Twitter Sentimental Analysis

Final Report

Name: Muaz Maqbool
Roll no: 15-4053
Submitted to: Mubashir Baig

Abstract

Gathering and analyzing data from social media is difficult these days. It is one among the biggest challenge of this decade to analyze and perform textual analysis. As twitter gains more popularity in social media tweets on twitter becomes valuable source of information. An effort is being made to facilitate in this regard using three methods. First is simple lexicon-based method which use **Vader Sentimental Lexicon** to provide negative, positive and neutral sentiment to the sample data. Second and third is a machine Learning technique **Naïve Bayes and Linear Support Vector machine** as classifier and **N-grams** to tokenize the text input. Finding of this research is that both methods provided different accuracy in the test data. **Vader Sentimental Lexicon** resulted in 65% accuracy while **Naïve Bayes** resulted in 76% accuracy and Linear SVM resulted in 79% accuracy.

Why Sentimental Analysis?

From the start of this century socializing online is one among the biggest trend. The first social network startup launched In 1997 with the name of six degrees. It allowed users to create profile and enabled the add friend feature which is most common in Facebook in recent days. After that **Friendster** was founded in 2002 which was the first modern social network. Currently it has over 90 million users along with 60+ million unique visitors each month. Then the social network chain started getting strength with some renown networks like **HI5(2003), LINKEDIN (2003), MYSPACE (2003), Facebook (2006).** Facebook is not the first social network of the world. Some of the amazing examples of social networks are as under

- Twitter is one of the most popular social network website. In 2010, after a 7.0 magnitude earth quick in **Haiti** twitter served as a major hub of information.
- The biggest Question after US election was that by spending the least amount of money and having fewer vote count then Harley **Clinton**, how **Donald Trump** was able to make it through. According to certified resources he sketched his campaign using the information from the highly payed Data Analyst

Problem and Background Work

Given that the social networks are the biggest sources of the information why we are unable to process this data. Having such a huge dataset why people are unable to analyze and find the possible trends and solve the problems.

Problem?
The problem with such huge data is that with its increasing size it's almost impossible to analyze the shifting opinions of the populace.

THE solution…
The obvious solution to this problem is that categorize your data. And device an algorithm that can extract the data of your interest. Like for example if you are interested in a cricket match and want to analyze the opinion of the people about that match you can write an algorithm which can parse through posts and interpret the required information. The need for that is if you want to extract a particular interest of a person the remaining information is useless and can produce a negative impact towards your efficiency. For example, if you are to decide the political interest of a population you won't be interested in what kind of food they like to consume.

Is it possible?
For those who are able to understand the mechanics of devising a digital algorithm, might be aware of the fact that it is nearly impossible for someone to write such piece of code. It is not possible to write such a code using nested if and else or some conditionals to cover all the conditions by the sentimental analysis. So for that we need a complex logic to be formulated. IN that circumstances we need Artificial intelligence to make such decisions.

Role of AI in Social media mining

There has been a fair amount of research on how social media data can be processed. The first difficulty is that the method required to represented before processing. Because, data before processing has to be uniform because machine cannot classify the grammar like human mind does. Some algorithms like vender sentimental lexicon provided a useful information and a good progress in sentimental analysis by labeling the words and assigning them weight. But in this research, we will be working if they prove themselves to be useful in twitter sentimental analysis.

What Is Sentimental Analysis?

Sentimental analysis which is also known as opinion mining or emotion AI refers to use of Natural language processing and text analysis to systematically identify, extract, quantify and study affective states of the data along with subjective information

Sentiment

A sentiment can be defined using Merriam-Webster Dictionary as.

- A prediction or opinion
- An emotion or a refined feeling
- An idea colored by emotion

Anatomy of Sentiment

You can sub divide a sentiment into following parts.

Holding: Someone whose sentiment value is to be measured.

Target: The objective entity on which sentiment is being made.

Polarity: A sentimental can be classified into 2 polarities

Twofold
Two possible classification (Positive or Negative)

Three-Fold
Three possible classification (Positive, Negative or Neutral)

Aspect: it defines the features for which sentiment is expressed

Selected Methods:

Generically sentimental analysis can be done in two ways.

- Machine learning method
- Non-Machine learning method

Non-Machine Learning method:

VADER Sentiment Lexicon

In non-machine method lexicon-based methods are being used. The intuition behind this method is that the sentiment of text is being computed by any number of dominant words in the text. The dominance of the text is calculated simply by using word counting technique.

Simple Word Count

Given a statement lexicon I, a document d= $\{w_1, w_2, \ldots\ldots w_n\}$ where $w_i(1 < I < n)$ represents the ith word in d. Let pos(l,d) denotes the occurrence of positive words in the document and neg(l,d) denotes the occurrence of negative words in the document. Using the positive and negative occurrences in the document the overall sentiment is calculated as

$$neg(l,d) = pos(l,d) – neg(l,d)$$

The sentimental orientation of d can be 1 denoting as positive and -1 as negative can be defined as:

Entry	**Intensity**	**Std.**	**Human Evaluation Vector**
accomplish	1.8	0.6	[1, 2, 3, 2, 2, 2, 1, 1, 2, 2]
dangers	-2.2	0.87178	[-1, -1, -2, -4, -2, -3, -3, -2, -2, -2]
lol	2.0	1.18322	[3, 0, 3, 0, 3, 1, 3, 2, 3, 2]

Fig 1

Using the matrix above VADER lexicon defines the sentiment of a sentence by adding up the combination of polarity and intensity of each sentence.

Machine learning method.

The machine learning methods include 2 algorithms

1- Support Vector Machine
2- Naïve Bayes Rule

Naïve Bayes

The machine learning method which we will be using is Naïve Bayes Rule which use the Bayes rule using probabilistic model. Naïve Bayes Classifier works on a simple assumption that if the class is specified then all of the features are independent of each other. For the implementation of Naïve Bayes a particular representation of data(tweets) is required as classifier needs to be trained. For this project I have used N-gram representation. This method is based on an probabilistic inference rules. Given $x = (x1, x2, \ldots xn)$ be some n features of some test sample, and we want to label it to class with max $P[Ck \mid x1, \ldots, xn]$, the probability of the data being for each possible class Ck . In our case, xi is each N-gram feature and Ck is the target class. Using Bayes' theorem, the conditional probability is rewritten as $P[Ck \mid x1, \ldots, xn] \propto P[Ck]\, P[x1 \mid Ck] \ldots P[xn \mid Ck]$, where $P[Ck]$ is referred as prior and $P[x1 \mid Ck] \ldots P[xn \mid Ck]$ is referred as likelihood. The training part is to find $P[xn \mid Ck]$ on the dataset by counting, and the prediction is by labeling one sample to class y by

$$Y = \operatorname{argmax}_{k \in \{1,\ldots,K\}} (P[Ck]\, P[x1 \mid Ck] \ldots P[xn \mid Ck])$$

Linear SVM

SVM is based on finding the maximum margin between different classes by determining w and b in $w \,.\, x + b = 0$, the hyperplane separating classes. Mathematically, binary-class SVM minimizes

$$1/n(C \sum max(0,1 - y^{(i)} (w \cdot x^{(i)} + b)) + \lambda \, \| w \|^2)$$

where $y^{(i)} = \{-1,1\}$ is the true label, λ is the regularization term, C is the penalty parameter of error term,

and $max(0,1-y^{(i)}(w.x^{(i)} + b))$ is the loss of miss classification, also known as hinge loss. Note that there are many versions of SVM, what we use is a linear SVM with hinge loss due to computational efficiency. The training is done by finding w and b via coordinate descent and the prediction is done be $y = \operatorname{sgn}(w.x = b)$.

N-gram Representation:

If you have heared or studied the term textual analysis you must be aware of the word Bag-of-Words. This is one of the most important as well as common representation in textual analysis. In N-grams representation a vector of token is represented along with their numeric values. To convert a tweet into N-gram representation following methods are needed to be implemented in order to fulfil the task.

How N-grams work:

N grams picks the sentence and make three different divisions from the text.

-unigram
-bigram
-trigrams

Unigrams: unigrams are the single words extracted from the text.

Bigrams: The different possible combination from the text of two words or unigrams can be used to make bigrams

Trigrams: Trigrams is extracted from the set of unigrams keeping the size to 3 words.

Unigrams are then passed for further processing mentioned below

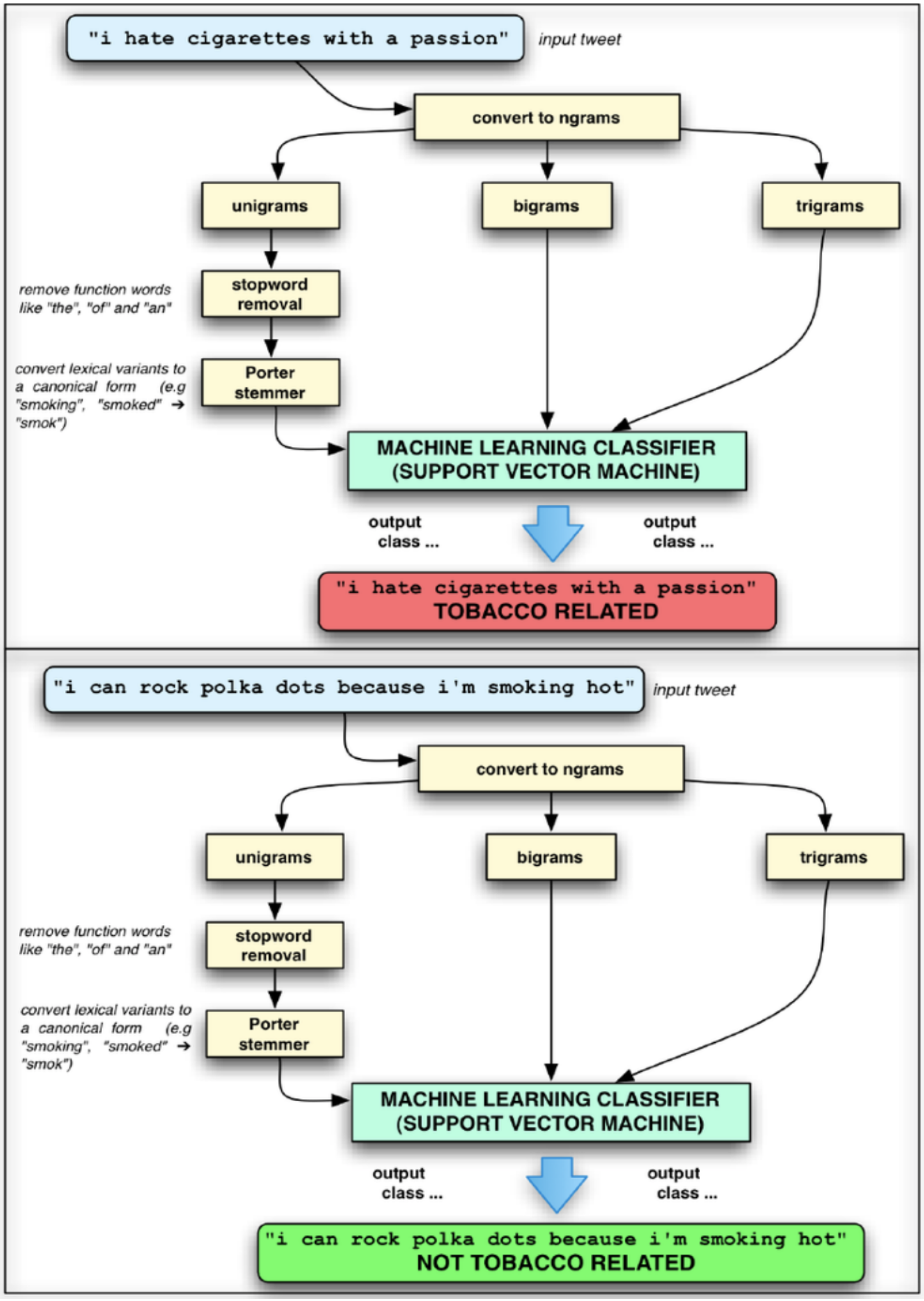
Convert the unigrams to the lower case.

Removal of stop words from the unigrams.

Create integer featured indices.

Convert feature sequences to feature vectors by certain computation.

A flow chart of how N-gram representation helps top perform textual classification is shown below to demonstrate the common use of n-gram representation along with machine learning (Support Vector Machine) algorithm. The example has been quoted from a paper [1]


"i hate cigarettes with a passion"
input tweet
convert to ngrams
unigrams
bigrams
trigrams
remove function words like "the", "of" and "an"
stopword removal
convert lexical variants to a canonical form (e.g "smoking", "smoked" → "smok")
Porter stemmer
MACHINE LEARNING CLASSIFIER (SUPPORT VECTOR MACHINE)
output class ...
output class ...
"i hate cigarettes with a passion" TOBACCO RELATED
"i can rock polka dots because i'm smoking hot"
input tweet
convert to ngrams
unigrams
bigrams
trigrams
remove function words like "the", "of" and "an"
stopword removal
convert lexical variants to a canonical form (e.g "smoking", "smoked" → "smok")
Porter stemmer
MACHINE LEARNING CLASSIFIER (SUPPORT VECTOR MACHINE)
output class ...
output class ...
"i can rock polka dots because i'm smoking hot" NOT TOBACCO RELATED

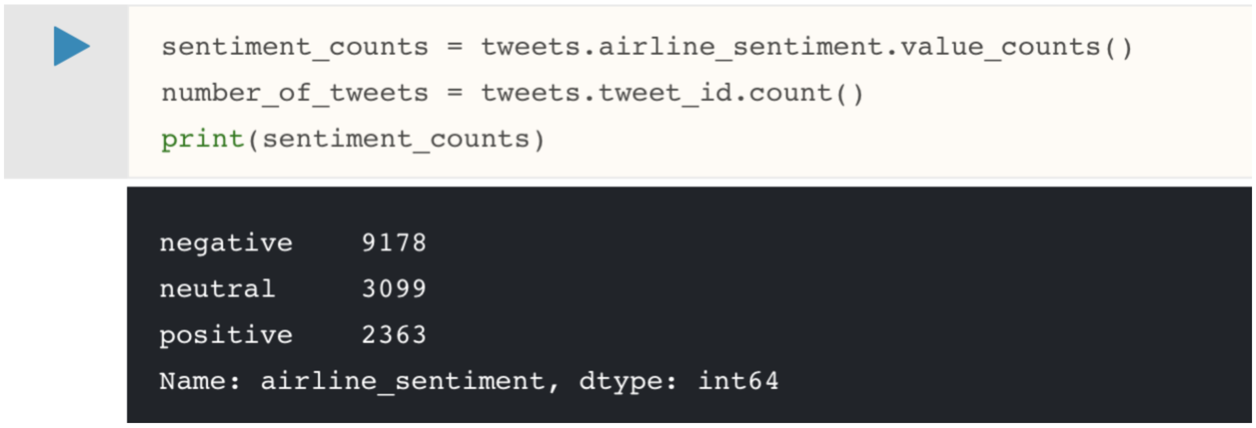
https://www.researchgate.net/publication/256290162_Using_Twitter_to_Examine_Smoking_Behavior_and_Perceptions_of_Emerging_Tobacco_Products

Fig2

Dataset:

For Naïve Bayes algorithm and Linear SVM I have used twitter airline data for Virgin American airline. Data can be accessed with the link listed. This dataset has almost 15000 rows along with time name sentiment value and remarks on each tweet.

Data Visualization:

