Sentiment Analysis of Twitter Data for Airlines

SHUBHAM BHANDARI[1]

[1] *Andrew Id: sbhandar, sbhandar@andrew.cmu.edu*
[1] *Applied Machine Learning - 11663*

11th December 2018

Twitter has rapidly turned into the customer service hub for the airline industry and it offers the opportunity for any brand to stand out from the competition, expanding its audience and even turning around a negative experience. According to a survey by Millward Brown and Twitter, 40% of travelers have contacted a travel brand on Twitter, but only 28% of them received a response. However, out of those who had contacted the airlines 73% of the users felt more positively about the brand. In fact, study has shown that replying to tweets has revenue generating potential. In this project, we use the tweets to learn about people's flight experience, which will help airlines the area to focus more on. In this paper, we worked on a data set comprising of tweets for 6 major airlines and performed a multi-class sentiment analysis. The analysis was carried out using different classification strategies: Naïve Bayes, AdaBoost, SVM and Logistic Regression. Based on the results obtained, the accuracies were calculated to draw a comparison between each classification approach.

1. INTRODUCTION

In the last few years, social media has become the de-facto online customer service platform. For customers who are finding it more and more difficult to get a person on the phone, social media frequent offers real-time answers to pressing travel questions. In the span of few minutes, a representative working on Twitter can process a customer's request, improving company's efficiency and help improve its reputation. Customer feedback is very crucial to airline companies as this helps them in improving the quality of services and facilities provided to the customers. Sentiment Analysis in airlines industry is methodically done using traditional feedback methods that involve questionnaire, forms or online surveys. These procedures might seem quite simple on an overview but are very time consuming and require a lot of manpower that comes with a cost in analyzing them. Moreover, the information collected from the questionnaire have a chance of being inaccurate and inconsistent, as it may not be taken seriously by the customers. The irrelevant details and incorrect details may resulting noisy data for analysis. Whereas on the other hand, Twitter is a gold mine of data with over 100 million people using the platform and tweeting more than half a billion tweets daily. With the rise and advancement of technologies, it has become easier to collect tweets and apply data analysis techniques on them[1]. Twitter is much more reliable as it's the genuine user who makes the request, complain or interacts with the airline directly for a purpose. Once the relevant tweets are collected by the airline's team garners all the data, pre-processing of the data is done to remove all the unwanted details/features and sentiment analysis is performed on the cleaned tweets[2]. This give the airline a broader perspective about the feeling and opinions of their customers. In this paper, several pre-processing techniques followed by the application of various machine learning strategies are used to determine the sentiment of the customers. The classifiers are them compared to each other for their accuracies and an attempt on why the algorithms behave better/worse than other strategies.

2. BACKGROUND AND RELATED WORK

The field of sentiment analysis have been developed in the recent years, with focus on many issues of sentiment analysis in regards to domain of the data sets, corpus types and multilingual context. Research on sentiment analysis of twitter data has usually been focused on (i) understanding the textual entity whether it is subjective or objective and (ii) identifying polarity of subjective texts. Sentiment analysis of twitter data has been extremely important component of any business and the work behind it can be highly exhaustive but could yield high revenue for the company.

Text data from twitter or micro blogging texts has lot of unnecessary data, which will be covered in later sections. The paper by Dimitrios Effrosynidis et. al[3]. covers various techniques that are usually used to clean and process the data collected

from these kind of data source.

In the paper Role of text pre-processing in twitter sentiment analysis[4], the authors examined the effects of pre-processing on twitter data for improving sentiment classification. Noisy data like URLs, hash tags, user mentions, punctuation, and stop words, and they identify the importance of slang words and spelling correction. To avoid the process of annotating the instances manually, new approach has been introduced which classifies the instances positive or negative with respect to a query term. The training data used by them is only emoticons[5].

3. DATA EXPLORATION

A. Data Description

For this project, the data-set was taken from Kaggle which contains a total of 14640 tweets. The tweets were based on six major US airlines: Delta, Southwest, virgin America, United and US Airways. They were a mixture of positive, neutral and negative sentiments. The data provided was hand-annotated (labelled) and provided in an excel file. As the data-set was labelled, supervised machine learning strategies was also developed. Each instance represents a customer's experience with the mentioned airlines. The class value is indicated by the variable airline_sentiment, which can be positive, negative or neutral. There are many cases where a person tweets multiple times, which can be for different airline. It is assumed that the person who tweets regularly or reviews airlines is always fair and not critique in nature. "airline_sentiment" is the class value and "text" is the feature that has to be considered, and all other features are meta-data attributes.

tweet_id	twitter id
airline_sentiment	class value (positive, negative or neutral)
negativereason	Reason for labelling it as a negative tweet
airline	airline carrier
name	Tweeter account name
retweet	number of retweets
text	text tweeted
tweet_coord	coordinates from where the tweet was sent
tweet_location	location of tweeter
user_time	Time zone of tweeter

The data set has 20% positive, 60%negative and 20%neutral tweets. The negative reasons label are bad flight (7.45%), cancelled flight (9.62%), customer service issue (39.8%), damaged luggage (0.84%), flight attendant complaints (6.05%), late flight (1.99%), long ques (19.97%), and lost luggage (8.23%). No such reason is provided for positively labelled data.

The data-set was divided into 3 parts, development data, training data and testing data, in the ratio of 1:7:2. To create the 3 data-sets stratified random sampling was done, to make sure the algorithm has enough training instances of each class value. The implementation of the code was done using Weka, LightSide and Jupyter which is an environment for the programming language Python.

B. Data Preprocessing

The procedure of cleansing and preparing text that is going to be classified in a machine learning algorithm is called as preprocessing. Text in twitter contains significant noise, noise is the part of text which is not useful for classification and often confuses the trained algorithm resulting in decline in accuracy. Fayyad, Piatetsky-Shapiro, and Uthurusamy (2003), observed that the total percentage of noise in twitter text dataset can reach up to 40%.

As twitter provides a platform but restricts the user from using more than 140 characters, the words used by the tweeter are prone to spelling errors, abbreviations and slangs. Quite often words are spelled incorrectly with some alphabets in the words repeating, for emphasizing their emotions. For ex. Use of word "Helloo!!!" instead of "Hello". It is observed quite frequently that the mis-spelled and the actual word are responsible or influencing the algorithm towards different class value.

To direct a tweet towards someone, the '@' + 'twitter id of airline' or user mention is used. It doesn't contain any sentiment and the correct approach is to remove it during pre-processing. Some tweets also include link to URL. The content of the link cannot be read in an algorithm, but each term in the URL will be gain some feature influence if not removed before using the data for training.

After analyzing the data, there were a lot of issues with the data were found out. A brief description of the pre-processing techniques used for each type of noise is listed below:

B.1. Remove Unicode Strings and Noise

Some of the tweets had Unicode strings like "x15" which could be formed during creating data-set This can be removed with help of ReGex either in LightSide or in Python.

B.2. Replacing URLs and user mentions

Each instance in the data set had a mention in the tweet. For categorizing tweets, a hash tag () before a relevant keyword or phrase in their tweet is used. Some of the tweets also share some link or upload an image (or a video) with their tweet. None of these contain any sentimental value and must be removed before processing the data. For example,
to convert www.* or https?://* to URL
we can use,
`re.sub('((www.[^\s]+)|(https?://[^\s]+))',URL,tweet)`/
forreplacing,
`@username to AT_USER`
wecanuse,
`re.sub('@[^\s]+','AT_USER',tweet)`

B.3. Replacing Slang and Abbreviations

Tweets written are very informal and contains many slang and abbreviations. Slang comprises words and phrases which are very informal and usually can only be understood with the context.

Removing numbers one of the processing techniques used is removing numbers from the tweet, as they don't have any sentimental values. This method was used after replacing the slangs, as there were some instances where a digit was used with a word. For example, there were multiple cases where people had written 'gr8' instead of 'great'
ReGex:
`re.sub(r'\d+', '', string)`

B.4. Replacing repetitive punctuation

There is a common norm of using punctuation's like exclamation mark, question marks or full stops repetitively. They are usually used to express intense emotion. In this technique if the punctuation is found more than once, it'll be replaced with a word. For example, if '!!!' is found in the text, it'll be replaced by 'multiExclamationmark'.
ReGex:

```
([?!.:,;]|\s)\1+
```

