INTRO TO BIOINFORMATICS CM221

TWITTER SENTIMENT ANALYSIS USING NAIVE BAYES CLASSIFIER

SHIVANI SOMAN UID:304946159

Introduction

Nowadays, people routinely usually publish their thoughts and opinions online on social networking sites like Twitter, Facebook and Instagram. Opinions range from movie reviews to product reviews to views on political campaigns. Consumers also generally look to the opinions of others before deciding on a particular product or service. Popularity of such social media sites is growing exponentially every day. In the first quarter of 2017, Twitter reported more than 328 million monthly active users[1] with more than 500 million tweets sent every day[2]. This provides a great platform for advertisers, political campaigners and marketing professionals from different businesses to infer the sentiments of the users regarding a particular asset or decision. This could prove useful in finding the popularity of a particular product and what people like or don’t like about it.

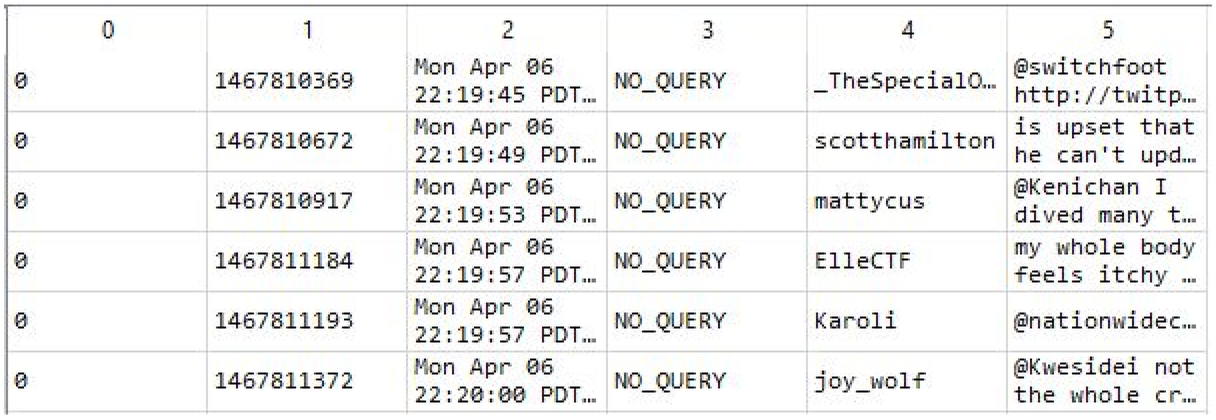
As stated in Wikipedia[3], “Sentiment analysis (sometimes known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.” It is the process of deducing the polarity of a user’s opinion (positive or negative) based on the input provided. Sentiment analysis is widely used and has a variety of applications in various domains like business, politics and finance.

Dataset Description

The dataset used for this project is the Sentiment140 [4] dataset created by Stanford University students. It is a comma separated file (CSV) consisting of 1.6 million English language tweets. The data file format contains 6 fields :

0 - the polarity of the tweet (0 - negative, 4 - positive)
1 - the id of the tweet (2087)
2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009)
3 - the query (lyx). If there is no query, then this value is NO_QUERY.
4 - the user that tweeted (robotickilldozr)
5 - the text of the tweet (Lyx is cool)

A glimpse of the data contained in the the file is provided in Fig. 1.



0	1	2	3	4	5
0	1467810369	Mon Apr 06 22:19:45 PDT…	NO_QUERY	_TheSpecialO…	@switchfoot http://twitp…
0	1467810672	Mon Apr 06 22:19:49 PDT…	NO_QUERY	scotthamilton	is upset that he can't upd…
0	1467810917	Mon Apr 06 22:19:53 PDT…	NO_QUERY	mattycus	@Kenichan I dived many t…
0	1467811184	Mon Apr 06 22:19:57 PDT…	NO_QUERY	ElleCTF	my whole body feels itchy …
0	1467811193	Mon Apr 06 22:19:57 PDT…	NO_QUERY	Karoli	@nationwidec…
0	1467811372	Mon Apr 06 22:20:00 PDT…	NO_QUERY	joy_wolf	@Kwesidei not the whole cr…

Figure 1: First 5 entries in dataset

Framework

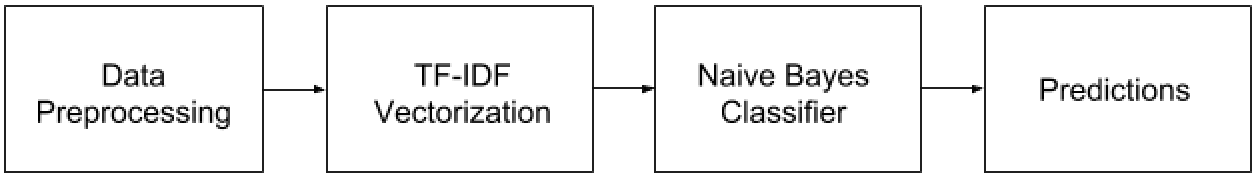


Figure 2: Sentiment Analysis Framework

The framework of this sentiment analysis project is as shown in the Fig 2. It contains four modules as follows:

- Data Preprocessing
- TF-IDF Vectorization
- Naive Bayes Classifier
- Prediction

We will look at each of these modules in detail in the following sections.

Data Preprocessing

In order to analyse the effectiveness of the algorithm, we divide the data into training and testing sets. We consider a ratio of 80:20 i.e. 80% training data (1.28 million tweets) and 20% testing data (320K tweets). The training data is used to train the classifier so that it learns the relations between the features of the tweet and the classified sentiment. The testing set is then used to predict the sentiment of tweets in it.

Several preprocessing tasks need to be carried out to convert the tweet into a form which can be easily processed. The following steps have been carried out:

Convert URLs

Certain tweets contain URLs starting with either “www” or “https”. These URLs have been replaced with a single word “URL” in the tweet as these links do not provide any significant information regarding the sentiment of the tweet and would only interfere with the feature generation process.

Convert Twitter Handle

Most tweets also contain the twitter handle (username) of the user with the format “@username”. This has also been replaced with the token “__USERHANDLE”.

Convert Hashtags

Tweets also contain words preceded by hashtags (“#”). These hashtags are very important in topic modelling and sentiment analysis as they generally contain information regarding what the tweet is about. In this project we replace words containing hashtags simply by removing the hash symbol and keeping the actual word in the tweet for future analysis.

Convert Repeating Letters

Many times people tweet certain words with extra characters like “helllooo” for emphasis. We convert these repeating characters to a maximum of two repeating characters. So, “helllooo” would be replaced with “helloo”.

Convert to Stem

Our TF-IDF Vectorizer considers the number of occurrences of a particular word to infer its importance. Many times we use different forms of the same word like “fishing” and “fished”. In this case, our vectorizer would consider the occurrences of these words separately. Stemming converts the words to their root stem so that words with similar meaning but different forms can be classified as one. In the above example, both “fishing” and “fished” would be converted to the root word “fish”.

Emoticons

We also group emoticons into categories of positive and negative. Positive emoticons like “:)” and “:D” are replaced with the token “__positive__” and negative emoticons like “:(” and “:’(“ are replaced with token “__negative__”.

TF-IDF Vectorization

TF-IDF, short for “term frequency-inverse document frequency” is a statistic that measures a word’s importance in a particular document or text corpus. It consists of three steps -

- Calculating the term frequency
- Calculating the inverse document frequency
- Combining the two metrics to get TF-IDF

Term Frequency

Term frequency (TF) is the measure of how frequently a term occurs in a particular document, or in this case, a particular tweet. But, it’s unlikely that a term that occurs twenty times in a document, is twenty times more important. Hence, normalization is needed to ensure fairness. In this project, we use sublinear TF as a way of normalization. Instead of considering the number of times a word occurs in our tweet, we calculate the term frequency as follows:

$$\mathbf{tf_{(w,d)} = 1 + \log(n_{(w,d)})}$$

where,
$tf_{(w,d)}$ is the weight of the word *w* in document *d*
$n_{(w,d)}$ is the actual frequency of word *w* in document *d*

Inverse Document Frequency

Inverse Document Frequency (IDF) measures the importance of a word in the entire corpus of documents or tweets. Sometimes, insignificant terms like ‘a’ , ‘the’ , etc. have high term frequency and would occur is almost all documents but are of lower importance overall.

Hence, in order to give more importance to rare words, which are generally domain specific and provide more information about the sentiment of the tweet, we calculate the IDF as follows:

$$\mathbf{idf_w = log(N/D_w)}$$

where,
Idf_w is the inverse document frequency of word *w*
N is the total number of documents (tweets) in the corpus
D_w is the number of documents (tweets) that contain word *w*

TF-IDF

Finally we calculate the TF-IDF values of each word by combining the TF and IDF values as follows:

$$\mathbf{tfidf_{(w,d)} = tf_{(w,d)} * idf_w}$$

where,
tfidf(w,d) is the term frequency-inverse document frequency of word *w* in document *d*
tf(w,d) is the weight of word *w* in document *d*
Idfw is the inverse document frequency of word *w* in the entire corpus

We compute the TF-IDF values for all the words in the tweets in the training as well the testing set. The TFIDF Vectorizer outputs a sparse matrix where each row corresponds to the index of particular tweet and each column corresponds to the index of a word in that tweet with that word’s TF-IDF value stored that the specific row and column. We consider unigrams and bigrams of different words in each tweet which provides us with better understanding of the sentiment. The matrix of the training set is given as input to the classifier along with the sentiment classification of each tweet. The matrix of the testing set is then used to predict the sentiment classification.

Naive Bayes Classifier

Naive Bayes is a simple probabilistic classifier which is based on the Bayes Theorem. It assumes that all the observations are independent of each other. In the context of text classification, it assumes that each word is independent of the other words in the sentence. At first, this assumption might seem wrong, but it actually works well for sentiment analysis. It's what makes this model work on small amount of data or mislabeled data. The basic equation of Naive Bayes is given as follows:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

where,
$x = (x_1, x_2, \ldots\ldots x_n)$ is a vector representing n features
C_k represents the k^{th} class
$p(C_k|x)$ is the posterior probability of the class being C_k given observation x
$p(C_k)$ is the prior probability of class C_k
$p(x|C_k)$ is the likelihood of observation x belonging to class C_k
$p(x)$ is the probability of the observations, also known as average likelihood

The prior probability is calculated as the ratio of the number of entries with class C_k over the overall total number of entries. The denominator p(x) is ignored as it would be common for all classes. In this way, the algorithm Naive Bayes calculates the posterior probabilities of of each class $C=(C_1, C_{2, \ldots\ldots} C_m)$ given a feature vector x and classifies x into the class which gives highest probability. In practice, various different distributions of Naive Bayes are used, like Gaussian, Bernoulli or Multinomial. In this project, we use Multinomial Naive Bayes to classify our tweets.

Multinomial Naive Bayes

Multinomial Naive is Bayes implements Naive Bayes (NB) for multinomially distributed data and is classically used in text classification. It normally requires integer feature counts like word frequencies, but in practice, TF-IDF values also work rather well. The likelihood of a word being in a particular class with Multinomial NB is calculated as follows:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where,
N_{yi} is the number of times feature *i* appears in class *y*, which in our case is the sum of the TF-IDF values of word *i* with class *y*
N_y is the total count of all features for class *y*, which in our case is the total sum of all TF-IDF values in class *y*
α is the the smoothing prior where $\alpha \geq 0$ accounts for features not in the training sample and prevents zero probabilities

Predictions

In this way, the likelihoods of the various words in a tweet are calculated. The product of all likelihoods of words in a particular tweet gives us the total likelihood of the tweet in a particular class. From this likelihood value and the previously computed priors, the posterior is calculated which helps us determine the class of the tweet as positive or negative. First, the classifier is fit on the training set which helps it to compute the likelihoods of all the words in the corpus by the two classes. And finally, the testing set is inputted to the classifier which outputs a vector containing the predicted values of sentiment for each tweet in the testing set.

Evaluation

We use various metrics to evaluate our project such as precision, recall, f1 score and support. The computed metrics are provided in the table below.

Class	Precision	Recall	F1 Score	Support
Negative	0.79	0.80	0.80	159789
Positive	0.80	0.79	0.80	160211

Table 1: Computed Metrics

As is shown in the table, our classifier performs pretty well. It also gives an accuracy of 79.83%. The support shows that if all tweets would have grouped into just one class we would get accuracy approximately equal to 50% while our classifier predicts sentiments with a much higher accuracy. The confusion matrix of our predictions is also provided in Fig. 3.

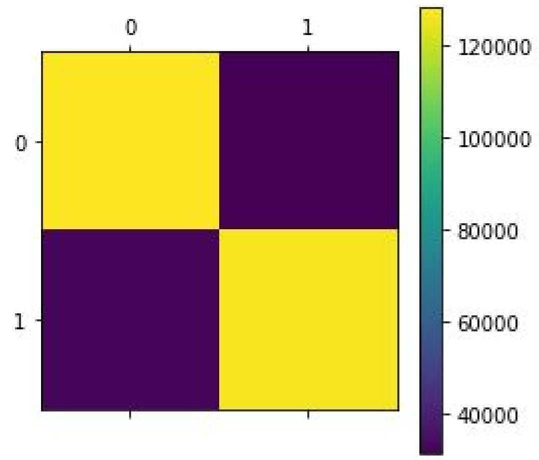


Figure 3: Confusion Matrix

The confusion matrix shows the number of predictions in class 0 i.e. negative and class 1 i.e. positive. The top left cell corresponds to the true negatives, i.e tweets that are negative and were also classified as negative, the top right cell is the false positives, negative tweets classified as positive, bottom left are the false negatives, positive tweets classified as negative and bottom right shows the true positives, the positive tweets correctly classified as positive. As you can see from the matrix, the number of true positives and true negatives are much higher in number as compared to the false positives and false negatives, which shows the success of our classifier.

Conclusion

Our model successfully generates the sentiments of 320k tweets. We have thoroughly pre-processed the data to ensure maximum efficiency in classifying the tweets. We also considered emoticons as context variables in order to increase accuracy of the model. Instead of the regular Naive Bayes classifier, we used the Multinomial Naive Bayes which is more appropriate for the kind of result we require. We also computed the TF-IDF values of all words which gives more emphasis to rare words instead of just considering the count of words in each class. This model gives us a very good accuracy of approximately 80%. A future enhancement could be to use a more complex machine learning model like Support Vector Machine which could provide us with even better results. This project has also been made publicly available on Github at the given link.

Github Link: https://github.com/shivanisoman/Twitter-Sentiment-Analysis

References

[1] https://www.statista.com/topics/737/twitter/

[2] http://www.internetlivestats.com/twitter-statistics/

[3] https://en.wikipedia.org/wiki/Sentiment_analysis

[4] http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip