time",
    "goi" : "get over it",
    "gps" : "global positioning system",
    "gr8" : "great",
    "gratz" : "congratulations",
    "gyal" : "girl",
    "h&c" : "hot and cold",
    "hp" : "horsepower",
    "hr" : "hour",
    "hrh" : "his royal highness",
    "ht" : "height",
    "ibrb" : "i will be right back",
    "ic" : "i see",
    "icq" : "i seek you",
    "icymi" : "in case you missed it",
    "idc" : "i do not care",
    "idgadf" : "i do not give a damn fuck",
    "idgaf" : "i do not give a fuck",
    "idk" : "i do not know",
    "ie" : "that is",
    "i.e" : "that is",
    "ifyp" : "i feel your pain",
    "IG" : "instagram",
    "iirc" : "if i remember correctly",
    "ilu" : "i love you",
    "ily" : "i love you",
    "imho" : "in my humble opinion",
    "imo" : "in my opinion",
    "imu" : "i miss you",
    "iow" : "in other words",
    "irl" : "in real life",
    "j4f" : "just for fun",
    "jic" : "just in case",
    "jk" : "just kidding",
    "jsyk" : "just so you know",
    "l8r" : "later",
    "lb" : "pound",
    "lbs" : "pounds",
    "ldr" : "long distance relationship",
    "lmao" : "laugh my ass off",
    "lmfao" : "laugh my fucking ass off",
    "lol" : "laughing out loud",
    "ltd" : "limited",
    "ltns" : "long time no see",
    "m8" : "mate",
    "mf" : "motherfucker",
    "mfs" : "motherfuckers",
    "mfw" : "my face when",
    "mofo" : "motherfucker",
    "mph" : "miles per hour",
    "mr" : "mister",
    "mrw" : "my reaction when",
    "ms" : "miss",
    "mte" : "my thoughts exactly",
    "nagi" : "not a good idea",
    "nbc" : "national broadcasting company",
    "nbd" : "not big deal",
    "nfs" : "not for sale",
    "ngl" : "not going to lie",
    "nhs" : "national health service",
    "nrn" : "no reply necessary",
    "nsfl" : "not safe for life",
    "nsfw" : "not safe for work",
    "nth" : "nice to have",
    "nvr" : "never",
    "nyc" : "new york city",
    "oc" : "original content",
    "og" : "original",
    "ohp" : "overhead projector",
    "oic" : "oh i see",
    "omdb" : "over my dead body",
    "omg" : "oh my god",
    "omw" : "on my way",
    "p.a" : "per annum",
    "p.m" : "after midday",
    "pm" : "prime minister",
    "poc" : "people of color",
    "pov" : "point of view",
    "pp" : "pages",
    "ppl" : "people",
    "prw" : "parents are watching",
    "ps" : "postscript",
    "pt" : "point",
    "ptb" : "please text back",
    "pto" : "please turn over",
    "qpsa" : "what happens",
    "ratchet" : "rude",
    "rbtl" : "read between the lines",
    "rlrt" : "real life retweet",
    "rofl" : "rolling on the floor laughing",
    "roflol" : "rolling on the floor laughing out loud",
    "rotflmao" : "rolling on the floor laughing my ass off",
    "rt" : "retweet",
    "ruok" : "are you ok",
    "sfw" : "safe for work",
     "sk8" : "skate",
    "smh" : "shake my head",
    "sq" : "square",
    "srsly" : "seriously",
    "ssdd" : "same stuff different day",
    "tbh" : "to be honest",
    "tbs" : "tablespooful",
    "tbsp" : "tablespooful",
    "tfw" : "that feeling when",
    "thks" : "thank you",
    "tho" : "though",
    "thx" : "thank you",
    "tia" : "thanks in advance",
    "til" : "today i learned",
    "tl;dr" : "too long i did not read",
    "tldr" : "too long i did not read",
    "tmb" : "tweet me back",
    "tntl" : "trying not to laugh",
    "ttyl" : "talk to you later",
    "u" : "you",
    "u2" : "you too",
    "u4e" : "yours for ever",
    "utc" : "coordinated universal time",
    "w/" : "with",
    "w/o" : "without",
    "w8" : "wait",
    "wassup" : "what is up",
    "wb" : "welcome back",
    "wtf" : "what the fuck",
    "wtg" : "way to go",
    "wtpa" : "where the party at",
    "wuf" : "where are you from",
    "wuzup" : "what is up",
    "wywh" : "wish you were here",
    "yd" : "yard",
    "ygtr" : "you got that right",
    "ynk" : "you never know",
    "zzz" : "sleeping bored and tired"
}
```

[19]:
```
def convert_abbrev_in_text(tweet):
    t=[]
    words=tweet.split()
    t = [abbreviations[w.lower()] if w.lower() in abbreviations.keys() else w␣
 ↪for w in words]
    return ' '.join(t)
```

Text processing completed

[20]:
```
tweets['processed_tweets'] = tweets['tweet'].apply(lambda x: process_tweets(x))
tweets['processed_tweets'] = tweets['processed_tweets'].apply(lambda x:␣
 ↪convert_abbrev_in_text(x))
print('Text Preprocessing complete.')
tweets
```

```
Text Preprocessing complete.
```

[20]:
```
         sentiment                                              tweet  \
400068           0  I officially hate JB haters!!!The brothers you…
949261           1  omg gonna get new shirts from the US!  green d…
969256           1  @jordancrockett1 just the verse I was reading …
504087           0  @kellbell68 damn I thought you were talking to…
1543485          1               @spacebetween41 ha same to you dear
…              …                                                …
893186           1    @LeAnnBu lol nahh but wen i do i kan handle it
805761           1  i got the twilight 2-disc special edition yest…
1210674          1  Pinkpopdag 2 was vette shit: Volbeat, Placebo …
862490           1  Thank you to everyone who has been following m…
730945           0                I just ate so much Mac and cheese!

                                          processed_tweets
400068   officially hate jb hatersthe brother youtube a…
949261   oh my god gon na get new shirt you green day w…
969256                         verse reading asked text
504087         damn thought talking noticed offspring song
1543485                                            ha dear
…                                                    …
```

```
893186                 laughing out loud nahh wen kan handle
805761   got twilight 2disc special edition yesturday w…
1210674  pinkpopdag vette shit volbeat placebo amp youm…
862490   thank everyone following really enjoyed tweeti…
730945                                ate much mac cheese

[200000 rows x 3 columns]
```

[21]:
```
#removing shortwords
tweets['processed_tweets']=tweets['processed_tweets'].apply(lambda x: " ".
 ↪join([w for w in x.split() if len(w)>3]))
tweets.head(5)
```

[21]:
```
         sentiment                                             tweet  \
400068           0  I officially hate JB haters!!!The brothers you…
949261           1  omg gonna get new shirts from the US!  green d…
969256           1  @jordancrockett1 just the verse I was reading …
504087           0  @kellbell68 damn I thought you were talking to…
1543485          1               @spacebetween41 ha same to you dear

                                          processed_tweets
400068   officially hate hatersthe brother youtube acco…
949261                              shirt green watchman
969256                            verse reading asked text
504087         damn thought talking noticed offspring song
1543485                                                dear
```

Now lets split the data into a training set and a test set using scikit-learns train_test_split function https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[22]:
```
tweets_data = tweets["processed_tweets"]
tweets_labels = tweets["sentiment"]

#Split data into train_tweets, test_tweets, train_labels and test_labels
train_tweets, test_tweets, train_labels, test_labels =␣
 ↪train_test_split(tweets_data,
                                                                             ␣
 ↪tweets_labels,
                                                                             ␣
 ↪test_size=0.3,
                                                                             ␣
 ↪random_state=10)
```

What we need to build our classifier is “probability of positive tweet” P(pos) , “probability of negative tweet” P(neg), “probability of word in tweet given tweet is positive” P(w|pos) and “probability of word in tweet given tweet is negative” P(w|neg). Start by calculating the probability that a tweet is positive and negative respectively

[23]:
```
P_pos = len(train_labels[train_labels.to_frame()["sentiment"]==1]) /
 ↪len(train_labels)
P_neg = len(train_labels[train_labels.to_frame()["sentiment"]==0]) /
 ↪len(train_labels)
print(P_pos,P_neg)
```

```
0.5002857142857143 0.4997142857142857
```

For P(w|pos), P(w|neg) we need to count how many tweets each word occur in. Count the number of tweets each word occurs in and store in the word counter. An entry in the word counter is for instance {'good': 'Pos':150, 'Neg': 10} meaning good occurs in 150 positive tweets and 10 negative tweets. Be aware that we are not interested in calculating multiple occurances of the same word in the same tweet. Also we change the labels from 0 for "Negative" and 1 for "Positive" to "Neg" and "Pos" respectively.For each word convert it to lower case. You can use Python's lower. Another handy Python string method is split.

