SENTIMENT CLASSIFICATION ON PRODUCT REVIEWS

Author: Arunava Munshi

JUNE 5, 2019

Table of Contents

1 Introduction

This project is about Sentiment Analysis, which is a technique of mining of emotions, opinions or attitudes from different natural language sources (e.g. text, speech or other sources) by using Natural Language Processing (NLP) techniques. Sentiment analysis is also called opinion mining, as it involves the classification of text into classes such as 'Strong positive' or 'Negative' (Vaghela & Jadav, 2016). This type of analysis is involved in converting unstructured data into structured data and extracting 'Features' which are used by the model to learn to classify text into the given polarity levels (Vohra, 2012). The sentiment analysis done for this project is about building a system that classifies a large number of Yelp reviews into different polarity levels or opinions based on what was written by the Yelp reviewer for five different.

The goal of this project is to pre-process the large dataset of reviews, or in other words, clean the data, treat the data according to how the classification algorithm can take in data, and build the most appropriate machine learning classification algorithm to have results as accurate as possible. The reviews will be classified into 'Strong negative', 'Negative', 'Neutral', 'Positive' and 'Strong positive', which are represented from numbers 1 to 5 respectively. The accuracy of this model will mostly depend on the feature selection, which also highly depends on the pre-processing of the text. We have used Python coding for this project.

1.1 Data set

All team members from each group has been given with three datasets, which are as follows-

- **train_data.csv:** This dataset contains an identifier field named 'trn id' and the product reviews. Data size: 650000 x 3
- **train_label.csv:** This dataset also has the same identifier 'trn id' and the respective five sentiments for each review. Data size: 650000 x 2
- **test_data.csv:** This dataset contains the identifier field named 'trn id' and the customer reviews, but there is no sentiment given for this data. Data size: 50000 x 2

Our aim is to predict the sentiments for the test_data.csv, given the data in train_data.csv and train_label.csv. In order to do that we follow the following steps such as Pre-Processing, Feature Selection and Model building, which we shall discussed in the following sections.

2 Pre-processing and feature generation

2.1 Pre-processing

Pre-processing is the first and foremost part of any text mining. Here we mainly clean the unstructured data into a manageable format through the Data Wrangling technique. Here we follow the below steps.

1. Firstly, “train_data.csv” and “train_label.csv” are read and combined into one dataset.
2. The train data is split into five parts based on the label. This step is carried out because we want to extract a balanced sample from the data.
3. Any duplicated product reviews are removed because it could later affect the accuracy of the appropriate model.
4. From step 2, from each train data split, we extract a sample of 5,000. We take a balanced sample such as this one since the original data’s labels are balanced the same way and therefore we would want our sample to be as unbiased as possible. We sample the data before running pre-processing code because the run time of the code is much faster than when trying to process the full large dataset. The final outcome is a dataset with 25,000 samples which includes 5,000 samples from each train data split for each of our experiments conducted later.
5. Since emoticons also have emotional meaning, the most common emoticons that were converted to readable emotion text for each review are listed below:

X-(	Angry	(-:	Happy
:-D	Very happy	=/	Mad
</3	Broken heart	:-B	Nerd
O.o	Confused	^_^	Overjoyed
B-)	Cool	:-/	Perplexed
:_(	Crying	=(	Sad
:'(	Crying	:-(	Sad
-	Dazed	:(	Sad
:-(	Frown	:S	Sarcastic
:(	Frown	=O	Shocked
:-(	Frown	:-o	Shocked
=P	Tongue out	:P	Tongue out
:-P	Tongue out	:o	Shocked
=)	Happy	:-J	Tongue in cheek
:-)	Happy	:-\	Undecided
:)	Happy	;-)	Winking
<3	Heart	;)	Winking
:-|	Indifferent	|-O	Yawn
X-p	Joking	=D	Very happy
XD	Laughing	)-:	Sad

6. A dictionary of all contractions (stackoverflow.com, 2018) with their expansion as the value is included to find and expand all contractions in the product reviews. We expand these contractions because the words could possibly get incorrectly tokenized and have the meaning of the word change. For example, we have changed ‘won’t’ into ‘will not’, ‘can’t’ into ‘cannot’.
7. All words in each review have been converted to lowercase words. This is done because case normalization helps to treat uppercase and lowercase words the same in the product reviews.

