Sentiment Analysis of IMDB movie reviews

Manpreet Kaur
Northeastern University,
Boston, Massachusetts
lnu.m@husky.neu.edu

Sujith Srivardhan Arram
Northeastern University,
Boston, Massachusetts
arram.s@husky.neu.edu

Vipul Sharma
Northeastern University,
Boston, Massachusetts
sharma.vip@husky.neu.edu

Abstract

In this project we developed machine learning models that use movie reviews by users to classify the sentiment of reviews.

1 Introduction

More than 80% of the internet users [or more than 60% of American users] have done online research on a product at least once. Given the availability of a large volume of online review data (Amazon, IMDB, etc.), sentiment analysis becomes increasingly important. In this project, a sentiment classifier is built which evaluates the polarity of a piece of text being either positive or negative. Online review data has become available now in large volumes (from websites like Amazon, IMDB, etc.) With the constant growth in this data, there is a huge scope to develop or improve the quality of the resources that are being exchanged in day to day life. To explore more in depth, we decided our research should start at a point and then we decided to start the analysis on movie reviews to understand the quality of a movie a user would like to watch. We chose we can achieve this with the help of the movie reviews available on IMDB [International Movie Database] and perform sentiment analysis on them [positive or negative].

Our Naïve Bayes model trains and evaluates data into two different categories. Why movies? Movies are never ending. The data generated by movies is tremendous, and the need for movie reviews is ever growing as the number of movie goers only increase day by day. Also, movies are one of the most abundantly available resources online for entertainment enjoyed by active internet users. Why IMDB? We chose the platform carefully after a good amount of background work by considering various factors like diversity in the movies available [various languages and genres], number of users rating the movies, and range of movies available. At the end of the research, we should be able to extend our analysis to more domains available in the market, like shopping on e-commerce websites, buying properties on real estate websites, etc. Also, out paper should be able to stand as a base for classifying the reviews into various emotions like [happy, sad, drama, horror, etc.]

2 Related Work

Significant amount of research has been performed to date on analyzing online reviews. Combining sentiment analysis on online data and analysis on online reviews in general, there have been few good studies that generated good results. Few studies explored even analysis of audio and video reviews. A study on twitter data using SVM [10] gave an accuracy of 71%. A sentiment analysis was performed on twitter data posted by users was used to generate this result. They used a 10-fold cross validation for evaluating their results. Also, there was a different paper [11] which demonstrates that it is possible to pull significant information from textual information in a shallow network using deep neural networks, that performs sentiment analysis on IMDB movie reviews, dataset used from Kaggle, with an accuracy of 89%.

3 Approach

3.1 Dataset

In this project, we generated our own dataset by crawling the IMDB [1] website. We picked 20 highest rated and 20 lowest rated movies as the base list. From each movie in the base list, we performed a DFS crawl on the related movies content. We extracted 100 movies for each of the movies in the base list. This resulted in list of 1329 movies, after removing the duplicates.

We stored the movie reviews using their IMDB ids and accessed using a base URL. For each of the movies in the list, we extracted 20 reviews using the python Beautiful Soup package [2].

We stored all the reviews in a dictionary with the review as the key and the rating as the value. The average rating for all the reviews was 6.59. We marked the reviews with rating 1-6 as negative, 7 as neutral and 8-10 as positive. Finally, collected a total of 6000 positive and 6000 negative reviews.

3.2 Preprocessing Data

In the first step, we are converting the entire dataset [all the files in the corpus] to lowercase using the string library available on python. Then, generating a list of stop words that account to 2% of total collection of the unigrams [using the term frequencies to generate this] and removing the stop words from the corpus. Later converting all the stem words to their root words using the porter stemmer library. From the obtained. No, removing all the punctuations from the obtained corpus. As a final step, we are removing the stop words from the obtained corpus. Running the process twice-once for unigrams generation and the other time for bigrams generation. Bigrams are appended using a '_' between two terms. The entire dataset accounts to up to 40MB, with an average of 1.6KB per review [stored in individual files] containing only alphabets or spaces ['_' in case of bigrams].

3.3 Models

Feature selection

The first feature selection we used was using unigrams. We followed the bag of words model. All the words in the reviews were treated as the features.

The second feature selection was using bigrams. We created bigrams for all the reviews and followed the bag of words model for the bigrams. All the bigrams in the reviews were features for the classifier.

We also used the Term Frequency-Inverse Document Frequency (TF-IDF). The word frequency of each review is weighted inversely by the number of reviews the word appears in. Hence, the word that appears frequently, which may not have much contribution for the information, will have small weight.

Naïve Bayes

We are dividing the dataset in two parts – 90% training data and 10% test data.

We are using two class classification – positive and negative for the movie reviews. For the classification, we have implemented the Naïve Bayes classifier. It is a probabilistic classifier which is based on Bayes theorem and assumes independence between the features. It computes the posterior probability of both the classes for each review and chooses the classes with the highest probability.

Training the classifier based on [5]. Let V be the vocabulary of all words in the documents in D. For each class $c_i \in C$,

$P(c_i) = |D_i| / |D|$

Let ni be the total number of word occurrences in all the documents of class c_i. Let w_j be each word in the vocabulary, n_{ij} be the number of occurrences of w_j in all the documents of class c_i,

$P(w_i|c_i) = (n_{ij} + 1) / (n_i + |V|)$

For prediction, naïve bayes classifier assigns the class which has the highest posterior probability.

