# Sentiment Analysis and Rating Prediction for Movie Reviews

Ashok Marannan, Kavin Mani, Kirthanaa Raghuraman
Department of Computer Sciences
University of Wisconsin - Madison

*Abstract*— **Sentiment analysis of movie reviews is one of the most widely discussed topics in machine learning. With the improvement in internet connectivity, millions of people around the world are gaining access to the internet and post reviews/ratings in popular sites. Analyzing such reviews/ratings and the factors that affect the ratings often provide interesting insights. In this paper, we analyze the performance of various classifiers on predicting the polarity and ratings of movie reviews. We also analyze the impact of various features on classifier performance.**

***Keywords— Sentiment Analysis; Polarity Identification; Machine Learning;***

## I. INTRODUCTION

Sentiment analysis is the process of identifying subjective information in source materials like the sentiment in textual documents as positive and negative. Sentiment analysis is widely used in analyzing customer feedbacks, reviews and people's sentiments in social media. It has a wide variety of applications like marketing, customer service etc.

Sentiment analysis systems are used in almost every domain because opinions are central to almost all human activities. They are key influencers of our behaviors. Sentiment analysis uses natural language processing and text analysis to identify and extract information about a particular field of interest. Due to the popularity of the social media such as blogs and social networking sites such as Facebook, Twitter etc. the interest in sentiment analysis has increased to a higher extent.

There are several challenges in Sentiment analysis. The first is that an opinion word that is considered to be positive in one situation may be considered negative in another situation. The second challenge is that people don't always express opinions in the same way. The usual text processing relies on the fact that small differences between two pieces of text don't change the meaning very much.

The unstructured textual data on the internet often carries expression of opinions of users. Sentiment analysis tries to identify the expressions of opinion and mood of writers. A simple sentiment analysis algorithm attempts to classify a document as 'positive' or 'negative', based on the opinion expressed in it. The document-level sentiment analysis problem is essentially as follows: Given a set of documents D, a sentiment analysis algorithm classifies each document $d \in D$ into one of the two classes, positive and negative. Positive label denotes that the document d expresses a positive opinion and negative label means that d expresses a negative opinion of the user. More sophisticated algorithms try to identify the sentiment at sentence-level, feature-level or entity-level.

Sentiment analysis helps to find words that indicate sentiment and helps to understand the relationship between textual reviews and the consequences of those reviews. One such example is predicting online movie review ratings. A movie has many different aspects such as direction, screenplay, cast, runtime, story etc. and the reviewer may tend to give his/her opinion based on these.

The project starts with extracting data from the website. Reviews are collected on IMDb (http://www.imdb.com/). The second step is to apply sentiment classification using TF-IDF approach. This involves text preprocessing, text transformation, validating feature effectiveness using classifiers and sentiment classification.

The goal of the system described in this paper is to compare the effectiveness and performance of various traditional and new algorithms for predicting polarity and rating for movie reviews.

The rest of the paper provides the following details: Section II explains the approach, Section III discusses the experimental results, Section IV gives the limitations and Section V gives the conclusions of our work.

## II APPROACH

*a) Dataset*

We have used Stanford Large Movie Review Dataset[1] for conducting the experiments. The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets. In the unsupervised set, reviews of any rating are included and there are an even number of reviews > 5 and <= 5.

*b) Additional Features from IMDb:*

We wanted to see how other features like director, cast, genre, language etc. would affect the movie rating. So we crawled IMDb using the Scrapy Python library [4] and extracted the following features for each movie whose reviews are part of the data set:

| Feature | Type |
|---|---|
| Title | Nominal |
| Director | Nominal |
| Writer | Nominal |
| Lead star | Nominal |
| Genre | Nominal |
| Plot | Nominal |
| Run time | Numeric |
| Language | Nominal |
| Certification | Nominal |
| Country | Nominal |
| Box office collection | Numeric |
| Budget | Numeric |
| Opening collection | Numeric |
| Number of nominations | Numeric |
| Number of awards won | Numeric |

**Table 1 List of features and types crawled from IMDb**

*c) Pre Processing*

The data set containing the reviews cannot be used directly for classification. We had to do a lot of pre processing to ensure that the format of data is right and that the values are cleaned.

*Removing tags and other special characters* – The reviews had a lot of html tags (<br/> etc.) and some special characters that needed to be removed. We applied extraction methods such as regular expression and removed them.

*Missing Values* - For the attributes we crawled from IMDb, we replaced the missing values with the average of the feature values for numeric features and 'NULL' for nominal features.

*Normalization* - We normalized the numeric feature values using the following formula:

$$x' = \frac{x_i - \mu}{\sigma}$$

where $x'$ is the normalized value of instance i

μ is the mean of values for the feature

σ is the standard deviation of the feature

Removing stop words – We used the Python Stop – words package [3 to obtain the dictionary of stop words. We then processed the reviews by removing all the stop words.

*Stemming* – We used the Lovins stemmer provided in WEKA[1] to stem the words to the root words.

*TF-IDF transformation* – In order to obtain the term document matrix[6] of the word vectors in the movie reviews, we used the TF-IDF transformation feature of WEKA[2]. We also removed words whose document frequency is less than 5.

*d) Classification*

*1) Naïve Bayes Classifier:* Naïve Bayes classifier is the probabilistic classifier based on the Bayes theorem. Naïve Bayes classifier assumes independence between the features. Naïve Bayes classifier is one of the most powerful classifier and is widely used for a variety of problems. In practice it has been shown to perform well even when the independence assumption is not followed in the data. We used the implementation provided by WEKA [2] for this.

*2) J48 Classifier:* Decision tree learner is a predictive model that that maps an observation about a data sample to the conclusion about data label by using a decision tree. In decision tree, leaves represent class labels, internal nodes represent features and the branches represent the value of the feature in the internal nodes. Decision tree classifier is simple and comprehensible to human understanding. WEKA [2] provides J48 classifier which is the java implementation of the popular C4.5 decision tree algorithm.

*3) Random Forest Classifier:* Random Forests are ensemble learning methods used for machine learning tasks like classification, regression, etc. Ensemble learning is a technique for combining many weak learners to produce a strong learner. Random forests operate by constructing a multitude of decision trees at training time and outputting the majority label. We used the implementation provided by WEKA [2] for this.

*4) Logistic Regression:* Logistic regression classifier is a direct probability model that is used to predict a binary response based on one or more features. The probabilities describing the possible outcomes of a single trial are modeled as a function of features using a logistic function. Logistic regression is a special case of linear model and is similar to linear regression. However, underlying assumptions for logistic regression are different from the linear regression. The conditional distribution in logistic regression is Bernoulli distribution while in linear regression, it is Gaussian distribution. Also, the logistic regression predicts the probability of an instance being positive. We used the implementation provided by WEKA [2] for this.

*5) Linear Regression:* Linear regression is an approach for modeling the relationship between a scalar dependent variable *y* and one or more explanatory variables (or independent variables) denoted *X*. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. We used the WEKA implementation of L1- regularized linear regression to predict the ratings of movie reviews.

*6) Support Vector Machine:* Support Vector Machine is a non-probabilistic binary classifier. In essence the support vector machine learns a maximum margin separating hyper- plane for classification. For data that is not linearly separable, there are two approaches to learn the hyper plane, linear SVM with slack variables or kernel based SVM. For our data set we tried linear SVM model using Liblinear [5]. Liblinear provides implementation of linear SVM parameter estimation using coordinate descent.

7) Multilayer Perceptron: A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable. We used the implementation provided by WEKA [2] for this.

*e) Experimental Setup*

We divided our experiments into two parts:

**Polarity Identification** – We ran experiments with different combinations of feature sets to predict the polarity of the movie reviews using the classifiers mentioned above (except linear regression).
The feature sets considered during each experiment are listed as follows:

| Experiment | Features Considered |
|---|---|
| Polarity_ex1 | Features extracted from movie reviews. |
| Polarity_ex2 | Features extracted from movie reviews, features crawled from IMDb. |
| Polarity_ex3 | Features extracted from movie reviews, features crawled from IMDb and ratings for each movie. |

**Table 2 Features considered for each experiment in Polarity Identification**

**Rating prediction** - We ran experiments with different combinations of feature sets to predict the polarity of the movie reviews using the classifiers mentioned above (except logistic regression). For classifiers that required the class labels to be nominal, we converted the rating to a nominal attribute with 10 distinct values from 1 – 10.
The feature sets considered during each experiment are listed as follows:

| Experiment | Features Considered |
|---|---|
| Rating_ex1 | Features extracted from movie reviews. |
| Rating_ex2 | Features extracted from movie reviews, features crawled from IMDb. |
| Rating_ex3 | Features extracted from movie reviews, features crawled from IMDb and polarity of review for each movie. |

**Table 3 Features considered for each experiment in Rating Prediction**

10-fold cross validation was used to evaluate the performance of all the classifiers.

## III. Experimental Results And Discussions

*Experimental Results for Polarity Identification:*

| *Experiment* | *Classifier* | *Accuracy in %* | *Precision* | *Recall* |
|---|---|---|---|---|
| Polarity_ex1 | Naive Bayes | 75.1803 | 0.760 | 0.758 |
| | J48 | 69.4532 | 0.714 | 0.710 |
| | Random Forest | 77.2341 | 0.762 | 0.754 |
| | Logistic Regression | 77.2452 | 0.781 | 0.777 |
| | Multi Layer Perceptron | 75.7436 | 0.76 | 0.757 |
| | SVM | 83.4214 | 0.823 | 0.821 |
| Polarity_ex2 | Naive Bayes | 77.1432 | 0.755 | 0.752 |
| | J48 | 72.2345 | 0.691 | 0.690 |
| | Random Forest | 78.1008 | 0.761 | 0.760 |
| | Logistic Regression | 78.2432 | 0.782 | 0.780 |
| | Multi Layer Perceptron | 77.046 | 0.755 | 0.750 |
| | SVM | 84.321 | 0.810 | 0.819 |
| Polarity_ex3 | Naive Bayes | 78.1714 | 0.785 | 0.782 |
| | J48 | 70.1982 | 0.703 | 0.702 |
| | Random Forest | 79.1078 | 0.772 | 0.771 |
| | Logistic Regression | 79.23378 | 0.783 | 0.782 |
| | Multi Layer Perceptron | 79.7436 | 0.79 | 0.797 |
| | SVM | 85.321 | 0.853 | 0.851 |

**Table 4 Accuracy, Precision and Recall metrics for various classifiers for Polarity Identification**

*Experimental Results for Rating Prediction:*

| *Experiment* | *Classifier* | *Accuracy in %* | *Precision* | *Recall* |
|---|---|---|---|---|
| Rating_ex1 | Naive Bayes | 69.1734 | 0.691 | 0.690 |
| | J48 | 65.1632 | 0.663 | 0.650 |
| | Random Forest | 66.1098 | 0.682 | 0.661 |
| | Linear Regression | 69.2337 | 0.692 | 0.670 |
| | Multi Layer Perceptron | 70.4521 | 0.701 | 0.691 |
| | SVM | 70.9210 | 0.799 | 0.785 |
| Rating_ex2 | Naive Bayes | 70.1444 | 0.733 | 0.731 |
| | J48 | 67.4684 | 0.671 | 0.670 |
| | Random Forest | 67.1078 | 0.673 | 0.671 |
| | Linear Regression | 70.2353 | 0.707 | 0.702 |
| | Multi Layer Perceptron | 72.523 | 0.729 | 0.700 |
| | SVM | 74.4340 | 0.743 | 0.733 |
| Rating_ex3 | Naive Bayes | 71.1444 | 0.709 | 0.702 |
| | J48 | 67.4242 | 0.677 | 0.670 |
| | Random Forest | 70.1798 | 0.704 | 0.701 |
| | Linear Regression | 74.4325 | 0.743 | 0.742 |
| | Multi Layer Perceptron | 75.5453 | 0.755 | 0.750 |
| | SVM | 76.3343 | 0.764 | 0.671 |

**Table 5 Accuracy, Precision and Recall metrics for various classifiers for Rating Prediction**

We have evaluated the performance of the classifiers based on 3 measures namely accuracy, precision and recall. Accuracy is the percentage of classifications made correctly. Figure 1 shows the accuracy of different classifiers for Polarity Identification for the 3 experimental settings.

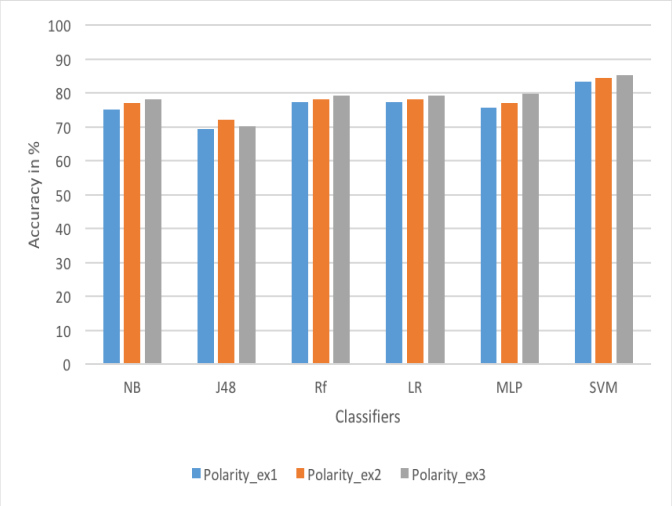


**Fig 1 Accuracy comparison of classifiers for 3 experimental settings for Polarity Identification**

From the Figure 1 and Table 4, it is evident that linear SVM was consistently having a higher accuracy when compared to other models. For polarity Identification, the average accuracy of classifiers for Polarity_ex1 was lesser than that for Polarity_ex2. The average accuracy of all classifiers for Polarity_ex3 was higher than the other two. This was expected since the key feature of 'rating' will strongly discriminate the polarity.

Figure 2 shows the accuracy of different classifiers for Rating Prediction for the 3 experimental settings.

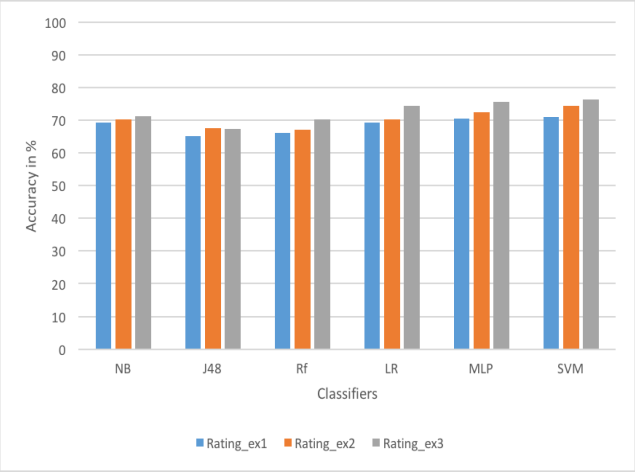


**Fig 2 Accuracy comparison of classifiers for 3 experimental settings for Rating Prediction**

As for rating prediction, from the Figure 2 and Table 5, both linear SVM and linear regression performed better than other classifiers. The average accuracies of Rating experiments mirrored that of Polarity experiments, though the average accuracy of classifiers for Rating_ex3 was a bit lower than the Polarity_ex3.

Figure 3 shows the average accuracy of the classifiers for the three experimental settings for both polarity Identification and rating prediction respectively.