8. Each review in the dataset is tokenized into English words to perform the following steps.
9. All digits in the data have been removed since they do not have enough meaning to be represented as a feature for sentiment analysis.
10. Part-of-Speech tagging is performed because we would want to know what type of word it is. For example, 'Badly' has the tag 'Adverb' or 'Restaurant' has the tag 'Noun', etc.
11. Lemmatization is then performed on the POS tagged tokens. This is preferred over stemming because unlike lemmatization, stemming does not consider the difference of meaning between words like 'cook' and 'cooking', with 'cook' possibly meaning the expertise and not the activity.
12. Emphasis words such as words with repeated characters in the word like 'Wooooooowww' have been replaced with the word 'emphasis' because we can capture these emphasis words which are connected to their respective sentiments.
13. Finally Bigrams (pair of words) are created and combined into one list with the Unigrams (single independent words) to be processed for feature selection.

Certain stop words like 'not' or 'but' or 'will not' are not removed because they contribute negative and positive meaning and have significant impact on the sentiments.

2.2 Feature generation

Feature selection is the second most important step after text pre-processing because these features are fed into the classification models for predictions. In a text mining problem like sentiment analysis, the individual text components called tokens (can be word or stems) are treated as features. In text mining problems, documents (here reviews) are represented in sparse matrix format called document vectors. There are two popular document vectors named Count Vector and tf-idf Vector are available to us.

Count vector representation gives us the frequency of each word within each document called term frequency. This is one of the most popular methods of structuring text data which is by nature unstructured data. The assumption behind count vector representation is that the order of the words does not matter as much as the words themselves and hence does not consider order.

Term Frequency-Inverse Document Frequency (TF-IDF), on the other hand, gives us the term weight for the terms within each document. In TF-IDF, a word with a high term frequency in a count vector representation would have a high weightage unless that word also has high document frequency. Stopwords for example would not be given a high weight in any document because even though they appear many times in a single document, they would not receive a high weightage since they do not have much sentiment meaning. Because of this reason, TF-IDF is preferred over Count Vector representation and is used in our case. Below is the formula for TF-IDF:

$$tf \cdot idf\,(w,d) = tf\,(w,d) * idf\,(w)$$

where, 'w' is the word, 'd' is the document, tf is the term frequency of the word and idf is the inverse document frequency which is a measurement of how important the word is. In

this case, we have created three feature sets with below statistics and the third one is taken because of its final highest model accuracy.

Feature set statistics:

	Feature 1	Feature 2	Feature 3
Total number of documents	25,000	25,000	25,000
Size of the vocabulary	46,930	45,128	45,128
Lexical diversity	64.656	50.4699	50.4699
Number of features	1039	950	1374
Tokens Accepted	Unigram Token	Unigram Token	Both Uni and Bigram

3. Models

Our aim here is to develop a sentiment classifier with five different sentiments levels. Researches so far have achieved great success in sentiment analysis through classifiers such as Support Vector Machine (SVM) and Naïve Bayes (S. M. VOHRA, 2012). It is also observed that SVM performs generally better than Naïve Bayes when the sentiment classification is restricted to only two classes (Vaghela & Jadav, 2016). Here we will mainly dissect these two classifications into our sentiment analysis and see which model is performing better. We choose these two classifiers over the other classifiers because of some reasons. Firstly, this is a classification task on text corpus, which is massive in size. In text based classification, the word tokens are generally considered as the features. Considering that sentiment classification from product reviews is more dependent on individual words, the classifiers such as Random Forest, LDA, QDA etc. have major focus into the interrelatedness among the features that makes the model interpretation unnecessarily complex and often becomes more CPU and memory intensive. Whereas, on the other hand, Naïve Bayes treats each and every feature independent of the other, making the model more interpretable for text based feature selection. SVC does the classification depending only on the support vectors (discussed later), which are usually small in number. So this algorithm also has advantage in terms of interpretability within text based classifications.

3.1 Model 1: Naïve Bayes

Naïve Bayes (Zubrinic, Milicevic, & Zakarija, 2013), which is a Bayesian Classifier, does a strong assumption that its attributes are independent of each other. This, as discussed above, provides an advantage in sentiment classifications from product reviews where each word from a review pointing to a particular sentiment is independent of the other words.

$$P(x|y=c) = \prod_{i=1}^{D} P(x_i|y) = c$$

(Zubrinic, Milicevic, & Zakarija, 2013)

Above formula shows the so called NB assumption. Even though, this assumption does not hold true in reality, NB often performs way better than other classifiers because of this simplicity and for similar reason, this works well for sentiment analysis. Based on the assumptions of Naïve Bayes, a document class probability can be calculated as the probability of the attribute values divided by the word attributes. This model considers the probability of each word token is independent of the context of the word and thus this classification technique is a better fit for Sentiment analysis where each word directly affects the sentiment and the correlation among them are not much relevant.

3.2 Model 2: Support Vector Machine (SVM)

Support Vector Machine (SVM) is also a classification technique that tries to minimize the structural risk of model (Zubrinic, Milicevic, & Zakarija, 2013). It has an important property that it can learn independent of feature dimensionality and it's also independent of the number of features. SVM, in its simplest form, separates positive examples set from negative examples set and thus, one side of this hyper plane is considered positive and the other as negative, making this plane a decision boundary for a two class SVM. This can be mathematically interpreted as – given a training data points D, a point set n can be interpreted as

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n$$

(Zubrinic, Milicevic, & Zakarija, 2013)

Where y_i cab is either +1 or -1. Even though fundamentally SVM is a two-class classifier, it can show multi-classification by combining many binary classifiers. In a high dimension classifier like sentiment analysis, where the tradition approach considers most feature irrelevant, each feature can have important weightage. Because SVM protects model from over-fitting, it is independent of the number of features and can potentially handle the large feature space like sentiment text. So, it is good classifier for sentiment analysis.

3.4 Discussion of model difference(s)

In this project, we are using "scikit-learn" package to develop our model in python. In both the cases multinomial Naïve Bayes and SVM classifiers have been used. Naïve Bayes model works on the basis of conditional independence of attributes, which is not necessarily the optimality of Naïve Bayes. So, that is why, with the strong assumption of the attributes are not dependent on each other, NB still fetches optimal result for problems like sentiment analysis. Whereas on the other hand, SVM, with its karnel feature, provides the concept of dimension reducibility for high dimension problems such as sentiment analysis. Dimension reduction provides a higher interpretability to SVM over Naïve Bayes when the number of features is very large. But at the same time, SVM may not work as optimal as compared to Bayes classifier, which can give highly optimal results. However, a research (Alves & Baptista, 2014)revealed that SVM performs mostly better than Naïve Bayes in terms of sentiment analysis. As part of this assignment we are using linear SVC, a variation of SVM, which can scale better than SVM for high dimensional problems and way computationally efficient than SVM.

4 Experiment setups

4.1 Model Parameter Settings

In this work, we set few of the model parameters for both Naïve Bayes and Linear SVC (variation of SVM) we are using here. Because it is a multiclass problem, we used multi_class = "ovr" option for Linear SVC. We also set tol=1e-5 as tolerance level or stopping criteria and the model will stop producing a better model when the tolerance criteria is met. For Naïve Bayes, we have used MultinomialNB() function because it is a multiclass problem. We have also set fit_prior = True, to allow our model learning from the prior probabilities.

4.2 Validation Setup

For the purpose of this project, we have taken train test split, which is also a type of training validation approach. In this case, we have taken 25000 random samples from training set with equal number of each sentiment in order to make the sample data balanced. Then we did a 80% and 20% train-test split for the random rows within this balanced data set. For the final prediction on test data, the entire 25000 sample dataset is used to train the model.

4.3 Accuracy Computation

For the accuracy computation we have taken we have taken the 'True Positive Rate' (in percentage) as precision rate for the entire predictions as well as the 'True Positive Rate' (in percentage) for each and every sentiment. The formula is mentioned below.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

(Wikipedia, 2017)

The sample accuracy for a model will look like below.

	precision	recall	f1-score	support
1	0.54	0.72	0.62	950
2	0.46	0.38	0.41	1016
3	0.46	0.45	0.45	1011
4	0.47	0.50	0.49	999
5	0.69	0.57	0.62	1024
accuracy			0.52	5000
macro avg	0.52	0.52	0.52	5000
weighted avg	0.52	0.52	0.52	5000

5 Experimental results

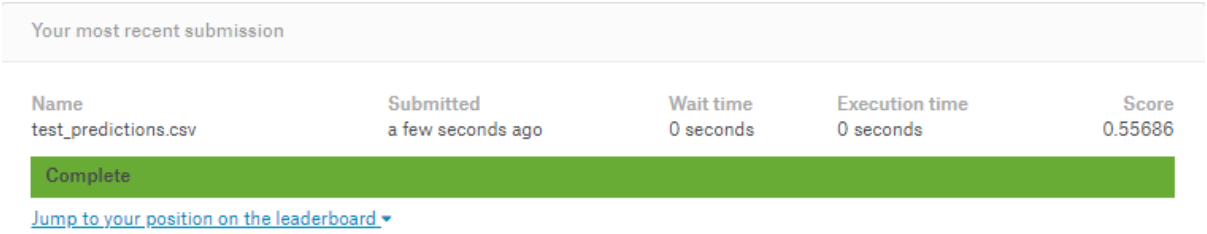
We have run the Linear SVC and Multinomial Naïve Bayes for the above said three features and the results are as follows.

Feature Set	Model	Accuracy
Feature 1	Linear SVC	Total Precision = 0.54 Precision for Sentiment 1 = 0.62 Precision for Sentiment 2 = 0.49 Precision for Sentiment 3 = 0.48 Precision for Sentiment 4 = 0.47 Precision for Sentiment 5 = 0.65
	Multinomial Naïve Bayes	Total Precision = 0.52 Precision for Sentiment 1 = 0.55 Precision for Sentiment 2 = 0.45 Precision for Sentiment 3 = 0.45 Precision for Sentiment 4 = 0.47 Precision for Sentiment 5 = 0.69
Feature 2	Linear SVC	Total Precision = 0.54 Precision for Sentiment 1 = 0.65 Precision for Sentiment 2 = 0.48 Precision for Sentiment 3 = 0.42 Precision for Sentiment 4 = 0.49 Precision for Sentiment 5 = 0.65
	Multinomial Naïve Bayes	Total Precision = 0.54 Precision for Sentiment 1 = 0.62 Precision for Sentiment 2 = 0.46 Precision for Sentiment 3 = 0.41 Precision for Sentiment 4 = 0.48 Precision for Sentiment 5 = 0.71
Feature 3	Linear SVC	Total Precision = 0.56 Precision for Sentiment 1 = 0.66 Precision for Sentiment 2 = 0.53 Precision for Sentiment 3 = 0.48 Precision for Sentiment 4 = 0.48 Precision for Sentiment 5 = 0.65
	Multinomial Naïve Bayes	Total Precision = 0.54 Precision for Sentiment 1 = 0.59 Precision for Sentiment 2 = 0.49 Precision for Sentiment 3 = 0.44 Precision for Sentiment 4 = 0.49 Precision for Sentiment 5 = 0.71

So, we can see from the above table that Linear SVC is in general working better than Multinomial Naïve Bayes.

6. Conclusion

So in conclusion we can say that **Linear SVC** is our final model based on the accuracy obtained from the above accuracy tables. Not only for the overall model precision, Linear SVC is predicting on higher precision for different sentiment level. The optimal feature set will be feature 3 in this case again on accuracy. So we accept **1734** raw text features including Unigrams and Bigrams. From the test data we found the final accuracy of our model = **0.55686**.



From this project, we learnt how to do sentiment analysis based on customer reviews. Additionally, we learnt raw text processing or text mining using Python, selecting features from them and finally building different classifiers with their accuracy measures. We also understood how to compare among different classification models and accordingly select the best one. We look forward to learn better text mining, feature selection techniques (LDA), in depth model tuning in future.

References

Alves, A. L., & Baptista, C. d. (2014). A Comparison of SVM Versus Naive-Bayes Techniques for Sentiment Analysis in Tweets: A Case Study with the 2013 FIFA Confederations Cup. *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, (pp. 123-130).

S. M. VOHRA, J. B. (2012). A COMPARATIVE STUDY OF SENTIMENT ANALYSIS TECHNIQUES. *JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN COMPUTER ENGINEERING*, 313-317.

stackoverflow.com. (2018, 11 03). *Replace apostrophe/short words in python*. Retrieved 06 05, 2019, from stackoverflow.com: https://stackoverflow.com/questions/43018030/replace-apostrophe-short-words-in-python

Vaghela, V. B., & Jadav, B. M. (2016). Analysis of Various Sentiment Classification Techniques. *International Journal of Computer Applications (0975 – 8887) Volume 140 – No.3*.

Vohra, S. M. (2012). A Comparative Study of Sentiment Analysis Techniques. *Journal of Information, Knowledge and Research in Computer Engineering*, 313-317.

Wikipedia. (2017, June). *Sensitivity and specificity*. Retrieved 06 06, 2019, from wikipedia.org: https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Zubrinic, K., Milicevic, M., & Zakarija, I. (2013). Comparison of Naive Bayes and SVM Classifiers in Categorization of Concept Maps. *INTERNATIONAL JOURNAL OF COMPUTERS Issue 3, Volume 7*.