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

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

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:

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

or

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

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

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

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

In [2]:
```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

In [3]:
```python
tweets=pd.read_csv('twitter_sentiment_analysis.csv',encoding='latin',
                    names = ['sentiment','id','date','query','user','tweet'])
tweets
```

Out[3]:

	sentiment	id	date	query	user	tweet
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
...	...	...	...	...	...	...
1599995	4	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best fee...
1599996	4	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interv...
1599997	4	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me f...
1599998	4	2193602064	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	tinydiamondz	Happy 38th Birthday to my boo of alll time!!! ...

	sentiment	id	date	query	user	tweet
1599999	4	2193602129	Tue Jun 16 08:40:50 PDT 2009	NO_QUERY	RyanTrevMorris	happy #charitytuesday @theNSPCC @SparksCharity...

1600000 rows × 6 columns

```
In [4]: tweets = tweets.sample(frac=1)
        tweets = tweets[:200000]
        print("Dataset shape:", tweets.shape)
```

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

```
In [5]: tweets['sentiment'].unique()
```

```
Out[5]: array([4, 0], dtype=int64)
```

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

```
In [6]: tweets['sentiment']=tweets['sentiment'].replace(4,1)
        tweets
```

```
Out[6]:
```

	sentiment	id	date	query	user	tweet
1426777	1	2059282302	Sat Jun 06 16:37:01 PDT 2009	NO_QUERY	Marrriia	I've had @thereadyset in my head all dayyyy
1229836	1	1991441622	Mon Jun 01 07:02:43 PDT 2009	NO_QUERY	CanadianJennie	@wickedcanadagal heheh thanks! oh well i figu...
1458845	1	2063701877	Sun Jun 07 03:44:21 PDT 2009	NO_QUERY	drinkFUZE	@ the Suntrust Sunday Jazz Brunch in Riverwalk...
287379	0	1994201579	Mon Jun 01 11:34:12 PDT 2009	NO_QUERY	mojomoonjo	@ficar22 Live 2 far way from Carlsbad now..No ...
1325617	1	2015160109	Wed Jun 03 03:53:01 PDT 2009	NO_QUERY	kzluvskim23	@littlemisskim23 had fun too dont be sorry fo...
...	...	...	...	...	...	...
1045571	1	1957564583	Fri May 29 00:53:58 PDT 2009	NO_QUERY	anoopmenon	@iamazad yes bot no more OD-ing
1062532	1	1964171663	Fri May 29 13:49:58 PDT 2009	NO_QUERY	MissDoze	@Pdotwee Yeah, u can say that. Surprisingly, ...
64048	0	1687897368	Sun May 03 09:51:34 PDT 2009	NO_QUERY	Olive228	***sighs** @ Paris, early may, still wearing a...
882539	1	1685983121	Sun May 03 03:07:27 PDT 2009	NO_QUERY	hollow_af	eating an oaty backed bar, mmm
638663	0	2234517748	Thu Jun 18 23:12:24 PDT 2009	NO_QUERY	Bear5562	shit have to get to the basement now. err

200000 rows × 6 columns

Removing the unnecessary columns.

```
In [7]: tweets.drop(['date','query','user'], axis=1, inplace=True)
        tweets.drop('id', axis=1, inplace=True)
        tweets.head(10)
```

Out[7]:

	sentiment	tweet
1426777	1	I've had @thereadyset in my head all dayyyy
1229836	1	@wickedcanadagal heheh thanks! oh well i figu...
1458845	1	@ the Suntrust Sunday Jazz Brunch in Riverwalk...
287379	0	@ficar22 Live 2 far way from Carlsbad now..No ...
1325617	1	@littlemisskim23 had fun too dont be sorry fo...
1161145	1	Good morning Twitter - Watching Scrubs with m...
1000485	1	hey all you night owls
653492	0	Fml I think I just got toothpaste in my eye!
1295982	1	Sat watching jeremy kyle at da mo people! Sort...
258815	0	@souljaboytellem http://twitpic.com/6ccu7 - i ...

Checking if any null values present

In [8]:
```
(tweets.isnull().sum() / len(tweets))*100
```

Out[8]:
```
sentiment    0.0
tweet        0.0
dtype: float64
```

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

In [9]:
```
#converting pandas object to a string type
tweets['tweet'] = tweets['tweet'].astype('str')
```

Check the number of positive vs. negative tagged sentences

In [10]:
```
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:  100187
No. of negative tagged sentences is: 99813
```

In [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
```

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('wordnet')
stopword = set(stopwords.words('english'))
print(stopword)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\albin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\albin\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\albin\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\albin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

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

Data Cleaning

In [12]:
```
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)
    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[1:]
    # 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)
```

In [13]:
```
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"
}
```

In [14]:
```
def convert_abbrev_in_text(tweet):
    t=[]
    words=tweet.split()
    t = [abbreviations[w.lower()] if w.lower() in abbreviations.keys() else w for w
    return ' '.join(t)
```