B.5. Removing Hashtags

Hashtags are very important for twitter , as the word after the hashtag is actually going to categorize that particular tweet into that 'word' category. So removing that word itself wont be a solution, as it may signify something important, so just removing hashtag should do the work.
Regex for removing is

```
re.sub(r'#([^\s]+)', r'\1', tweet)
```

4. MODELS

A. Naive Bayes

Naïve Bayes uses Bayes Theorem and is considered the most suitable model for comparison with baseline performance. For our classification problem it can be generalized as:

The algorithm assumes that all the features are independent of each other, which is a naïve assumption to make. An idea of its working can be intuited from the formula. Using the tweet as an input, the algorithm will determine the label from words given in training data. During testing the words are compared with those provided in the training data, and the Naïve Bayes method determines using that the most likely sentiment.

$$P(label|features) = \frac{(P(label)) * P(features|label)}{P(features)}$$

Weka provides the Simple Naive Bayes classifier, which will be used for the project as baseline performance, and Multinomial event model within Naive Bayes.

B. Support Vector Machine

One of the best working algorithms that perform well on sentiment analysis classification is Support Vector Machine (SVM). The kernel function which has various filter will affect the performance estimate, therefore suitable kernel is chosen to improve the efficiency of classification. Support vector separates data linearly and non linearly. The goal of SVM is to separate positive and negative training examples by finding n-1 hyper planes. For this project, I have used Weka SMO implementation for our task. There are filter, wrapper and embedded approaches for feature selection provided within WEKA which are used for the project.

K(u, v) = $u^T v$

K(u, v) = exp-mod(u-v)^2 $\frac{}{2\sigma^2}$

C. AdaBoosting

It is a machine learning meta-algorithm, which can work in conjunction with many other learning algorithms. The output of other 'weak learners' is combined into a weighted sum that represents the final output of the boosted classifier. The performance of the decision tree is boosted by using this ensemble technique. In adaboost, in each iteration the weights are applied to each of the training samples. The AdaBoost.M1 algorithm works by calling a weak classifier several times. For our task, we simply let the weak classifier be an indicator function of a certain word. Each time a weak classifier is called, it is provided with a different distribution over the training examples. The idea of boosting is that it assigns higher probability to the part that it doesn't classify correctly, in the hope that the new weak classifier can reduce the classification error by focusing on it. The algorithm selects the weak classifiers to use, and learns the weight for each weak classifier. In the end, hypotheses from each iteration are combined into one final hypothesis.

D. Logistic Regression

Logistic regression is implemented when the outcome is either 0 or 1. Both Logistic Regression and Naive Bayes are linear classifiers, Naive Bayes is generative classifier which uses the concept of likelihood under the assumption that all features are independent, whereas Logistic Regression is discriminating and uses logistic function to get likelihood. As the data set will only be classified into positive and negative category, it will certainly have a better performance than Naive Bayes.

Sigmoid Function:

$h_\theta = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

E. Experimentation

E.1. Data Exploration

The development data set was used to perform the exploratory data analysis. In this process manual study of the instances, along with the feature table was studied. Different types of algorithms were ran using WEKA to check which algorithm gives the highest performance estimate, and feature engineering and parameter tuning would be done on the basis of that.

AdaBoost, Naive Bayes, Logistic Regression and Support Vector Machines were used for data exploration by comparing their performance estimates after running on the development data. The results of the algorithms is given below:

Classifier	%Correct	Kappa
Naive Bayes	59.62	0.39
AdaBoost	68.41	0.47
SVM	71.83	0.5162
Logistic Regression	64.58	0.43

As shown in the table the SVM had the best performance with accuracy of 0.7183 and kappa of 0.5162. Therefore, SVM model was used error analysis and parameter tuning.

E.2. Baseline Performance

To get the baseline performance, LightSide was used. First the Cross-validation data was loaded into the LightSide application, where feature extraction under the condition of unigram and Track Feature Hit Location was performed. After that the build models tab was selected and SVM algorithm was selected to train and evaluate the model. A baseline performance of 0.721 was achieved along with the kappa value of 0.52 .

For error analysis, after training the model on cross-validation data set, it was tested on development data. The accuracy and kappa value achieved were 0.691 and 0.48 respectively.