[24]:
```
new_train_labels = train_labels.replace(0, "Neg", regex=True)
final_train_labels = new_train_labels.replace(1, "Pos", regex=True)
word_counter = {}
for (tweet, label) in zip(train_tweets, final_train_labels):
# ... Count number of tweets each word occurs in and store in word_counter
# where an entry looks like ex. {'word': 'Pos':98, 'Neg':10}
  lower=tweet.lower()
  word_arr=set(lower.split())
  for word in word_arr:
    if word not in word_counter:
      if label=="Pos":
        word_counter[word]={"Pos":1,"Neg":0}
      if label=="Neg":
        word_counter[word]={"Pos":0,"Neg":1}
    else:
      if label=="Pos":
        word_counter[word]["Pos"]+=1
      if label=="Neg":
        word_counter[word]["Neg"]+=1
```

Let's work with a smaller subset of words just to save up some time. Find the 1500 most occuring words in tweet data.

[25]:
```
nr_of_words_to_use = 1500
popular_words = sorted(word_counter.items(), key=lambda x: x[1]['Pos'] +␣
 ↪x[1]['Neg'], reverse=True)
popular_words = [x[0] for x in popular_words[:nr_of_words_to_use]]
```

[26]:
```
print(popular_words)
```

```
['good', 'like', 'today', 'love', 'time', 'work', 'going', 'laughing', 'loud',
'back', 'know', 'really', 'want', 'still', 'think', 'well', 'thanks', 'night',
'need', 'home', 'would', 'miss', 'much', 'feel', 'last', 'make', 'tomorrow',
'hope', 'morning', 'great', 'though', 'haha', 'twitter', 'wish', 'right',
'come', 'week', 'sleep', 'happy', 'thing', 'could', 'sorry', 'friend',
'tonight', 'getting', 'better', 'people', 'watching', 'look', 'wait', 'nice',
'hour', 'yeah', 'thank', 'hate', 'take', 'school', 'next', 'weekend', 'even',
'soon', 'never', 'cant', 'show', 'dont', 'awesome', 'life', 'long', 'tweet',
'working', 'first', 'little', 'year', 'movie', 'feeling', 'sick', 'best',
'everyone', 'watch', 'tired', 'girl', 'already', 'find', 'always', 'sure',
'something', 'done', 'please', 'suck', 'ready', 'phone', 'another', 'keep',
'house', 'cool', 'looking', 'made', 'went', 'hurt', 'trying', 'pretty',
'thought', 'song', 'help', 'ever', 'start', 'finally', 'someone', 'away',
'maybe', 'lost', 'left', 'sound', 'summer', 'guess', 'mean', 'baby', 'amazing',
'damn', 'early', 'missed', 'nothing', 'game', 'tell', 'follow', 'hear',
'actually', 'bored', 'glad', 'later', 'birthday', 'coming', 'weather', 'rain',
'live', 'play', 'also', 'exam', 'head', 'might', 'said', 'stuff', 'excited',
'sunday', 'call', 'thats', 'party', 'talk', 'waiting', 'world', 'yesterday',
'hard', 'give', 'found', 'since', 'luck', 'monday', 'friday', 'cold',
'beautiful', 'around', 'many', 'stop', 'music', 'making', 'follower', 'late',
'must', 'till', 'read', 'gone', 'video', 'check', 'anything', 'almost', 'leave',
'anyone', 'shit', 'iphone', 'missing', 'finished', 'lunch', 'food', 'dinner',
'cute', 'listening', 'least', 'month', 'sweet', 'funny', 'without', 'woke',
'book', 'place', 'okay', 'family', 'poor', 'forward', 'enjoy', 'mine', 'class',
'free', 'facebook', 'picture', 'enough', 'thinking', 'welcome', 'believe',
'hair', 'cause', 'wanted', 'fuck', 'playing', 'didnt', 'everything', 'idea',
'probably', 'totally', 'every', 'stay', 'wrong', 'stupid', 'real', 'money',
'update', 'anymore', 'name', 'whole', 'eating', 'outside', 'busy', 'headache',
'coffee', 'room', 'following', 'brother', 'message', 'saturday', 'sooo', 'post',
'came', 'beach', 'dream', 'rest', 'lovely', 'crazy', 'learned', 'goodnight',
'else', 'seen', 'minute', 'hopefully', 'face', 'kinda', 'took', 'hell',
'taking', 'forgot', 'able', 'leaving', 'plan', 'send', 'final', 'city',
'problem', 'word', 'hello', 'seems', 'super', 'true', 'meet', 'shopping',
'mother', 'break', 'used', 'part', 'wont', 'started', 'office', 'tried',
'computer', 'full', 'awww', 'hahaa', 'heart', 'trip', 'rock', 'sitting',
'either', 'drink', 'raining', 'using', 'course', 'site', 'blog', 'half', 'mind',
'news', 'email', 'talking', 'stuck', 'sister', 'dude', 'remember', 'care',
'quite', 'change', 'pain', 'alone', 'internet', 'seriously', 'link', 'seeing',
'ticket', 'kind', 'heard', 'laugh', 'reply', 'concert', 'hehe', 'photo',
'reading', 'watched', 'told', 'favorite', 'awake', 'online', 'called',
'breakfast', 'instead', 'text', 'hand', 'fine', 'afternoon', 'fucking',
'person', 'anyway', 'open', 'bought', 'loved', 'broke', 'walk', 'boring',
'wake', 'starting', 'asleep', 'hungry', 'sunny', 'june', 'train', 'study',
'sleeping', 'season', 'turn', 'second', 'heading', 'drive', 'lucky', 'jealous',
'story', 'dear', 'hoping', 'definitely', 'enjoying', 'laptop', 'shower',
'couple', 'bring', 'test', 'goin', 'move', 'homework', 'award', 'reason',
'happened', 'mood', 'about', 'wonderful', 'soooo', 'point', 'lady', 'sore',
'finish', 'fail', 'congrats', 'album', 'bday', 'worry', 'crap', 'running',
'moment', 'fall', 'write', 'dead', 'church', 'holiday', 'together', 'episode',
'short', 'foot', 'smile', 'catch', 'youtube', 'store', 'perfect', 'hang',
'list', 'doesnt', 'agree', 'sometimes', 'visit', 'writing', 'three', 'star',
'meeting', 'evening', 'date', 'direct', 'close', 'seem', 'studying', 'listen',
'town', 'high', 'page', 'clean', 'weird', 'pool', 'line', 'died', 'dance',
'interesting', 'chance', 'math', 'happen', 'ipod', 'side', 'worth', 'loving',
'tour', 'english', 'knew', 'sigh', 'account', 'ride', 'throat', 'supposed',
'drinking', 'followfriday', 'cream', 'past', 'chocolate', 'wonder',
'unfortunately', 'college', 'upset', 'body', 'window', 'fast', 'water',
'driving', 'tuesday', 'band', 'mister', 'gave', 'comment', 'worst', 'tweeting',
'broken', 'website', 'moving', 'saying', 'easy', 'team', 'forget', 'sadly',
'whats', 'sent', 'green', 'fell', 'london', 'paper', 'special', 'nite', 'wear',
'wishing', 'pick', 'earlier', 'wedding', 'york', 'answer', 'sunshine', 'worse',
'parent', 'park', 'jonas', 'understand', 'spend', 'cleaning', 'beer', 'shot',
'rather', 'card', 'hanging', 'black', 'apparently', 'join', 'question', 'moon',
'flight', 'lazy', 'horrible', 'camera', 'project', 'support', 'blue', 'dress',
'vote', 'sleepy', 'cake', 'scared', 'forever', 'white', 'vacation', 'finger',
'thursday', 'woman', 'nope', 'possible', 'shop', 'application', 'kidding',
'fair', 'longer', 'boyfriend', 'power', 'beat', 'spent', 'sign', 'slow',
'number', 'shoe', 'shame', 'kill', 'garden', 'wondering', 'hubby', 'warm',
'cheer', 'havent', 'hows', 'chat', 'decided', 'cousin', 'inside', 'meant',
'officially', 'plus', 'what', 'looked', 'proud', 'slept', 'rainy', 'bike',
'felt', 'load', 'issue', 'july', 'different', 'myspace', 'chicken', 'huge',
'voice', 'road', 'babe', 'stomach', 'learn', 'fact', 'hospital', 'figure',
'tummy', 'lonely', 'absolutely', 'case', 'note', 'especially', 'yall',
'graduation', 'company', 'order', 'save', 'light', 'mate', 'worked', 'share',
'shall', 'liked', 'killing', 'apple', 'glass', 'service', 'ahhh', 'shirt',
'pizza', 'miley', 'yummy', 'behind', 'safe', 'packing', 'ouch', 'airport',
'gorgeous', 'front', 'google', 'guitar', 'hill', 'interview', 'club', 'group',
'bitch', 'daughter', 'turned', 'goodbye', 'download', 'exactly', 'small',
'except', 'david', 'radio', 'taken', 'film', 'alright', 'realized', 'wife',
'freaking', 'french', 'door', 'drunk', 'exciting', 'father', 'waking',
'although', 'changed', 'random', 'wine', 'shoot', 'vega', 'doctor', 'wednesday',
'sims', 'worried', 'lose', 'smell', 'needed', 'isnt', 'husband', 'others',
'business', 'passed', 'terrible', 'hotel', 'bummed', 'tear', 'mommy', 'sale',
'xoxo', 'screen', 'state', 'deal', 'living', 'ball', 'ahead', 'played',
'twilight', 'dying', 'peace', 'staying', 'mile', 'along', 'dark', 'scary',
'singing', 'fantastic', 'gettin', 'hold', 'happens', 'giving', 'drop', 'sort',
'nearly', 'ache', 'usually', 'posted', 'walking', 'storm', 'touch', 'traffic',
'sold', 'alot', 'couldnt', 'profile', 'cook', 'lame', 'relaxing', 'aint',
'puppy', 'near', 'somewhere', 'nobody', 'lately', 'joke', 'watchin', 'child',
'type', 'headed', 'plane', 'itunes', 'country', 'currently', 'annoying',
'indeed', 'everybody', 'memory', 'appreciate', 'upload', 'round', 'awful',
'normal', 'dang', 'mention', 'taste', 'record', 'blood', 'asked', 'buddy',
'chillin', 'bird', 'completely', 'event', 'enjoyed', 'pissed', 'pink', 'lakers',
'sing', 'clothes', 'taylor', 'trouble', 'single', 'hmmm', 'kick', 'kiss',
'version', 'sooooo', 'self', 'stopped', 'disappointed', 'double', 'doubt',
'silly', 'argh', 'swimming', 'caught', 'hilarious', 'heat', 'none', 'matter',
'wearing', 'track', 'essay', 'hangover', 'fixed', 'sexy', 'hahaha', 'dunno',
'tweetdeck', 'cover', 'history', 'whatever', 'serious', 'falling', 'extra',
'confused', 'suppose', 'prime', 'failed', 'closed', 'folk', 'wasnt', 'often',
'choice', 'blackberry', 'info', 'mall', 'anywhere', 'nose', 'ended', 'peep',
'starbucks', 'training', 'gosh', 'cheese', 'hannah', 'awwww', 'feelin',
'laying', 'bloody', 'revision', 'result', 'camp', 'chris', 'mess', 'future',
'practice', 'design', 'teeth', 'there', 'congratulation', 'dropped', 'nick',
'minister', 'board', 'honey', 'angel', 'burnt', 'anyways', 'planning', 'street',
'death', 'experience', 'trailer', 'bless', 'quoti', 'quot', 'lesson',
'exhausted', 'middle', 'south', 'pack', 'piece', 'bummer', 'badly', 'view',
'knee', 'background', 'putting', 'area', 'shake', 'young', 'milk', 'speak',
'stand', 'usual', 'quick', 'demi', 'paid', 'bank', 'chill', 'spending',
'learning', 'quiet', 'magic', 'ring', 'america', 'fire', 'tweeps', 'roll',
'mark', 'search', 'pray', 'shut', 'outta', 'machine', 'contact', 'shift',
'shes', 'brain', 'freakin', 'complete', 'doin', 'wall', 'sense', 'gotten',
'major', 'checking', 'race', 'gutted', 'john', 'happening', 'nail', 'daddy',
'darn', 'fever', 'australia', 'added', 'before', 'sending', 'tree', 'killed',
'kitty', 'unless', 'adam', 'mobile', 'dancing', 'bunch', 'calling', 'keeping',
'assignment', 'laundry', 'mouth', 'moved', 'checked', 'straight', 'trek',
'midnight', 'present', 'epic', 'shout', 'prob', 'showing', 'article', 'begin',
'burn', 'fresh', 'asking', 'quotthe', 'holy', 'sell', 'twit', 'mail', 'sweetie',
'system', 'available', 'island', 'just', 'american', 'stick', 'bill', 'telling',
'cancelled', 'promise', 'seat', 'depressed', 'count', 'channel', 'everyday',
'france', 'twittering', 'arrived', 'teacher', 'science', 'wrote', 'become',
'brought', 'bite', 'blast', 'rough', 'however', 'review', 'grandma', 'finding',
'depressing', 'degree', 'catching', 'spring', 'return', 'price', 'spot',
'drama', 'battery', 'twice', 'cooky', 'favourite', 'personal', 'fight',
'followed', 'gift', 'mama', 'deserve', 'bottle', 'loss', 'space', 'series',
'gunna', 'orange', 'nervous', 'surgery', 'noticed', 'prayer', 'invite',
'buying', 'germany', 'cell', 'energy', 'four', 'thru', 'floor', 'cried',
'release', 'west', 'apart', 'youre', 'kept', 'dentist', 'imagine', 'august',
'player', 'girlfriend', 'waste', 'updated', 'neck', 'million', 'library',
'client', 'keyboard', 'fish', 'delicious', 'craving', 'file', 'letting',
'social', 'realize', 'bear', 'productive', 'talked', 'hurting', 'entire',
'rule', 'pleasure', 'apartment', 'throw', 'brazil', 'style', 'winter',
'wanting', 'perhaps', 'studio', 'sometime', 'important', 'block', 'match',
'annoyed', 'festival', 'grad', 'somebody', 'session', 'color', 'relax',
'surprise', 'tough', 'sharing', 'shining', 'official', 'matt', 'report',
'retweet', 'secret', 'positive', 'helping', 'five', 'option', 'travel', 'leaf',
'picked', 'across', 'code', 'lake', 'swear', 'cooking', 'king', 'xbox', 'given',
'connection', 'expensive', 'chip', 'possibly', 'kate', 'flower', 'simple',
'copy', 'squarespace', 'ugly', 'prom', 'animal', 'flying', 'losing', 'station',
'marathon', 'adorable', 'allowed', 'alive', 'topic', 'oops', 'stage', 'mcfly',
'stress', 'mornin', 'lord', 'boat', 'afraid', 'afford', 'tonite', 'cloud',
'weight', 'crash', 'offer', 'canada', 'revising', 'england', 'ohhh', 'tweeter',
'setting', 'chicago', 'bread', 'form', 'hehehe', 'student', 'messed', 'woohoo',
'button', 'strange', 'truly', 'image', 'ending', 'finishing', 'nasty', 'ahaha',
'server', 'workout', 'voted', 'empty', 'finale', 'lookin', 'medium', 'healthy',
'everyones', 'step', 'feed', 'skin', 'market', 'fault', 'size', 'allergy',
'jack', 'sushi', '2009', 'fave', 'soup', 'schedule', 'choose', 'angry', 'sport',
'wind', 'desk', 'butt', 'lack', 'blah', 'local', 'celebrate', 'barely',
'awhile', 'original', 'driver', 'hero', 'addicted', 'oooh', 'crappy',
'national', 'paris', 'stayed', 'contest', 'brilliant', 'biggest', 'older',
'walked', 'cough', 'pant', 'montana', 'easier', 'fake', 'decide', 'control',
'beginning', 'mmmm', 'cable', 'everywhere', 'boston', 'married', 'florida',
'ahhhh', 'nightmare', 'notice', 'cost', 'wash', 'score', 'texas', 'bright',
'quote', 'ampamp', 'credit', 'posting', 'difficult', 'meal', 'north',
'literally', 'swim', 'suggestion', 'trust', 'managed', 'tune', 'wave',
'character', 'chick', 'wing', 'couch', 'signed', 'coast', 'daily', 'click',
'handle', 'breaking', 'sandwich', 'grow', 'goodmorning', 'planned', 'flat',
'blocked', 'strong', 'strawberry', 'swine', 'arent', 'diet', 'chinese', 'sarah',
'chilling', 'kitchen', 'woot', 'honest', 'mini', 'extremely', 'smart',
'goodness', 'conversation', 'excellent', 'painting', 'german', 'attention',
'lovin', 'spam', 'jump', 'bath', 'crossed', 'british', 'blessed', 'level',
'human', 'reminds', 'latest', 'gross', 'james', 'earth', 'aunt', 'third',
'grrr', 'cupcake', 'tennis', 'shitty', 'transformer', 'attack', 'alex',
'conference', 'thunder', 'dammit', 'figured', 'juice', 'yard', 'mistake',
'paint', 'pulled', 'blow', 'clear', 'error', 'scene', 'smiling', 'user',
'dressed', 'disney', 'general', 'football', 'bacon', 'anytime', 'dumb', 'jean',
'lived', 'grade', 'applications', 'land', 'twitpic', 'youll', 'everytime',
'turning', 'heavy', 'evil', 'ordered', 'treat', 'spanish', 'hahaaha', 'table',
'blame', 'content', 'graduate', 'reality', 'conan', 'interested', 'quality',
'nooo', 'feature', 'access', 'tattoo', 'member', 'fuckin', 'baseball', 'andy',
'hearing', 'classic', 'letter', 'ughh', 'joined', 'ruined', 'boot', 'software',
'detail', 'sugar', '2morrow', 'onto', 'talent', 'killer', 'idol', 'fabulous',
'pair', 'information', 'teach', 'grand', 'expect', 'hella', 'uncle', 'imma',
'cheap', 'salad', 'macbook', 'status', 'mike', 'excuse', 'prefer', 'main',
'remembered', 'shoulder', 'limit', 'mostly', 'crashed', 'adventure', 'truck',
'wide', 'cali', 'liking', 'tryin', 'fear', 'bunny', 'deleted', 'counting',
'soccer', 'senior', 'freak', 'diversity', 'ryan', 'slightly', 'comp', 'total',
'born', 'stressed', 'properly', 'kevin', 'massive', 'brand', 'updating',
'neighbor', 'praying', 'ankle', 'dollar', 'queen', 'chuck', 'model', 'workin',
'somehow', 'quit', 'michael', 'havin', 'pound', 'term', 'texting', 'rent',
'hurry', 'tooth', 'clue', 'otherwise', 'xxxx', 'painful', 'tweeted', 'washing',
'deep', 'sunburn', 'danny', 'brown', 'hole', 'cash', 'expected', 'threw',
'cavs', 'thunderstorm', 'network', 'upgrade', 'ruin', 'cleaned', 'likely',
'mode', 'realised', 'swift', 'california', 'grocery', 'visiting', 'acting',
'performance', 'youu', 'tiny', 'health', 'kicked', 'broadcasting', 'toast',
'rite', 'chair', 'bathroom', 'anybody', 'released', 'wooo', 'candy', 'pancake',
'picking', 'product', 'bang', 'situation', 'cuddle', 'alarm', 'wouldnt', 'golf',
'jersey', 'charge', 'decision', 'surprised', 'hardly', 'testing', 'soul',
'closer', 'tech', '2nite', 'knowing', 'nephew', 'preparing', 'decent', 'reach',
'continue', 'speed', 'crack', 'lying', 'advice', 'chapter', 'heck', 'growing',
'drag', 'dave', 'cloudy', 'whenever', 'agreed', 'lead', 'opening', 'crew',
'gear', 'wild', 'lyric', 'response', 'freezing', 'rice', 'public', 'effect',
'downtown', 'current', 'invited', 'susan', 'doll', 'artist', 'exercise',
'burger']
```

Now lets compute P(w|pos), P(w|neg) for the popular words

[29]:
```
P_w_given_pos = {}
P_w_given_neg = {}
for word in popular_words:
    # Calculate the two probabilities
    total_Pos=word_counter[word]["Pos"]
    total_Neg=word_counter[word]["Neg"]
    total=total_Pos + total_Neg
    Prob_word_given_pos=((total_Pos/total)*(total/len(train_tweets))) /P_pos
    P_w_given_pos[word]=Prob_word_given_pos
    Prob_word_given_neg=((total_Neg/total)*(total/len(train_tweets))) /P_neg
    P_w_given_neg[word]=Prob_word_given_neg
```

[31]:
```
classifier = {
    'basis'   : popular_words,
    'P(pos)'    : P_pos,
    'P(neg)'    : P_neg,
    'P(w|pos)' : P_w_given_pos,
    'P(w|neg)' : P_w_given_neg
    }
```

Train and predict Write a tweet_classifier function that takes your trained classifier and a tweet and returns wether it's about Positive or Negative using the popular words selected. Note that if there are words in the basis words in our classifier that are not in the tweet we have the opposite probabilities i.e P(w_1 occurs)* P(w_2 does not occur) * …. if w_1 occurs and w_2 does not occur. The function should return wether the tweet is Positive or Negative. i.e 'Pos' or 'Neg'.

[33]:
```
def tweet_classifier(tweet, classifier_dict):
# ... Code for classifying tweets using the naive bayes classifier
  popular_words = classifier_dict['basis']
  P_pos = classifier_dict['P(pos)']
  P_neg = classifier_dict['P(neg)']
  P_w_given_pos = classifier_dict['P(w|pos)']
  P_w_given_neg = classifier_dict['P(w|neg)']

  prob_pos = P_pos
  prob_neg = P_neg
  tweet=tweet.split()
  for word in popular_words:
      if word in tweet:
          prob_pos *= P_w_given_pos[word]
          prob_neg *= P_w_given_neg[word]
      else:
          prob_pos *= 1 - P_w_given_pos[word]
          prob_neg *= 1 - P_w_given_neg[word]

  if prob_pos > prob_neg:
      return 'Pos'
  else:
      return 'Neg'
```

[34]:
```
def test_classifier(classifier, test_tweets, test_labels):
    total = len(test_tweets)
    correct = 0
    for (tweet,label) in zip(test_tweets, test_labels):
        predicted = tweet_classifier(tweet,classifier)
        if predicted == label:
            correct = correct + 1
    return(correct/total)
```

[35]:
```
new_test_labels = test_labels.replace(0, "Neg", regex=True)
final_test_labels = new_test_labels.replace(1, "Pos", regex=True)
```

[36]:
```
acc = test_classifier(classifier, test_tweets, final_test_labels)
print(f"Accuracy: {acc:.4f}")
```

```
Accuracy: 0.7246
```

Optional work In basic sentiment analysis classifications we have 3 classes "Positive", "Negative" and "Neutral". Although because it is challenging to create the "Neutral" class. Try to improve the accuracy by filtering the dataset from the perspective of removing words that indicate neutrality.

[ ]:
```

```