Was the movie good enough?

COMP 551 - Mini Project 2

Ameya Bhope (260849407)   Surya Kumar Devarajan (260815492)
Paniz Bertsch(260862054)

February 2019

Abstract

The objective of the project was to explore the sentimental analysis of movie reviews as positive or negative. The reviews were taken from the popular website IMDb, where users review movies, TV series and internet streams and also rate them numerically between 1 and 10. The target of the project was to perform sentimental analysis on these reviews and determine whether the movie got a positive or a negative review. For this we tried an array of models along with many preprocessing techniques and found that Linear Support Vector Classifier model obtain the best accuracy. The project was in form of a in class competition on the Kaggle website. We achieved an accuracy of 91.575% on validation set and 90.26% on the Kaggle test set.

Keywords— Machine Learning, Sentiment Analysis, NLP, Text Classification

1 Introduction

In this project, we analyzed performance of multiple algorithms for predicting the sentiment of IMDb reviews and competed in a Kaggle competition with other groups to achieve the best accuracy. Our dataset consisted of 25000 test data, and 25000 labeled IMDb reviews (12500 positive and 12500 negative reviews) which we split into training and validation data randomly to predict the sentiment.

To build and select our best models, we used multiple classes from SciKit-learn library[1] for preprocessing of data and classification. We also, implemented a Bernoulli Naive Bayes model from scratch to compare its performance with other sophisticated classifiers and developed a model validation pipelines with binary countvectorizer and tf-idf vectorizer. After fine-tuning features and parameters, we found that Linear Support Vector classifier obtains the best accuracy in predicting the sentiment of test data with 91.575% accuracy on validation set.

In this report, we describe the work that has already been documented about sentiment analysis, then we describe the dataset, explore some insights about the data and also how the classification model was setup. Further we detail our approach in section 4 and then summarize our results in section 5. In section 6 we outline and discuss our key takeaways from the project and also propose some of the possible improvement techniques.

2 Related Work

With exponential increase in number of internet and social media users, and popularity of mobile devices in recent years, sentiment analysis has been a popular research area to gain insights into success or failure of products. Opinion mining and sentiment analysis is also used in a variety of areas such as health-care, tourism, sports, politics and financial sector[2]. One of the principle works for applying machine learning techniques to text classification problems and sentimental analysis is [3]. Many preprocessing, feature extraction and modeling algorithms have been developed and tested in recent years with different pros and cons, and time complexity. Among popular models, ANN (Artificial Neural Network) gives better accuracy in predicting sentiments of reviews but requires greater processing power while Support Vector

Machine is a faster, generalized classifier however its performance limited by kernel selection[4]. Naive Bayes is simple and easy to understand, and although makes independence assumption regarding the features it performs comparable to some of the most sophisticated models as seen in [5] and finally decision trees, while easy to visualize, suffer from low stability and high variance[6].

3 Dataset and Setup

The dataset used for this project is curated from IMDb. The dataset is split evenly with 25000 reviews deliberated for training the model and 25000 for testing. The training dataset containing 25000 reviews is comprised of 12500 positive sentiment instances and 12500 negative sentiment instances. Based on this data, we were supposed to train our model and predict the target variable *sentiment* for the 25000 movies in the test set. The data was provided in text files each with an unique id and they were in their parent folder indicating whether it is positive or negative instance. We split the training dataset further into validation set to evaluate the performance of our models. By experimenting, we found that taking 4000 samples in validation set out of the 25000 samples gives us the best results. This is done by using the scikit learn's libraries.

The movie reviews are messy real world data, containing colloquial language with lots of punctuations, slangs and also html tags. This data first needed to be processed. For this purpose we implemented an array of data cleaning techniques. Some of them were removing the punctuation marks, html tags along with prepositions and pronouns which don't necessarily affect the review significantly. Prior to this we replaced some words which are written in informal way with their root method of writing. e.g. Mustn't was replaced by must not.