We used the python scikit-learn library's metrics class [7] calculate the accuracy, precision, recall and the F1 score of the predictions generated by the Naïve Bayes classifier for different feature sets.

Accuracy	84.59%
Precision	84.95%
Recall	80.03%
F1	82.42%

Table 2: Result of Naïve Bayes with unigram features

Accuracy	78%
Precision	79.35%
Recall	74.75%
F1	76.98%

Table 3: Result of Naïve Bayes with bigram features

Accuracy	83.22%
Precision	85.62%
Recall	81.65%
F1	83.59%

Table 4: Result of Naïve Bayes with unigram TFIDF features

4 Evaluation

The evaluation of our implementation is done by comparing the efficiency of the algorithm through our implementation with the standard library implementation. A Support Vector Machine (SVM) was also done to give a better overview.

The evaluation process was divided into two sub-tasks: feature extraction process and training the classifier. We trained the Support Vector Machine (SVM) classifier and Multinomial Naïve Bayes classifier on tf-idf weighted word frequency features. Finally, we analyzed the effect of using this scheme while checking the performance of the trained model on the test movie reviews files.

Both the classifiers were run against the simple word count approach and the tf-idf weighted matrix. A confusion matrix is built at the end for comparison of the results. The best performing model was SVM classifier implemented using sklearn library.

4.1 Naïve Bayes

As previously mentioned we are using the 10-fold validation approach, where the dataset is divided in two parts – 90% for training and 10% for test data. For evaluating we used Multinomial Naïve Bayes algorithm for or implementation.

The results below show the number of text files whose sentiment were correctly identified when predicted by classifiers.

n = 12000	Predicted: Positive	Predicted: Negative
Actual: Positive	4980	1020
Actual: Negative	1074	4926

Accuracy: 82.55 % True Positives: 9906

Table 5: MN Naïve Bayes (scikit implementation)

4.2 SVM

The SVM classifier was run and trained against the k-fold validation too, for the two approaches: conventional word count and tf-idf weighted factor input.

We evaluated the prediction performance of our models by computing the prediction accuracies and the confusion matrices. The tf-idf based model performed better than all the algorithms used for implementation and evaluation.

n = 12000	Predicted: Positive	Predicted: Negative
Actual: Positive	5184	816
Actual: Negative	750	5250

Accuracy: 86.95 % True Positives: 10434

Table 6: SVM (Tf-Idf weighted factor)

5. Conclusion

Overall the SVM model performed slightly better than our implementation of Naïve Bayes model. From our custom implementation results in Table 7, we can infer that our Naïve Bayes model is better at predicting the sentiment of the movie review than the implementation using scikit; at an accuracy of 84.59%.

n = 12000	Predicted: Positive	Predicted: Negative
Actual: Positive	4803	5348
Actual: Negative	1197	652

Accuracy: 84.59 % True Positives: 10,151

Table 7: Our Implementation (Naïve Bayes)

Acknowledgement

We would like to thank Dr. Stacy Marsella for guiding us on how to approach this problem and giving us a deep insight, and for advising us to use bigrams for better covering a broader analysis. We would also like to thank Sklearn Library, Pandas as well as the tutorial code available on their site http://scikit-learn.org/stable/ and https://pandas.pydata.org/ was used majorly for evaluating our project.

Srivardhan worked on the creation of dataset by crawling IMDB and generating unigrams and bigrams out of them. Manpreet implemented the Naïve Bayes algorithm, and assisted in evaluation calculating the F1 score. Vipul wrote the code for evaluation: implementing MN Naïve Bayes and SVM and creating the confusion matrices for the evaluation.

This project report was a collaborative effort. Manpreet worked on the approach and the reference section, Srivardhan on the introduction and related works, and Vipul on the evaluation, acknowledgement and document formatting.

References

[1] [Needham.1990]. Col Needham *Internet Movie Database (IMDB)*, 1990.

[2] [Richardson.2012]. Leonard Richardson python library *BeautifulSoup4*, 1990.

[3] [Loper et al. 2002]. Edward Loper, Steven *Bird NLTK: The Natural Language Toolkit Library. Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1: 63-70*, 2011

[4] [Porter.1980]. Martin F. Porter *An algorithm for suffix stripping*, 1980.

[5] [Mausam et al. 2011] Mausam, Dan Weld, Prabhakar Raghavan, Hinrich Schutze, Guillaume, Obozinski, David D. Lewis *Text Categorization using Naïve Bayes*, 2011.

[6] [McCallum et al. 1998] Andrew McCallum, Kamal Nigam *A Comparison of Event Models for Naive Bayes Text Classification*, 1998

[7] [Buitinck et al., 2013] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt and Gael Varoquaux *API design for machine learning software: experiences from the scikit-learn project: 108-122, September* 2013

[8] [Pang et al. 2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan *Thumbs Up?: Sentiment Classification Using Machine Learning Techniques: 79-86*, 2002

[9] [Pang et al. 2008] Bo Pang and Lillian Lee *Opinion Mining and Sentiment Analysis: 1-135*, January 2008

[10] [Procedia 2013] Samad Hasan Basari, Burairah Hussin, Gede Pramudya, and Junta Zeniarja: *Opinion Mining of Movie Review using Hybrid Method of Support Vector Machine and Particle Swarm Optimization*

[11] Alec Yenter, Abhishek Verma: Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. *In Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017 IEEE 8th Annual*