Twitter Sentiment Analysis

Jimil Desai
Machine Learning (CSE523)
Ahmedabad University
Ahmedabad, India

Khush Kalavadia
Machine Learning (CSE523)
Ahmedabad University
Ahmedabad, India

Tejas Chauhan
Machine Learning (CSE523)
Ahmedabad University
Ahmedabad, India

Hardi Kadia
Machine Learning (CSE523)
Ahmedabad University
Ahmedabad, India

***Abstract*—With the advancement of social media platforms and its growth there is a huge volume of data present in these platform. Internet has brought the world so close together and provided people a means to express themselves. Social Networking sites like Twitter, Facebook, Instagram are gaining popularity among people as they allow users to express their opinions on variety of topics, have discussions and post messages. This gives rise to opportunities to detect and predict sentiments of people on various topics. There has been a lot of work in the field of sentiment analysis of twitter data. This project mainly focuses on sentiment analysis of twitter data, which is helpful to analyze the information in the tweets where the opinions are highly unstructured and heterogeneous. The aim of this project is to come up various machine learning models to accurately detect sentiment of a tweet.**

***Index Terms*—Sentiment Analysis, Twitter, Machine Learning, Logistic Regression, Naive Bayes, TFIDF Vectorizer**

I. Introduction

The invention of the Internet has revolutionised the world. It has changed the way people think, express their views and opinions. Users find it more convenient and satisfying to express their views via blogs, online forums, social media etc. Internet has given people a new voice. Millions of people use social networking apps like Twitter, Facebook, Instagram etc. on a daily basis to express their opinions. Millions of tweets are posted everyday, where users express their views on variety of topics. Social Media generates a huge volume of sentiment rich data in the form of tweets, blogs, posts etc.

This gives us the opportunity of performing various operations on these data to understand sentiments of the user. Detecting sentiments can help businesses to improve their marketing tactics and advertisements. On the other hand it can also help detect depressions and human emotions. Applications are many.

In this paper, we take a look at twitter data and build models for classifying tweets into positive or negative sentiments. Understanding textual data is a significant task and for that we leverage various NLP (Natural Language Processing) Libraries to learn the vocabulary. After various pre-processing functions we have applied the Multinomial Naive Bayes Classifier and Logistic Regression models to predict the sentiment of a tweet. The rest of the paper discusses the same.

Identify applicable funding agency here. If none, delete this.

II. Literature Survey

A lot of research and time has gone into Sentiment Analysis and it is handled as a NLP (Natural Language Processing) task at many levels of granularity. Sentiment Analysis is considered a 3 layered approach. Starting from document level classification (Turney, 2002;), it has been handled at sentence level(Hu and Liu, 2004) and more recently at an abstract/phrase level(Wilson et al.2005). As far as the sentiment classification techniques are concerned there have been research going around mainly 3 topics which are Machine learning approach, lexicon based approach and Combination/Hybrid Approach.

Some of the recent results on sentiment analysis of Twitter data are by Go et al.(2009), (Bermingham and Smeaton, 2010). Pak and Paroubek (2010) used distant learning to acquire sentiment data. They built models using Naive Bayes Classifier and Support Vector Machines, reporting that Support Vector Machines outperforms other classifiers.

III. Implementation

A. Dataset Description

Twitter is a social networking microblogging platform that allows users to post short messages called tweets. To apply sentiment analysis on these tweets, we are using **Sentiment140 dataset with 1.6 million tweets**, which is obtained from kaggle. The dataset contains 1,600,000 tweets extracted using the twitter api. Each tweet has been annotated (0 = negative and 4 = positive) for the purpose to detect sentiment.

B. Data Pre-processing

Since the dataset contains text data (tweets), text preprocessing is an important step for NLP tasks. It transforms the given text into a more digestible form for the machine learning algorithms. One of the first tasks of pre-processing that was done was to drop irrelevant columns. The only required columns for the purpose of analysis were "Tweets" column and "Sentiment" column. After removing the unimportant columns, the tweet was pre-processed as follows: a) Lower casing: each tweet was converted to maintain uniformity b) Tweets starting with "http", "https" or "www" are replaced by empty string as they are not important for the sake of analysis. c) "@" and hashtag signs are replaced by empty strings. d) Non-alphanumeric characters are again replaced with a space e) Stop Words removal: Stop words are the english words which do not add much meaning to a sentence. They can be

ignored without sacrificing the meaning of the sentence. f) Lemmatizing: It is important to convert a word to its base form. Lemmatization is the operation that converts a word to its base form. The dataset was divided into training and testing sets with a ratio of 0.95 to 0.05 i.e 0.95 of the data was used for training the model and it was tested on the rest 0.05 of the data.

C. TF-IDF Vectoriser

Term Frequency Inverse Document Frequency (TF-IDF) indicates what the importance of a word is in order to understand the document. This is a very common algorithm to transfrom text into meaningful representation of numbers which is used to fit machine learning algorithm for prediction. TF-IDF will give us a numerical weightage of words which reflects how important the particular word is to a tweet in a set of tweets. TF-IDF consists of 2 operations, **Term Frequency** and **Inverse Document Frequency**.

Term Frequency is the number of times a word occurs in a single document. The weight of a term that occurs in a document is simple proportional to the term frequency. The term frequency is given by:

$$tf(t,d) = \frac{count\,of\,t\,in\,d}{number\,of\,words\,in\,d} \tag{1}$$

While computing TF, all terms are considered equally important. However, it is known that certain terms may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing Inverse Document Frequency. IDF diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. IDF would be very low for the most occuring words and high for rarely occuring words. The IDF for a term t and corpus size N is given by:

$$idf(t) = log(\frac{N}{df+1}) \tag{2}$$

df is document frequency. Document Frequency gives the number of occurrences of term t in all documents.

Combining tf and idf, the tf-idf weight for a term is given by:

$$tfidf(t,d) = tf(t,d) * log(\frac{N}{df+1}) \tag{3}$$

TF-IDF builds up the "vocabulary" for the model. To be precise, TD-IDF assigns weights ranging between 0 and 1 for each word. These weights along with the sentiments act as training pairs for the Naive Bayesian Model for further predictions. To get a more deeper insights on TF-IDF, this algorithm was applied from scratch and fed into the sklearn Multinomial Naive Bayesian classifier to get clasification results. This gave us the same accuracy as that of the inbuilt sklearn tf-idf vectorizer.

D. Naive Bayesian Classification

Sentiment analysis is basically a text classification problem. So it is natural to use classification algorithms to solve the problem. One of the first models that we applied was the bayes naive classification. Naive bayesian is a simple algorithm which uses probability of the events for its purpose. It is based on Bayesian Theorem which assumes that there is no interdependence amongst the variables. The classification is done as per the following formula:

$$P(sentiment|word1, word2, ...)$$

$$= \frac{P(word1, word2, ...) * P(sentiment)}{P(word1, word2...)} \tag{4}$$

Since we don't really care about the probabilities, we can skip the denominator entirely since it just scales the numerator. Hence the new formula can be:

$$P(sentiment|w1, ..., wn) = P(sentiment) \prod_{i=1}^{n} P(wi|sentiment) \tag{5}$$

Here, $sentiment$ represent the class for which we are calculating the probability, (in our case it is either of positive or negative) and word represent the tf-idf transformed feature vector. In order to calculate the final label we do require a prior, $P(sentiment)$ which calculated from the training data. To be specific we used Multinomial Naive Bayes algorithm to classify the tweets because data generated by tf-idf is discrete. Following steps are performed to predict the correct class:

- Create a frequency table based on the words.
- Calculate the likelihood for each of the classes based on the frequency table
- Calculate the posterior probability of each class
- The highest posterior probability is the outcome of the prediction experiment.

The hyperparameter α, which is the smoothing parameter is calculated via a Grid Search Algorithm, which essentially tests out a bunch of different values for the hyperparameter and selects the value with which the model performs the best i.e the accuracy is the highest. In our case we got the value for α as 2.

1) Scratch Implementation: The Naive Bayesian Classifier was also written and applied from scratch to get a more deeper intuition of the algorithm. We have build a generic text classifier that puts tweets into one of two categories - negative or positive sentiment. The pre-processing techniques are essentially same as described above for the in-built naive bayesian classifier. The major change with this approach was the vectorizer algorithm. Instead of TF-IDF, count vectorizer was used to vectorize the pre-processed text. Since our model relies on multiplying many probabilities, those might become increasingly small and machine might round them down to zero. To avoid this, we've use log probabilities by taking log

of each side in our equation. The new equation is given by (S denotes sentiments):

$$log\,P(S|w1,...,wn) = log\,P(S) + log\prod_{i=1}^{n} P(wi|S) \quad (6)$$

As for the implementation, for each class we've found the number of examples in it and the log prior probability, which is given the number of samples in the class divided by the total number of samples. Along with that we've also constructed a vocabulary, which is essentially the set of all words we've seen in the training data.

In order to predict sentiment from text data, we've used our class priors and vocabulary. For each review we use the log priors for positive and negative sentiment and tokenize the text. Then for each word, we check if it is in the vocabulary and skip it if it is not. Then we calculate the log probability (log posterior) of this word for each class. We add the class scores and pick the class with a max score as our prediction.

It is entirely possible for a word in our vocabulary to be present in one class but not another. This can lead to wrong predictions as the probability of finding this word in that class would become 0 and rendering the entire probability for that class to be 0. We have used Laplace Smoothing to fix this problem. We've simply added 1 to the numerator to prevent the probability going down to 0. This gives us a robust classifier which is capable of classifying tweets into positive or negative sentiments.

E. Logistic Regression

Logistic Regression is considered as one of the most popular algorithm for classification. In this model, the probability of a tweet being positive or negative is found.
The probability is an outcome of the process which tries to find a plane ($\pi = W^T * X + b$) that best separates the positive class and negative class. Positive class is the one which is in direction of the plane while negative class is in the opposite direction. A classifier is defined for the best separation of the classes. The classifier which represents the positive class is $W^T * Xi > 0$ and similarly $W^T * Xi < 0$ represents the negative class. Here, Xi represents the input observation which is the weight of the tweet that is a result of TF-IDF Vectorisation. W represents how important that input feature is to the classification decision. Using the same, we are able to achieve the following optimisation function where Y_i is the class label. The positive class has $Y_i = 1$ while negative class has $Y_i = -1$.

$$W^* = argmax(\Sigma_{i=1}^{n} Y_i W^T X_i) \quad (7)$$

Our optimisation function is not robust to handle the outliers. So, we are using the sigmoid function to squish the values of the outliers. We are able to simplify and regularise the above optimisation function to get the following one after using logarithm and converting the function in form of a minimisation function. Here, the regularisation term essentially penalises our model for choosing very large values of W, hence avoiding overfitting.

$$W^* = argmin(\Sigma_{i=1}^{n} log(1 + exp(-Y_i W^T X_i))) + \lambda W^T W \quad (8)$$

Now, we have received the optimum set of parameters W. We can plug in the value of W and X_i in the following equation to get the probability of tweet being positive. If the probability is greater than 0.5 than it is considered a positive tweet while the one less having value less than 0.5 is considered a negative tweet.

$$h(X_i, W) = \frac{1}{1 + exp(-W^T X_i)} \quad (9)$$

IV. Results

A. Naive Bayes Classifier - Inbuilt

Multinomial Naive Bayesian Classifier was applied on the training data and was tested on the testing data. The multinomial NB model performs decently well giving us the overall accuracy of **79%**. Training accuracy was **82.6%** and testing accuracy was **78.9%**. Both the accuracy are close to each other, which is a good sign indicating that the model is not overfit.

	precision	recall	f1-score	support
0	0.78	0.80	0.79	39989
1	0.80	0.78	0.79	40011
accuracy			0.79	80000
macro avg	0.79	0.79	0.79	80000
weighted avg	0.79	0.79	0.79	80000

Fig. 1. Naive Bayesian Classification Report

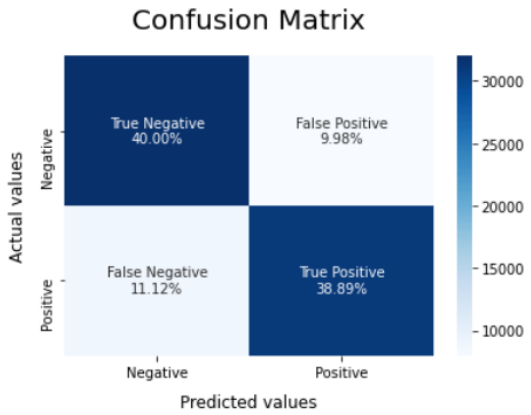

Fig. 2. Naive Bayesian Confusion Matrix

B. Naive Bayesian Classifier - Scratch Implementation

Multinomial Naive Bayesian Classifier was applied from scratch on the training data and tested on the testing data. The scratch implementation uses count vectorizor to vectorize the tweet text to numerical value. This model performs decently well giving us the overall accuracy of **77.24%**. This is accuracy is close to that of the inbuilt Naive Bayesian classifier

indicating that our model behaves similar to the inbuilt sklearn Naive Bayesian Classifier.

	precision	recall	f1-score	support
0	0.77	0.77	0.77	39999
1	0.77	0.77	0.77	40001
accuracy			0.77	80000
macro avg	0.77	0.77	0.77	80000
weighted avg	0.77	0.77	0.77	80000

Fig. 3. Naive Bayesian Classification Report (from scratch)

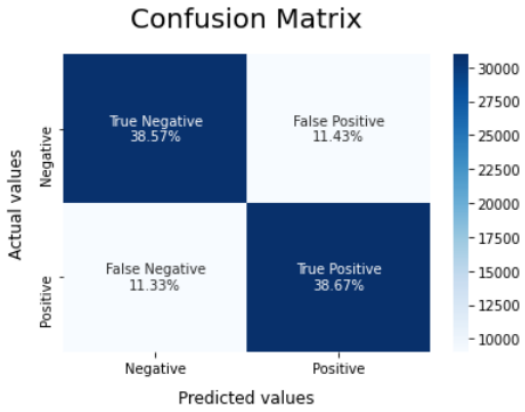

Fig. 4. Naive Bayesian Confusion Matrix (from scratch)

C. TF-IDF Scratch Implementation

TF-IDF was applied from scratch to build up the vocabulary and calculate the weight of each word from the training set. This tf-idf matrix was passed on to the naive bayesian classifier to predict the sentiments. The algorithm was built considering the value of ngram to be 1.

	precision	recall	f1-score	support
0	0.71	0.81	0.76	494
1	0.78	0.69	0.73	506
accuracy			0.74	1000
macro avg	0.75	0.75	0.74	1000
weighted avg	0.75	0.74	0.74	1000

Fig. 5. Naive Bayesian Classification Report (using custom TFIDF)

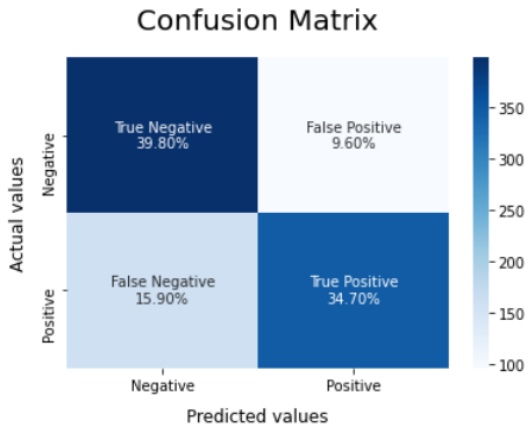

Fig. 6. Naive Bayesian Confusion Matrix (using custom TFIDF)

D. Logistic Regression

Logistic Regression was applied on the training data and was tested on the testing data. The logistic regression model performs decently well giving us the overall accuracy of **80%**. This model behaves similar to that of Naive Bayesian.

	precision	recall	f1-score	support
0	0.81	0.79	0.80	39989
1	0.79	0.82	0.81	40011
accuracy			0.80	80000
macro avg	0.80	0.80	0.80	80000
weighted avg	0.80	0.80	0.80	80000

Fig. 7. Logistic Regression Classification Report

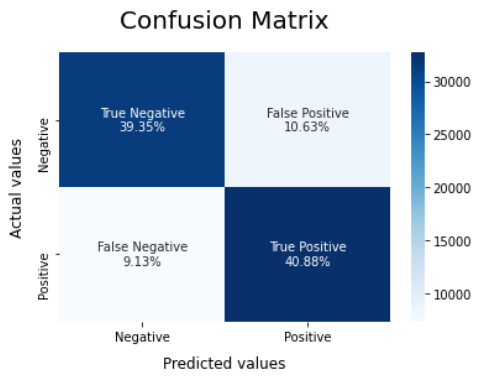

Fig. 8. Logistic Regression Confusion Matrix

E. Real time Classification

We have used pickle library to save our vectorizer and model to predict sentiments of random tweets in real time. These random tweets are first pre-processed, vectorized and fed into the model at real time to get the sentiment.

	text	sentiment
0	I hate you	Negative
1	I love you	Positive
2	I don't feel so good	Negative
3	All the best	Positive

Fig. 9. Tweet Classification Result

V. Conclusion

During the course of the project, we have successfully implemented the Naive Bayes Classifier and TF-IDF vectorizer from scratch on the twitter dataset to predict sentiments of unseen data. The Naive Bayesian Classifier gave us the accuracy of **77.24%**. SKLearn's Logistic Regression model and Naive Bayesian Model were also applied on the twitter dataset, which gave us the accuracy of **80%** and **79%** respectively.

References

[1] R. K. Dey, D. Sarddar, I. Sarkar, R. Bose, and S. Roy, "INTERNATIONAL JOURNAL OF SCIENTIFIC amp; TECHNOLOGY ... - IJSTR." [Online]. Available: http://www.ijstr.org/final-print/may2020/A-Literature-Survey-On-Sentiment-Analysis-Techniques-Involving-Social-Media-And-Online-Platforms.pdf. [Accessed: 14-Mar-2021].

[2] A. Adhikari, "Twitter Sentimental Analysis Using Naive Bayes Classifier(Process Explanation)," CoFoundersTown. [Online]. Available: https://cofounderstown.com/twitter-sentimental-analysis-using-naive-bayes-9eb73. [Accessed: 14-Mar-2021].

[3] Moin Nadeem, Mike Horn., Glen Coppersmith, Dr. Sandip Sen, "Identifying Depression on Twitter," [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1607/1607.07384.pdf. [Accessed: 14-Mar-2021].

[4] Md. Rafqul Islam, Muhammad Ashad Kabir, Ashir Ahmed, Abu Raihan M. Kamal, Hua Wang, Anwaar Ulhaq, "Depression detection from social network data using machine learning techniques," [Online]. Available: https://www.researchgate.net/publication/327251557_Depression_detection_from_social_network_data_using_machine_learning_techniques. [Accessed: 14-Mar-2021].

[5] Daniel Jurafsky, James H. Martin., "Logistic Regression," [Online]. Available: https://web.stanford.edu/ jurafsky/slp3/5.pdf. [Accessed: 14-Mar-2021].