Text processing completed

In [15]:
```
tweets['processed_tweets'] = tweets['tweet'].apply(lambda x: process_tweets(x))
tweets['processed_tweets'] = tweets['processed_tweets'].apply(lambda x: convert_abb
print('Text Preprocessing complete.')
tweets
```

```
Text Preprocessing complete.
```

```
Out[15]:
```

	sentiment	tweet	processed_tweets
1426777	1	I've had @thereadyset in my head all dayyyy	head dayyyy
1229836	1	@wickedcanadagal heheh thanks! oh well i figu...	wickedcanadagal heheh thanks oh well figure im...
1458845	1	@ the Suntrust Sunday Jazz Brunch in Riverwalk...	suntrust sunday jazz brunch riverwalk featurin...
287379	0	@ficar22 Live 2 far way from Carlsbad now..No ...	ficar22 live far way carlsbad nowno tulip enjo...
1325617	1	@littlemisskim23 had fun too dont be sorry fo...	littlemisskim23 fun dont sorry cooking spaghet...
...	...	...	...
1045571	1	@iamazad yes bot no more OD-ing	iamazad yes bot oding
1062532	1	@Pdotwee Yeah, u can say that. Surprisingly, ...	pdotwee yeah say surprisingly woke kinda early...
64048	0	***sighs** @ Paris, early may, still wearing a...	sigh paris early may still wearing jacket
882539	1	eating an oaty backed bar, mmm	ating oaty backed bar mmm
638663	0	shit have to get to the basement now. err	hit get basement err

200000 rows × 3 columns

```
In [16]: #removing shortwords
         tweets['processed_tweets']=tweets['processed_tweets'].apply(lambda x: " ".join([w f
         tweets.head(5)
```

```
Out[16]:
```

	sentiment	tweet	processed_tweets
1426777	1	I've had @thereadyset in my head all dayyyy	head dayyyy
1229836	1	@wickedcanadagal heheh thanks! oh well i figu...	wickedcanadagal heheh thanks well figure going...
1458845	1	@ the Suntrust Sunday Jazz Brunch in Riverwalk...	suntrust sunday jazz brunch riverwalk featurin...
287379	0	@ficar22 Live 2 far way from Carlsbad now..No ...	ficar22 live carlsbad nowno tulip enjoying ros...
1325617	1	@littlemisskim23 had fun too dont be sorry fo...	littlemisskim23 dont sorry cooking spaghetti s...

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

In [17]:
```
tweets_data = tweets["processed_tweets"]
tweets_labels = tweets["sentiment"]

#Split data into train_tweets, test_tweets, train_labels and test_labels
seed = 42
train_tweets, test_tweets, train_labels, test_labels = train_test_split(tweets_data
```

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

In [18]:
```
neg, pos = train_labels.value_counts()
total = train_labels.value_counts().sum()

P_pos = pos / total
P_neg = 1 - P_pos

assert P_pos + P_neg == 1
```

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.

In [19]:
```
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
    words = [word.lower() for word in set(tweet.split())]
    for word in words:
        if word not in word_counter:
            word_counter[word] = {'Pos' : 0 , 'Neg' : 0}
        word_counter[word][label]+=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.

In [20]:
```
nr_of_words_to_use = 1500
popular_words = sorted(word_counter.items(), key=lambda x: x[1]['Pos'] + x[1]['Neg'
popular_words = [x[0] for x in popular_words[:nr_of_words_to_use]]
```

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

In [23]:
```
P_w_given_pos = {}
P_w_given_neg = {}
```

```
pos = 0
neg = 0
# Count number of labels for popular words
for word in popular_words:
    pos += word_counter[word]['Pos']
    neg += word_counter[word]['Neg']

for word in popular_words:
    # Calculate the two probabilities
    P_w_given_pos[word]=word_counter[word]['Pos']/pos
    P_w_given_neg[word]=word_counter[word]['Neg']/neg
```

In [24]:
```
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
    }
```

In [25]:
```
new_test_labels = test_labels.replace(0, "Neg", regex=True)
final_test_labels = new_test_labels.replace(1, "Pos", regex=True)
```

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'.

In [26]:
```
def tweet_classifier(tweet, classifier_dict):
    """ param tweet: string containing tweet message
        param classifier: dict containing 'basis' - training words
                                          'P(pos)' - class probabilities
                                          'P(neg)' - class probabilities
                                          'P(w|pos)' - conditional probabilities
                                          'P(w|neg)' - conditional probabilities

        return: either 'Pos' or 'Neg'
    """
# ... Code for classifying tweets using the naive bayes classifier

    P_pos_given_w = classifier_dict['P(pos)']
    P_neg_given_w = classifier_dict['P(neg)']

    tweet_words = set(tweet.split())
    for word in classifier_dict['basis']:
        if word in tweet_words:
            P_pos_given_w *= classifier_dict['P(w|pos)'][word]
            P_neg_given_w *= classifier_dict['P(w|neg)'][word]
        else:
            P_pos_given_w *= 1-classifier_dict['P(w|pos)'][word]
            P_neg_given_w *= 1-classifier_dict['P(w|neg)'][word]

    return 'Pos' if P_pos_given_w > P_neg_given_w else 'Neg'
```

In [27]:
```
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)
```

In [28]:
```
acc = test_classifier(classifier, test_tweets, final_test_labels)
print(f"Accuracy: {acc:.4f}")
```

```
Accuracy: 0.7137
```

Optional work

In basic sentiment analysis classifications we have 3 classes "Positive", "Negative" and "Neutral". Try to improve your classifiers accuracy by including the "Neutral" class.

In [ ]: