Twitter Sentiment Analysis Of Movie Review with ML Techniques

Prashant Pathak(MT19051)
Mtech CSE
prashant19051@iiitd.ac.in

***Abstract*—Sentiment analysis is extracting and analyzing someone's views or opinions from textual data. People visiting various social networking websites and leave a huge amount of data in the form of tweets, comments, and blogs. Such kind of data can help us to detect the emotion or opinions of the people. Twitter Sentiment analysis is considered tricky compared to the other broad sentiment analysis because twitter data contain lots of slang. In this project, I will present a way in which we can do a sentiment analysis of twitter tweets regarding the movie reviews. We will apply techniques to extract features from our tweets and then apply various machine learning technique to correctly find the sentiment of the tweets.**

***Index Terms*—Sentiment Analysis, Feature Vector Machine Learning, Bag of Words, Doc2Vec**

I. Introduction

As the age of Social media is in the boom, people express themselves in many ways on these social platforms. People express themself with tweets on twitter, comments on Facebook, etc. Not only on social media platforms, but also in various e-commerce websites people express their views in the form of reviews. All these data can be used for finding the opinion of the masses. We can do a sentiment analysis to tell what people think about certain products and services. Doing an analysis of these data can tell companies what people are actually thinking about their product and what they can do to improve. Sentiment analysis combines various fields of study like statistics, natural language processing (NLP), and machine learning to identify and detect emotion.

There are two approaches to do sentiment analysis :

1) Lexicon based approach
2) Machine learning-based approaches

Lexicon based approach essentially uses a predefined dictionary in which words along with there sentiment score is stored. The lexicon model essentially scores every word with its sentient value and thus finds the overall sentiment of the sentence by summing the individual values. In the end, positive score means positive sentiment and negative score means negative sentiment and zero means neutral sentiment. Examples include "SentiwordNet", "Text Blob" and "Afinn". In this project, they have been used to give initial labels to tweets.

The machine learning-based approach treats sentiment analysis as a classification problem. In this sentence are initially labeled by some means and then we apply our traditional machine learning classification model to classify the data into several classes. Initial labeling can be done manually or by any means. In our project, we are going to explore Machine learning-based approach.

II. Data Gathering

As the user is continuously posting data on social networking the data in the social network is continuously floating. So for this project, I decided to fetch real-world data from the web. Twitter provides "Twitter APIs" to fetch its data. In code, I have used "Tweepy" which is a wrapper over "Twitter API" to fetch the data. Tweepy provides an easy API to fetch the data of twitter. To filter the data of the movie review tweets. Proper hashtags have been used. For example to fetch review tweets of "Joker", the search query used is "#jokerreview". Gathering data was difficult because there is no way to do proper filtering of tweets. In addition to movie reviews, other personal tweets were also fetched. Also in the standard access to twitter API, there is a limit on how many tweets you can fetch at a particular time. For the analysis purpose, a data set of 1700 is used which also has a duplicate record from retweets

III. Data Peprocessing

Twitter tweets are not in a format so that we can directly assign our model. Tweets are usually very noisy. It contains a lot of slang words. People also express their feeling used emojis. We have to handle all these before using our data further. Sections III-A and III-B below explain all the steps of pre-processing.

A. Data Cleaning

Following steps are performed for data pre-processing:

- **Removing Duplicate rows:** As tweets are gathered from Twitter API. It may contain duplicate data. Duplicate data not only just increase the data size but also does not add any value to our data. so, we remove duplicate rows from the data set.
- **Remove neutral words** Tweets contain words such as user-handle name, links of website, hashtags, etc. We can remove all these words because it has nothing to do with tweets' sentiment.
- **DE-encoding emojis:** Emojis in the tweets play an important role in determining its sentiment. People with the help of emojis share a lot of their emotions. For this reason, we have converted emojis into words so they can be utilized.

B. Correcting Data Format

- **Tokenization and Part of Speech(POS)-tagging:** In Lexicon models we have to pass individual words along with there POS-tags. For the above reason we first tokenize the data. In tokenization, we break the sentence into words. After breaking it into words we apply POS-tagging. In POS-tagging we assign part of speech (like noun,verb) to words.
- **Removing Stop-word:** Stop words are those words which are most common in the language and are often removed before processing of natural language data. We have not removed stop word before the POS tagging because removing stop words can change the actual POS-tags of the words.
- **Lemmatization and Stemming [3]:** Usually to maintain the grammar, the word changes its form. Example, walk become walking in its present continuous tense form . Lemmatization and Stemming are used to reduce inflectional, derivational forms of a word to a common base form. If words are not brought into base form then they will not match during counting and the proper word count will not work.

IV. Ground Truth Labelling

As data is fetched from the Twitter API, data is not labeled, So to apply supervised learning technique we have to first, label the data. All the model used here are Lexicon model which have dictionary base. 3 techniques: **Text Blob [6] ,SentiWordNet [7],Afinn**is used to label the data.

Text Blob sentiment analysis returns the polarity score of the sentence. The score is between [-1,1]. Less than zero return negative sentiment, 0 return positive sentiment and greater than 0 represent positive sentiment.

SentiWordNet is an enhanced lexical resource, explicitly devised for supporting sentiment classification and opinion mining applications. It has a large corpus of POS-tagged English words along with their sentiment which is used to give sentiment to new words. SentiWordNet returns the POS tags of each word. Scores are between [-1, 1] it used both word and their tags.

The AFINN lexicon is one of the simplest and most popular lexicons that can be used extensively for sentiment analysis. The latest version of the lexicon contains over 3300+ words with a polarity score associated with each word.

The final labeling is done by the voting of the three classifiers.In case of conflict, priority is given to SentiWordNet.

V. Data Visualization

- We visualize our data with the help of the word cloud.
- After initial visualization, I found that some of the words are common between all the 3 classes and have a high frequency.
- I removed the common word like 'watch', 'film', 'movi', 'review' from the data set.

Fig **??** shows the word cloud of positive words respective graph

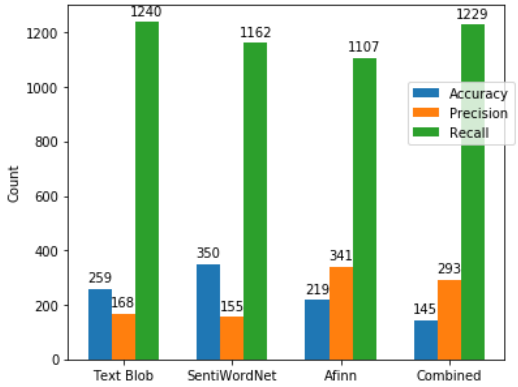

Fig. 1. Class count with different Lexicon Model

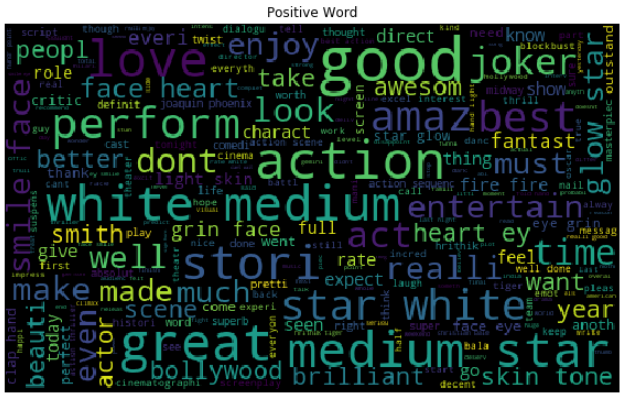

Fig. 2. Positive word cloud

VI. Feature Extraction

After the labeling process, we have a label. But the data is still in the text, so we have to vectorize it and convert it into the text if we want to send it to a machine learning model. In this project, I have tried a different method of vectorizing and choose the feature which gave the best result in the data.

A. Bag of Words

The first approach used is Bag of words. In this model, a text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity. With the help of the count, we make a sparse matrix. There is two way to doing Bag of word vectorization **Count Vectorization and Term Frequency — Inverse Document Frequency (TFIDF)**.

Count Vectorization involves counting the number of occurrences of each word appearing in a document. It tells us how many times a word appears in a document and how many times it appears in a particular sentence. We can count n-words together in the document. This is called N-gram vectorization.

The term frequency refers to how much a word appears in a document. Inverse document frequency refers to how commonly or rarely a word appears in a document. If the TFIDF score is pretty high, it means the words are pretty rare and are good at discriminating between documents. We

have tried both Count Vectorization and TFIDF with 1-gram,2-gram, and 3-gram and tested the baseline accuracy on the Naive Bayes. For the evaluation, we have used accuracy, precision, recall, and f1score.

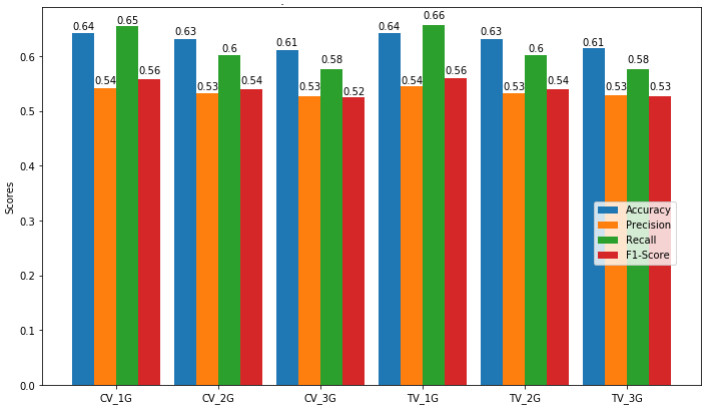

Fig. 3. Evaluation of "Bag of words" vectorization with different metric

B. Word embedding

Word embedding is another way of representing our data. We make a vector of each word by doing a one-hot encoding of the complete document. That is our normal text is now represented by an n-dimensional vector. Now what we want is that the words that have similar meanings should be near to each other like "good" and "great" and words that have different meanings should far form each other. In the former case, the cosine similarity must be high and in the latter case, cosine similarity must be low.

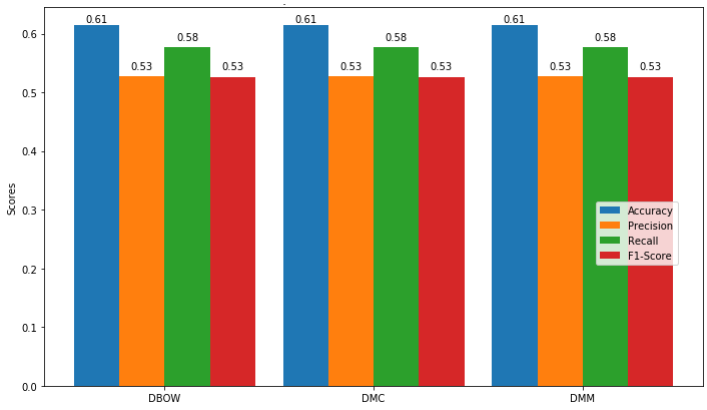

Fig. 4. Evaluation of "word embedding" vectorization with different metric

We can see in our case "Bag of word" with 1 gram performs the best. Both Count vectorization and TFIDF performs equally good. It gives an f1score of **56%** We have used different metrics because the data set is highly imbalances.

VII. Handling Imbalance Data

As we can from Fig 1 see that the data is highly imbalanced. So for proper classification, we have balances the data set. we have converted the count so as all the classes have an equal count of 600.

VIII. Performance of Classifier

In the project, I used Naive Bayes, SVM, and Logistic regression. For SVM and Logistic Regression, "gridsearchcv" is used for finding the best parameter and to prevent over-fitting.Best parameter of SVM are **C = 1 and best kernel is "linear"** and for logistic regression **C is 1000**

TABLE I
Evaluation Metric of Different Model

Model name	Accuracy	F1-score
Naive Bayes(On unbalances classes)	64.26	56
Naive Bayes	61.8	52.7
SVM	72.42	59.0
Logistic Regression	72.18	55.94

IX. Conclusion

We developed the project in two phase.In the first phase of the project we initially classified the data with the help of Lexicon data analyser and gave label to our data. Then in the second phase of the project we initially extracted the feature out of tweets and then used those feature to classify the data.We found that the 1-Gram extracted model performs the best on this data . Balancing the data does no good for our classifier as f1 score of Naive Bayes reduces after balancing the data. Linear SVM performs best for this tweeter data. and has a F1 score of **59%**

References

[1] Akshay Amolik, Niketan Jivane, Mahavir Bhandari and Dr.M.Venkatesan, 'Twitter Sentiment Analysis of Movie Reviews using Machine Learning Techniques.',published in IJET' on January 2016.

[2] https://towardsdatascience.com/sentiment-analysis-on-swachh-bharat-using-twitter-216369cfa534.

[3] https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[4] https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis

[5] https://github.com/aesuli/sentiwordnet

[6] https://medium.com/@joshsungasong/natural-language-processing-count-vectorization-and-term-frequency-inverse-document-frequency-49d2156552c1.

[7] http://thesocialchic.com/2013/04/26/how-to-master-twitter-search/