Once this was done, we *tokenized* every sentence, to split every word as a separate vector. Further we *lemmetized* the data by which we obtain the root word of a particular word. e.g. Walking would be changed to it root word walk. This is achieved by using the convenient python library *NLTK* (Natural Language Took Kit)[7] and its interface *WordNet*, which also takes account of synonyms while doing this processing. Post this, it is passed through a *tf-idf* (Term Frequency - Inverse Document Frequency) analyzer, which calculates how important that particular word is with respect to the entire document in a corpus. We used Sci-Kit Learn's [1] CountVectorizer() class. While vectorizing the words, we take into account the n-grams parameter. It is a simple continuous sequence of n-grams. We found that with trigram (1,3), we achieve the best performance across all the models. Another parameter here is the maximum and minimum number of reviews in which a particular word appears. Optimal value of min_df and max_df are 2 and 0.9 respectively. We also used [1]'s TfidfTransformer() class in which we used norm of penalty as 'l2'. We also set the use_idf parameter as True to enable inverse document frequency re-weighting.

Figure 1 shows the various words that were most frequent in both positive as well as negative sentiment.

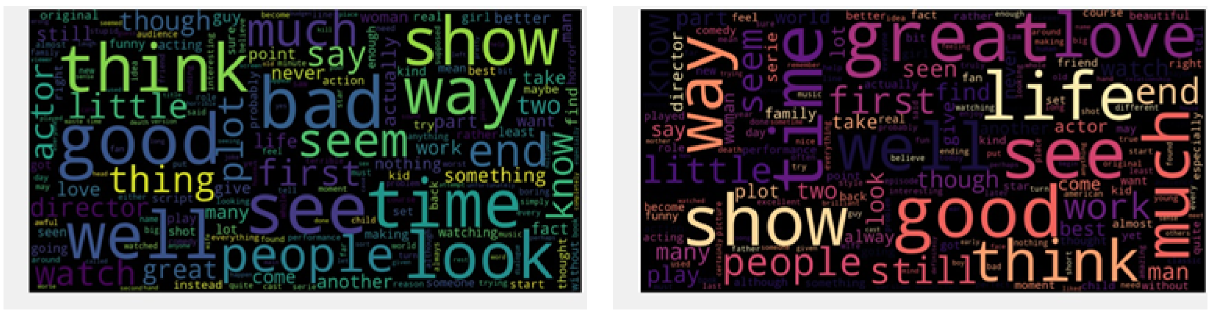

Figure 1: Wordcloud of positive (left) sentimental reviews and negative (right) sentimental reviews

4 Proposed Approach

After the preprocessing of data we experimented with a number of models with the objective of getting the best accuracy. For this purpose we firstly split our data into train and validation test, to compare the models we were implementing.

4.1 Implementation of the Bernoulli Naive Bayes Classifier

The Bernoulli Naive Bayes Classifier[8], which is one of the simplest probabilistic classifier which is based on the Bayesian Theorem, with an assumption that the features are naively independent was implemented from scratch. It takes into account the word count for the instances, matching the sentiment label. More the number of times the particular word has occurred in a sentiment label, greater will the probability that, if it appears in the test instance, the instance would belong to that particular label. It makes a strong naive assumption that the word count has no dependence over the other words. By making this assumption, the complexity is reduced significantly. We calculate the classifier delta value (log of probability), which if positive the instance has a positive sentiment and vice versa. We also implement *Laplace Smoothing*, for the classifier to have the capability to predict the outcome of words in the test set, which it has not encountered in the training examples.

4.2 Pipeline and Ensemble Methods

Pipeline gives a single interface for all steps of transformation and resulting estimator. We have used Binary occurrences and Tf-Idf vectorizer as the two different feature extraction pipelines for processing the text data and LinearSVC method for classification. We built a Model Validation Pipeline[9] using the above feature extraction and classifier. It encapsulates transformers and predictors inside and helps in finding the best estimator by performing a grid-search over the set of provided parameters. The steps involved in implementing were creating a pipeline for binary occurrences and tf-idf weighting with Linear SVC (Support Vector Classifier)[1]. Post this we wrapped these within a pipeline list and assigned parameters for both the feature extraction methods. Having these parameters also in a list we iterated over them to perform k-fold cross validation which was done by using Grid Search CV class, with k taken as 10, to find the best estimator, thus creating another pipeline.

We also used ensemble learning model in our project. Rather than having a single model and hoping for the best result, ensemble methods take a myriad of models into account, and average those models to produce one optimal predictive model. The ensemble learning technique is used to improve generalizability, robustness of the model over a single estimator. In this class of ensemble techniques we choose to Voting Classifier for our project. It combines different machine learning classifiers, and uses a majority vote technique to predict the target sentiments. We choose this particular class, as the participant models - LinearSVC, Logistic Regression, Ridge Regression were equally performing with accuracy scores in the high 90's. Thus, the VotingClassifier method balances out the individual weaknesses of these models.

5 Results

In this section, we make an analysis of the Naive Bayesian Classifier in closed form approach and compare its result with several sophisticated models. Finally, we report our results by executing the top performing model in the testing set on the in-class competition platform Kaggle.

We executed a variety of models varying the hyper-parameters (i.e. penalization norms, inverse of regularization strength - C, max iterations and other ones for optimization such as solver). We found that the out of the box models, with hyper-parameter tuning work really well. We obtained our best result from Linear Support Vector Classifier.

Our best classifier was Linear SVC model. This model outperformed all the other models we implemented. The penalty parameter for this model was chosen to 'l2' norm, which provides higher stability than 'l1' norm, which although provides robustness as the chance of occurance of extreme values is remote. The loss parameter was chosen as 'squared_hinged' loss. It denotes the maximum margin for

Model	Validation Accuracy	Time Taken
Naive Bayes (from scratch)*	87.275	11.3 seconds
Logistic Regression*	91.325	137 seconds
Linear Support Vector Classifier*	91.575	115 seconds
Ridge Classifier	91.475	288 seconds
Decision Tree	70.775	85.86 seconds
Multinomial Naive Bayes	89.775	13.62 seconds
Naive Bayes SVM	90.34	18 seconds
Stochasitic Gradient Descent	91.125	11.47 seconds
Ensemble Method	91.50	42.68 seconds

Table 1: List of models executed

classification. We tried various values of C which is the penalty parameter of the error term. By using these parameters we got our best accuracy of 90.26% on Kaggle.

Closely following the Linear SVC model, is the ensemble learning model. An ensemble learning model is trained in a way that it solves the same problem. This model was implemented in a way which uses the best predictor from a combination of 3 of the best individual performers - Linear SVC, Logistic Regression, Ridge Classifier. For all these individual k-fold cross validation was performed, ensuring that the model has got all the patterns from the dataset and its not picking up on too much noise, in other words its low on bias and variance.

Table 1 above shows the list models we executed[1]

Below in figure 2 show the graph of ROC curve and precision vs recall. The ROC curve is a plot of false positive rate (x-axis) and the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0 [10]. A good model assigns higher probability to a randomly chosen real positive occurrence than a negative occurrence on average. As seen in the figure, the curve bows up to the left of the curve, which satisfies the condition. The weighted average value of precision and recall (F1-score) was found to be 0.916, the area under curve was AUC=0.971, which demonstrates that the model can differentiate well between the label classes. The average precision was found to be 0.971. These statistics were obtained using sci-kit learn's libraries.

[2]

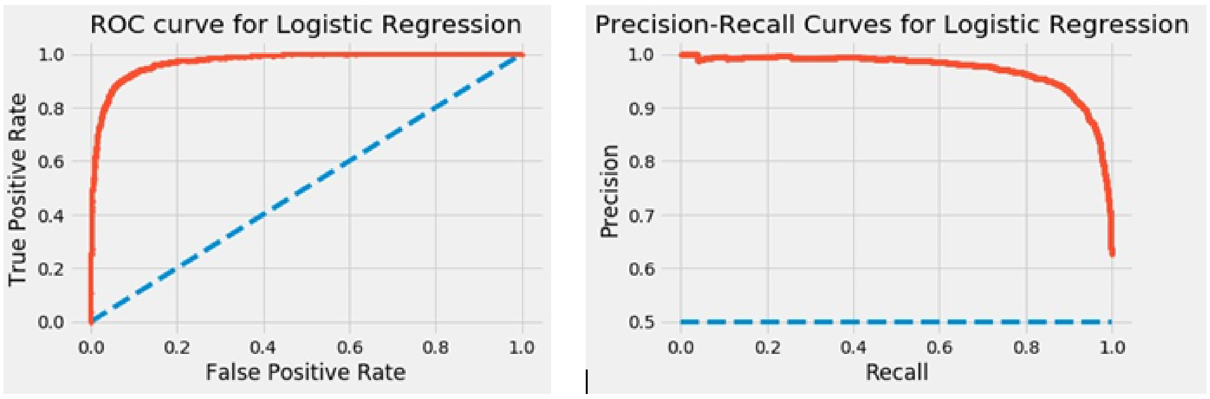

Figure 2: (Left) ROC curve summarizing the trade off between the true positive rate and false postive rate for LinearSVC model. (Right)Precision v/s recall graph for Logistic Regression

[1]The models with a '*' are the three models required in this project

[2]Some graphs and analyses were avoided because of the limitation of space, but they can be found in the Jupyter notebook provided

6 Discussion and Conclusion

In this project, we classified the sentiment of IMDb movie reviews using machine learning algorithms. We found that Linear Support Vector classifier obtains the best accuracy with an accuracy of 91.575% on validation set and 90.26% on the Kaggle test set. The ensemble learning method which combines the top 3 individual models gives the best performance based on the accuracy score and the runtime taken by it. In our experimentation we found that the tf-idf feature extraction with Linear SVC pipeline is better than the one with Binary occurrences. We also observed that the closed form of Bernoulli Naive Bayes Model performed only slightly inferior to the more sophisticated models. For decision trees, we found that it didn't give a good accuracy because of the large size of the dataset and as the tree grows, the generalization performance starts to degrade, as the algorithm starts including irrelevant attributes.

For future investigations, we believe RNN (Recurrent Neural Networks), LSTM (Long Short Term Memory) networks further improve the accuracy of our predictions in sentiment analysis, although their implementation will require significant computing power. Also, the accuracy maybe improved by using more sophisticated Natural Language Processing techniques like part of speech (POS) along with other lexicon based feature extraction and other noise elimination methods to improve our model further.

7 Statement of Contributions

We all three contributed to the project quite equally. We exchanged our ideas about the implementation of various tasks and also worked together to write the report.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Filipe R Lucini, Flavio S Fogliatto, Giovani JC da Silveira, Jeruza L Neyeloff, Michel J Anzanello, Ricardo de S Kuchenbecker, and Beatriz D Schaan. Text mining approach to predict hospital admissions using early medical records from the emergency department. *International journal of medical informatics*, 100:1–8, 2017.

[3] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[4] Rajesh Piryani, D Madhavi, and Vivek Kumar Singh. Analytical mapping of opinion mining and sentiment analysis research during 2000–2015. *Information Processing & Management*, 53(1):122–150, 2017.

[5] Tomas Pranckevičius and Virginijus Marcinkevičius. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221–232, 2017.

[6] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[7] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[8] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[9] Ryan Cranfill. Building a sentiment analysis pipeline in scikit-learn part 5: Parameter search with pipelines. 2016.

[10] Jason Brownlee. How and when to use roc curves and precision-recall curves for classification in python. 2018.