E.3. Error Analysis

For understanding the type of error in the model, explore result feature within the LightSide was used. There were many interesting errors wee identified, some of which are given below along with its description and method for removing that problem: N-grams Unigram doesn't usually contain as much information as compared to bigrams and trigrams. The basic principle behind n-grams is that they capture the language structure, like what letter or word is likely to follow the given one. The longer the n-gram (the higher the n), the more context you have to work with. Optimum length really depends on the application – if your n-grams are too short, you may fail to capture important differences. On the other hand, if they are too long, you may fail to capture the "general knowledge" and only stick to particular cases.

LightSide provides the option of Unigram, Bigram and Trigram in configure basic feature option, out of these when unigram + bigram feature was checked the improvement was highest.

Term Frequency Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence. The concept is that, the words which occur more frequently than the other have more possibility of not influencing the model. For example, the usage of stopwords like, a any the they, but, then etc.

Term Frequency - Inverse Document Frequency (TF-IDF) It was observed from the explore result tab that, words like don't, can't, 'thanks' were given less higher weight than words like 'disappointed', 'ridiculous' were given less weight. To get rid of this issue TF-IDF can be used.

This can be achieved using the StringToWordVector filter, in which the option of IDFTransform and TFTransform can be switched to true.

Stemming It was observed that words in their different form have received different feature weight and influence. To avoid such example and to make the algorithm consider all the words with a particular stem as same, stemming is used.

LightSide provides the option of stemming in configure basic feature which helped in solving this issue.

Handling Capitalized Words Quite often people tweet in capital letters to emphasize their point. This data when processed within the model can lead to confusion between capitalized and lower case letters, leading to different feature weight and influence. To avoid this regular expression can be used within the LightSide application or in python.

For example, by using the command ".lower()", tweets will be converted into lower case.

Although I realize about this problem during error analysis, I believe it should have be done during data preprocessing.

When the dataset was run through this particular model the performance estimate came around 0.76 and Kappa value was 0.57.

E.4. Parameter Tuning

Parameter tuning or Hyperparameter tuning is an important part of model tuning and the algorithm cannot be put to use unless this process is done. Hyperparameters are variables that govern the training process itself. These parameters are not related to the input data but are configuration variable. it is important to optimize the model parameters by training process: that is the data is run through the operations of the model with some parametric values and compared with some different values.

For tuning I used CVParameterSelection in Weka. A total of 5 folds in SMO model with configuration varying from 1 to 5. The result showed that the value of hyperparameter is optimal at C = 2.

T-Test TTest function was applied after doing the tuning to check if the improvement is significant or not.

Baseline	New
0.75	0.77
0.72	0.74
0.79	0.82
0.76	0.84
TTest	0.079

As the value of TTest is greater than 0.05, it is statistically significant.

5. FINAL EVALUATION

To generate the final result, cross validation and development dataset were used as training data and testing data which hasn't been used so far, will be used. The SMO with C = 2 model when used over the dataset gave the accuracy of 0.781 and kappa value as 0.601, which was a significant improvement over the baseline model.

6. DISCUSSION

The project performed using machine learning tools and techniques included background research on the topic, data exploration, which mainly included data preprocessing, different machine learning models, data exploration techniques, like error analysis, parameter tuning.

One of the important learning's was how small details in the text could result in an inefficient model performance and could lead to introducing a new pre-processing techniques. As tweets has a very informal language, regular expression played a major role in solving the project. Although I wasn't able to use regex in the LightSide application, I was able to switch to Python for performing preprocessing techniques, and then to LightSide or Weka for analyzing the results.

REFERENCES

[1]Cambria, Erik Rajagopal, Dheeraj Olsher, Daniel Das, Dipankar. (2013). Big Social Data Analysis. Big Data Computing. 401-414. 10.1201/b16014-19.

[2]BUDD, L.C.S., 2012. @doesyourairlinetweet? An empirical analysis of the use of twitter by 50 international airlines. Journal of Airline and Airport Management, 2 (2), pp. 124 - 135.

[3]Effrosynidis, Dimitrios Symeonidis, Symeon Arampatzis, Avi. (2017). A Comparison of Pre-processing Techniques for Twitter Sentiment Analysis. 21st International Conference on Theory and Practice of Digital Libraries (TPDL 2017).

[4] Tajinder Singh, Madhu Kumari, Role of Text Preprocessing in Twitter Sentiment Analysis, Procedia Computer

Science, Volume 89,

[5] Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." CS224N Project Report, Stanford 1.12 (2009).