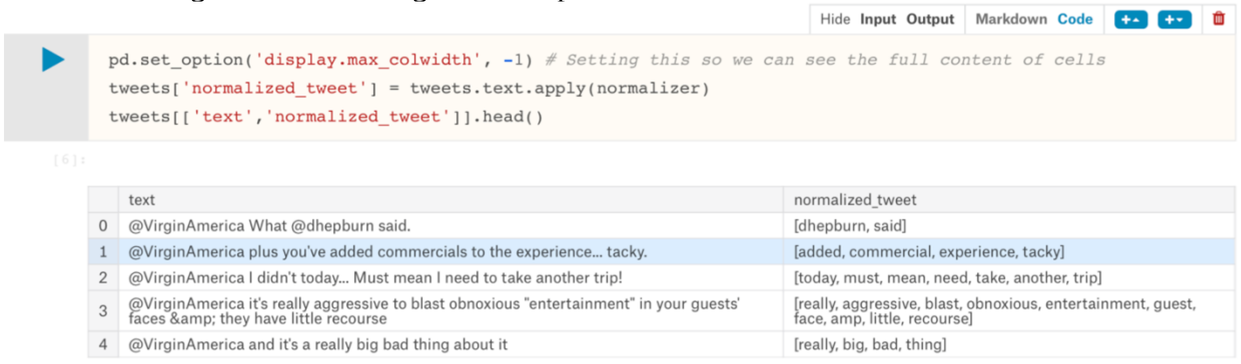


```
sentiment_counts = tweets.airline_sentiment.value_counts()
number_of_tweets = tweets.tweet_id.count()
print(sentiment_counts)
```

```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```

we divided 33% to the test test set and remaining to the training test.

After cleaning and removing the stop words I was able to normalize the tweets



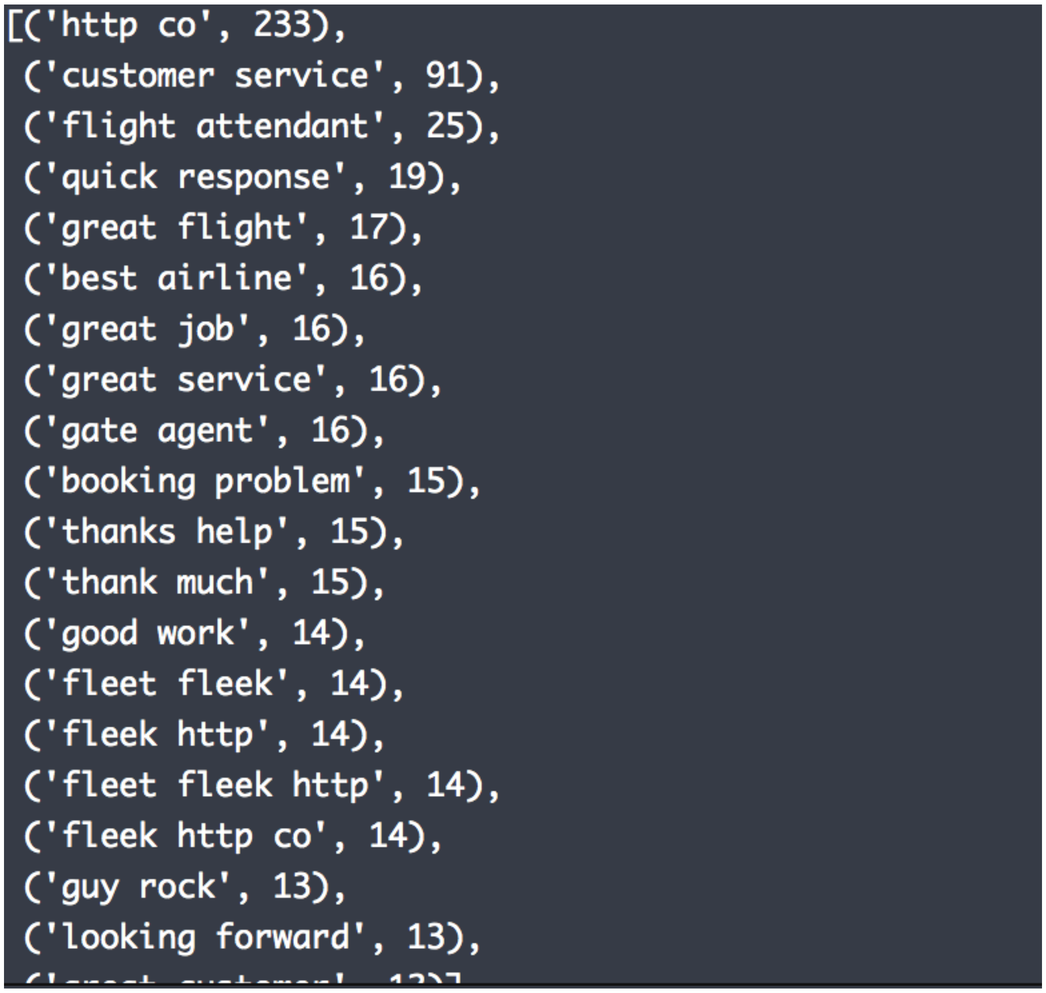
```
pd.set_option('display.max_colwidth', -1) # Setting this so we can see the full content of cells
tweets['normalized_tweet'] = tweets.text.apply(normalizer)
tweets[['text','normalized_tweet']].head()
```

	text	normalized_tweet
0	@VirginAmerica What @dhepburn said.	[dhepburn, said]
1	@VirginAmerica plus you've added commercials to the experience... tacky.	[added, commercial, experience, tacky]
2	@VirginAmerica I didn't today... Must mean I need to take another trip!	[today, must, mean, need, take, another, trip]
3	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces &amp; they have little recourse	[really, aggressive, blast, obnoxious, entertainment, guest, face, amp, little, recourse]
4	@VirginAmerica and it's a really big bad thing about it	[really, big, bad, thing]

Double negations, stop words were removed and words were unified to avoid any kind of mismatch. After normalizing the tweets I extracted the positive and negative bigrams. Which acted as our feature vector for classification in both the negative and positive case.

Positive bi-grams:

Some positive bigrams from the data are listed as under.

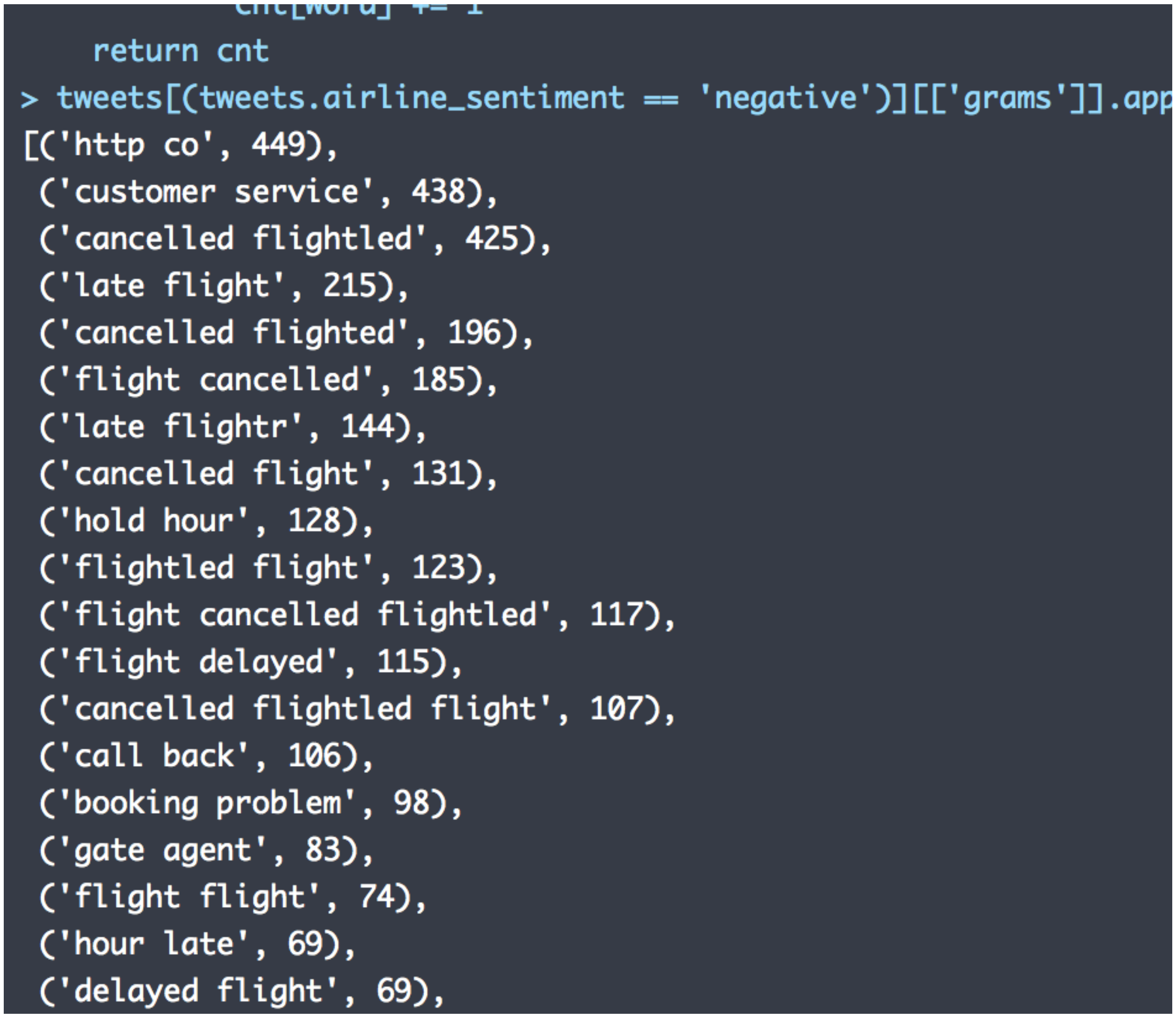


```
[('http co', 233),
 ('customer service', 91),
 ('flight attendant', 25),
 ('quick response', 19),
 ('great flight', 17),
 ('best airline', 16),
 ('great job', 16),
 ('great service', 16),
 ('gate agent', 16),
 ('booking problem', 15),
 ('thanks help', 15),
 ('thank much', 15),
 ('good work', 14),
 ('fleet fleek', 14),
 ('fleek http', 14),
 ('fleet fleek http', 14),
 ('fleek http co', 14),
 ('guy rock', 13),
 ('looking forward', 13),
```

Fig 3

Negative bi-grams:

Some of the dominant negative bigrams are listed as under.



```
            cnt[word] += 1
    return cnt
> tweets[(tweets.airline_sentiment == 'negative')][['grams']].app
[('http co', 449),
 ('customer service', 438),
 ('cancelled flightled', 425),
 ('late flight', 215),
 ('cancelled flighted', 196),
 ('flight cancelled', 185),
 ('late flightr', 144),
 ('cancelled flight', 131),
 ('hold hour', 128),
 ('flightled flight', 123),
 ('flight cancelled flightled', 117),
 ('flight delayed', 115),
 ('cancelled flightled flight', 107),
 ('call back', 106),
 ('booking problem', 98),
 ('gate agent', 83),
 ('flight flight', 74),
 ('hour late', 69),
 ('delayed flight', 69),
```

Fig 4

Results and Analysis

VADER Sentiment Lexicon:

As this algorithm needs no training for the classification of tweets so I have computed it';s accuracy on each tweet present in dataset. Lexicon methods are based on pre-trained word list (also known as Human Evaluation Vector) Since it is a non-machine learning algorithm it provides results in no time. For each tweet it will aggregate the sentiment of every tweet and formulate a paragraph sentiment score by adding up all. If threshold exceeds the average value then tweet will be labeled as positive otherwise negative. On this dataset it provides around 65% accuracy. Reason for lower accuracy can be that this method fails to classify neutral tweets in a better way
Some sample runs on this data set are given below

Running the Vader Sentiment Lexicon on some sample tweets gave following results:

1) **"Hary is smart, handsome, and funny.",**

Output: {'pos': 0.746, 'neu': 0.254, 'neg': 0.0}

2) **"The book was good.",** Output: {'pos': 0.492, 'neu': 0.508, 'neg': 0.0}

3) **"The book was kind of good.",** Output: {'pos': 0.343, 'neu': 0.657, 'neg': 0.0}

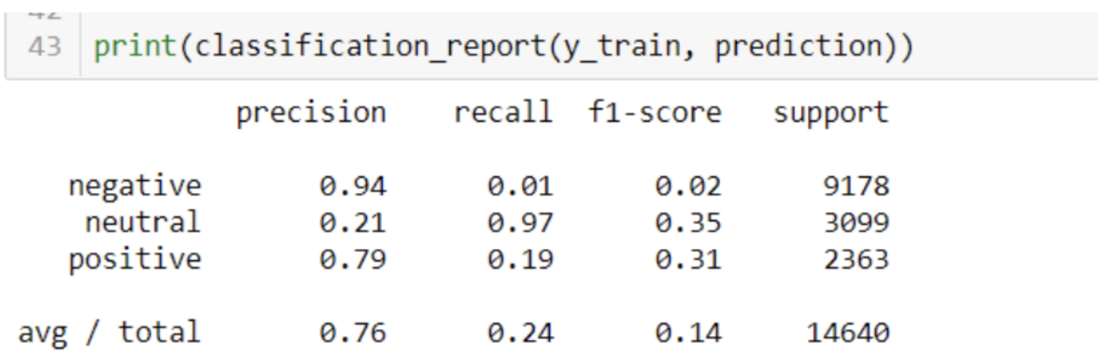
4) **"The plot was good, but the characters are evil and the dialog is not great.",**

Output: {'pos': 0.08, 'neu': 0.49, 'neg': 0.43}

5) **"At least it isn't a horrible book.",** Output: {'pos': 0.363, 'neu': 0.637, 'neg': 0.0}

6) **"Make sure you :) or :D today!",** Output: {'pos': 0.706, 'neu': 0.294, 'neg': 0.0}

Along with that in this dataset some positive reviews also include negative words in shape of feedback/suggestions. It misclassifies the positive tweets even on a lower threshold. Overall accuracy of this method is given by



```
print(classification_report(y_train, prediction))
              precision    recall  f1-score   support

    negative       0.94      0.01      0.02      9178
     neutral       0.21      0.97      0.35      3099
    positive       0.79      0.19      0.31      2363

 avg / total       0.76      0.24      0.14     14640
```

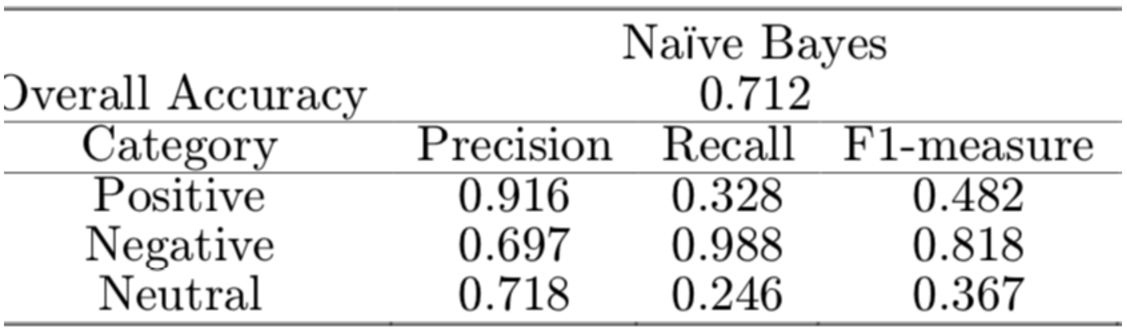
Fig 5

Machine Learning

Naïve Bayes:

I used N-gram features for naïve Bayes method as described above, removing the tokens which are the most frequent ones and changed all letters in tweets to lower case for the sake of uniformity, Naïve Bayes was very fast. Not providing the expected accuracy it provided around 71% accuracy on given dataset. I have used the default implementation of SkiKit python library passing it the data after cleaning. In the shape of Bigrams feature vector.

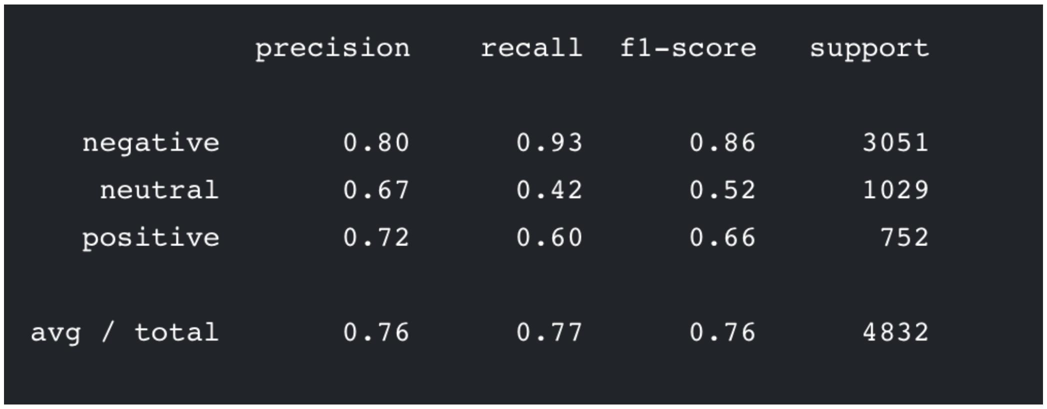
Initially naïve bayes was implemented using Bigrams which resulted in lower accuracy around 71%



	Naïve Bayes		
)verall Accuracy	0.712		
Category	Precision	Recall	F1-measure
Positive	0.916	0.328	0.482
Negative	0.697	0.988	0.818
Neutral	0.718	0.246	0.367

Fig 6

The reason for such unexpected accuracy was that I was only using the bigrams and trigrams to train my probabilistic model. When I only used the unigrams to train the classifier I was able to achieve higher accuracy around 76% percent the new updated matrix for naïve Bayes is


```
             precision    recall  f1-score   support

   negative       0.80      0.93      0.86      3051
    neutral       0.67      0.42      0.52      1029
   positive       0.72      0.60      0.66       752

avg / total       0.76      0.77      0.76      4832
```

naive Bayes took a lot more less time then the

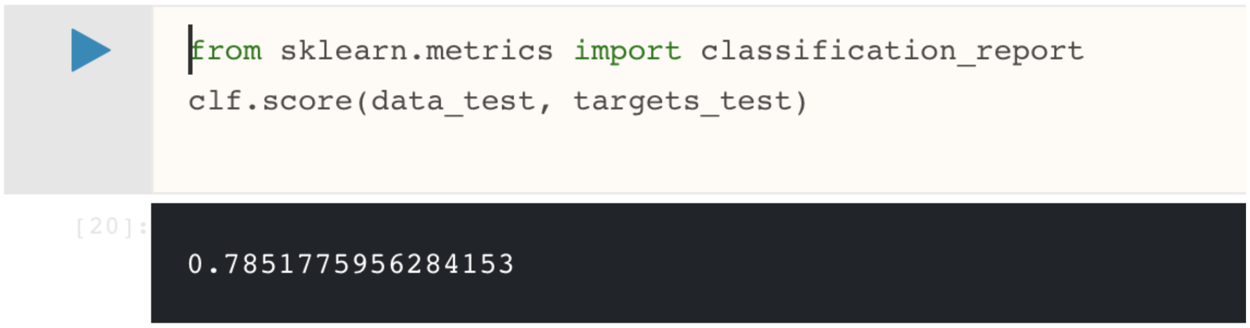
SVM

Initially I assumed that one method for both machine learning and non-machine learning will be enough but after comparing the initial accuracy of Naïve Bayes with bigrams and trigrams the results were not what was expected it only resulted in 71% which was almost nearest to the lexicon sentiment. So, I have to use another machine learning method overlooking the cause of lower frequency of Naïve Bayes. I used Support Vector Machine Algorithm. Linear support vector machine learning algorithm is used for single class classification so the technique I used was ONE Vs ALL classification so that I might be able to implement the classifier.

Implementation

Same cleaning procedure was adopted while implementing Linear SVM. First the data was Extracted from file into tweets and their sentiments. Stop words were removed and after that stemming was applied to the data to gain uniformity. NLTK generic Linear SVM implementation was being used to train the and predict the data.
After cleaning the dataset, ngrams Library of NLTK was used to generate uni, bi and trigrams.
SVM produced better results than the Naïve Bayes on Bigrams and trigrams.

Evaluation of results


```
from sklearn.metrics import classification_report
clf.score(data_test, targets_test)
```

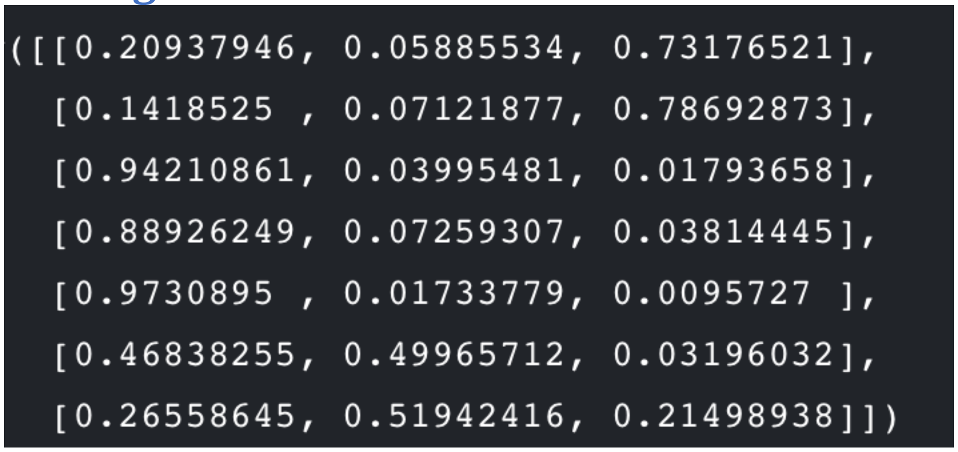
```
[20]:
0.7851775956284153
```

Fig 7

After the evaluation of Linear SVM I tried to check if it is working fine on some obvious sentences. Because, according to research material online it's classification results are around 85%. So, I formulated an array of sentences manually to classify them on my trained SVM classifier.

```
sentences =
   "What a great airline, the trip was a pleasure!",

   "My issue was quickly resolved after calling customer support. Thanks!",

   "What the hell! My flight was cancelled again. This sucks!",

   "Service was awful. I'll never fly with you again.",

   "You fuckers lost my luggage. Never again!",

   "I have mixed feelings about airlines. I don't know what I think.",
```

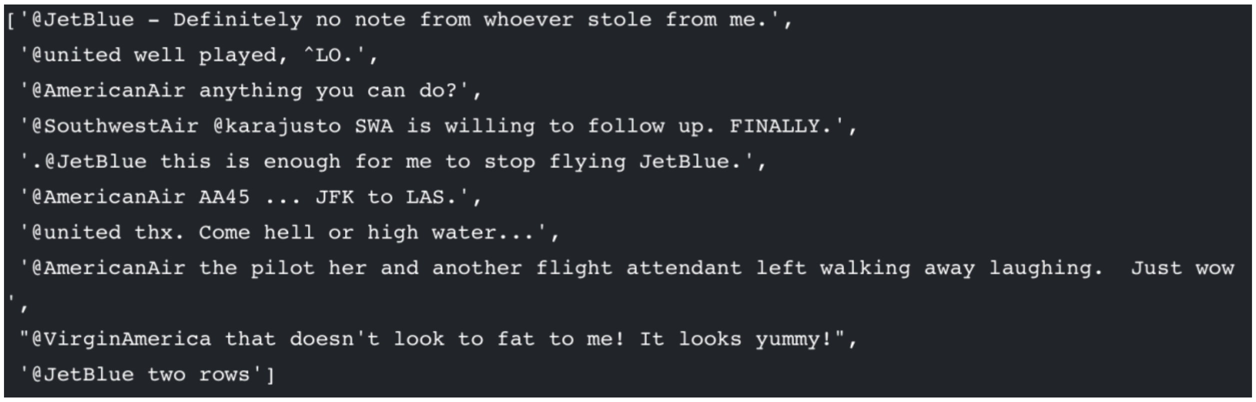
Negative. **Neutral.** **Positive.**



```
([[0.20937946, 0.05885534, 0.73176521],
  [0.1418525 , 0.07121877, 0.78692873],
  [0.94210861, 0.03995481, 0.01793658],
  [0.88926249, 0.07259307, 0.03814445],
  [0.9730895 , 0.01733779, 0.0095727 ],
  [0.46838255, 0.49965712, 0.03196032],
  [0.26558645, 0.51942416, 0.21498938]])
```

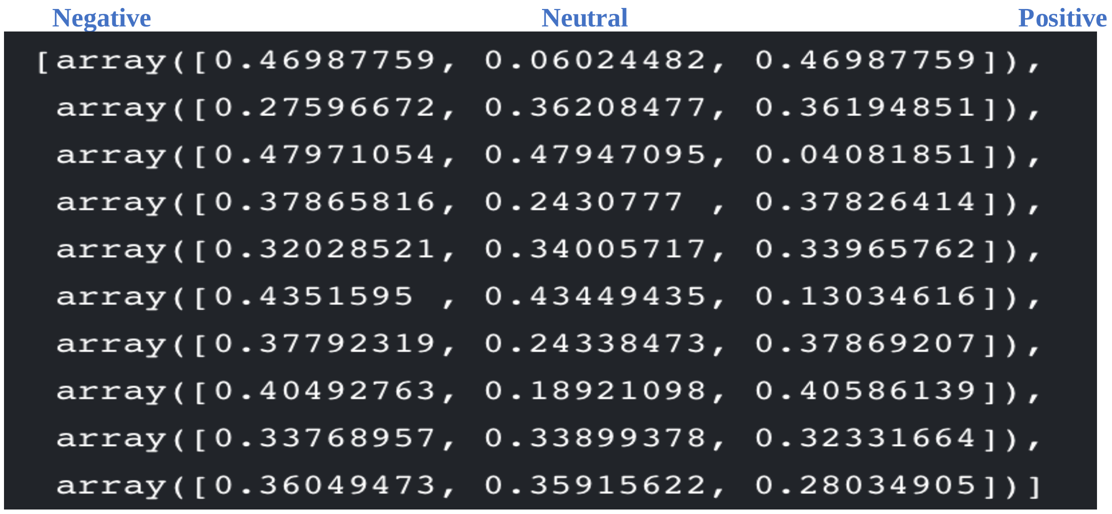
Fig8

Even on 78% accuracy all sentences are being classified in correct order. So, using the prediction matrix I extracted some of the tweets in which the max distance was totally wrong or you can phrase it as the tweets are most difficult to classify. I did this by using max margin function.



```
['@JetBlue - Definitely no note from whoever stole from me.',
 '@united well played, ^LO.',
 '@AmericanAir anything you can do?',
 '@SouthwestAir @karajusto SWA is willing to follow up. FINALLY.',
 '.@JetBlue this is enough for me to stop flying JetBlue.',
 '@AmericanAir AA45 ... JFK to LAS.',
 '@united thx. Come hell or high water...',
 '@AmericanAir the pilot her and another flight attendant left walking away laughing.  Just wow
',
 "@VirginAmerica that doesn't look to fat to me! It looks yummy!",
 '@JetBlue two rows']
```

The prediction of the above tweets done by Linear SVM is listed as under



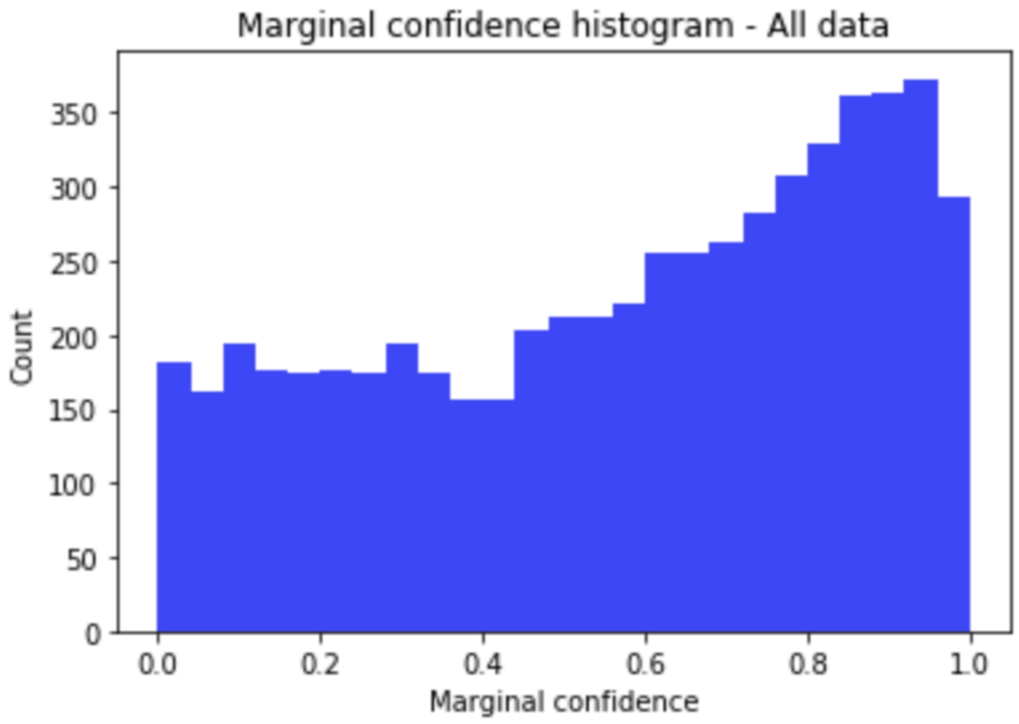
Negative Neutral Positive

```
[array([0.46987759, 0.06024482, 0.46987759]),
 array([0.27596672, 0.36208477, 0.36194851]),
 array([0.47971054, 0.47947095, 0.04081851]),
 array([0.37865816, 0.2430777 , 0.37826414]),
 array([0.32028521, 0.34005717, 0.33965762]),
 array([0.4351595 , 0.43449435, 0.13034616]),
 array([0.37792319, 0.24338473, 0.37869207]),
 array([0.40492763, 0.18921098, 0.40586139]),
 array([0.33768957, 0.33899378, 0.32331664]),
 array([0.36049473, 0.35915622, 0.28034905])]
```

From the above observation it can easily be concluded that most of the tweets that were misclassified with maximum distance are the ones with positive label. So, to visualize this assumption I plotted Marginal Confidence Histogram for the test set predictions. Using Matplotlib I plotted that histogram for all three classes Positive Negative and Neutral.

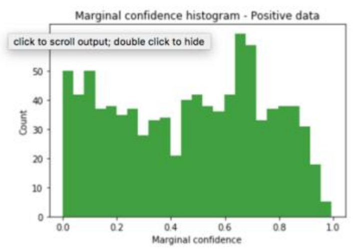
Fist histogram represents the marginal Confidence of the whole test-set

TEST SET

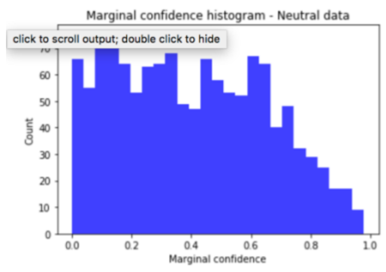


After that I Visualized the Positive Negative and Neutral Histograms to see the classification confidence of my classifier.

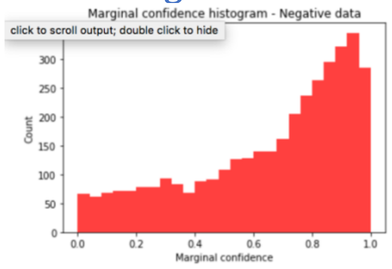
Positive



Neutral



Negative



It is clearly obvious that negative tweets are dominating in terms of classification accuracy. As almost 65% of the data is negative. So Linear SVM is failing to classify the positive data as dataset is skewed towards negative end.

Conclusion

Initially, I was trying to compare a machine learning and a non-machine learning algorithm: VADER Lexicon and Naïve Bayes. But, accuracy provided by naïve Bayes was not exactly expected whereas Vader Lexicon provided the assumed accuracy. So, I applied another machine learning method Linear Support Vector Machine. Which resulted in almost 79% accuracy. Naïve Bayes is more like a baseline method it performs worst as expected, however it runs much faster than Linear SVM, making it useful for large datasets. Another thing, by using hyper parameters in SVM the performance of SVM increased almost by 8%. But while visualizing the predictions of Linear SVM I found my mistake and was able to correct the accuracy of Naïve Bayes. After that, my analysis on this dataset is that this is negatively skewed dataset which makes it more difficult to train a classifier. To prove my assumption, I chopped the data set into 3 equal half's for negative positive and neutral tweets. Which confirms my claim as it provides 93% accuracy for a probabilistic model like naïve Bayes. The issues that occurred the most was regarding the dataset and I have visualized my every single assumption to validate my point

Future Work

In future I consider applying ANN to this dataset to see if it can increase the accuracy of sentimental analysis on this dataset. Moreover, Linear SVM can be applied after chopping this skewed dataset to increase the accuracy.

Reference

1. **Cavnar, William. (1994). Using An N-Gram-Based Document Representation With A Vector Processing Retrieval Model.. 0-.**

2. **Hutto, C.J. & Gilbert, Eric. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.**

3. **Jingcheng Du, Jun Xu, Hsingyi Song, Xiangyu Liu, Cui Tao J Biomed Semantics. 2017; 8: 9. Published online 2017 Mar 3. doi: 10.1186/s13326-017-0120-6**