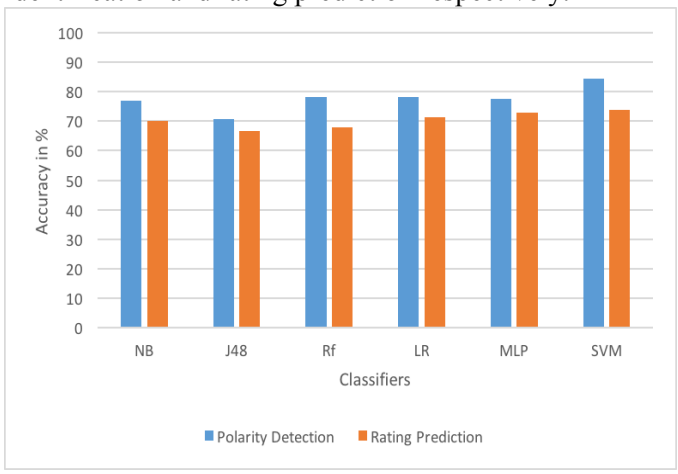


**Fig 3 Average Accuracy comparison of classifiers for for Polarity Identification and Rating Prediction**

From Figure 3, we can infer that average accuracy of classifiers in polarity Identification exceeds the average accuracy of the same classifiers for rating prediction. Also, in all the experimental settings, the J48 decision tree classifier performed poorly.

Figure 4 and Figure 5 show the learning curve of SVM classifier for Polarity Identification and Rating Prediction respectively.

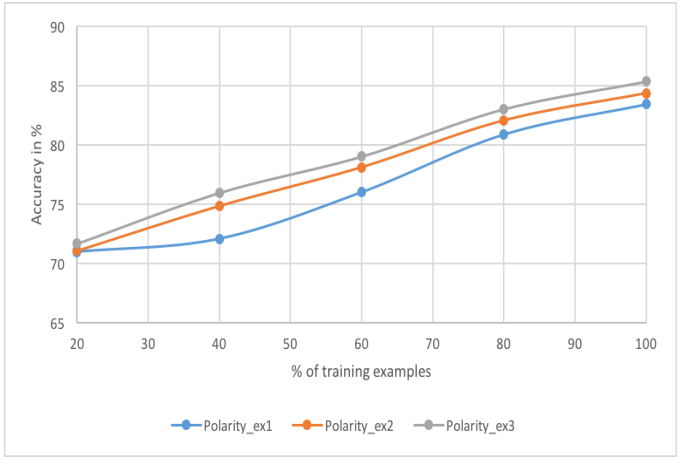


**Fig 4 Learning curve of SVM for Polarity Identification**

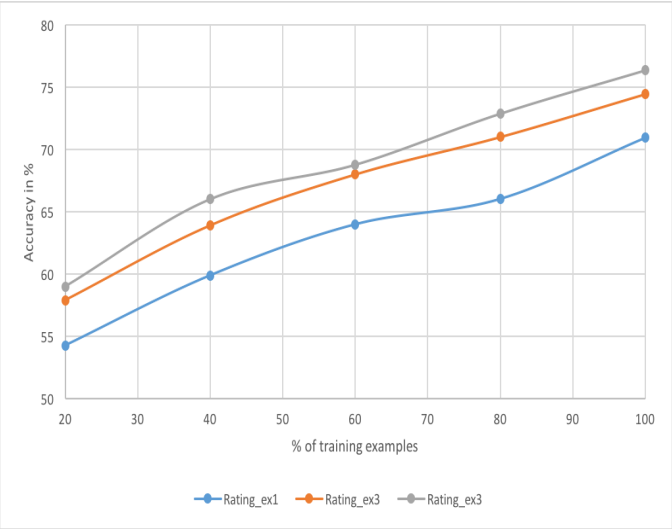


**Fig 5 Learning curve of SVM for Rating Prediction**

From Figures 4 and 5, we can infer that the accuracy increases with increase in training set size.

## V. Conclusion

In this paper, we analyzed the performance of various classifiers for polarity identification and rating prediction. We also analyzed how the accuracy increases when we considered additional features like director, cast, genre, runtime, certification, language, plot etc. We found that SVM outperformed the other classifiers in most of the experimental settings. Also, we found that some classifiers like Naïve Bayes did not improve in their classification accuracy by a huge margin even after the addition of these extra features.

## VI. Acknowledgement

We have benefitted immensely from the open source WEKA project by University of Waikato and libLinear library by National Taiwan University. We also are grateful to Stanford University for aggregating the reviews in proper format along with the ratings.

We would like to thank Professor Mark Craven for his valuable guidance. The materials covered in the lecture gave us very good insight about machine learning concepts that helped us to complete this project.

## References

[1] Stanford Large Movie Review Dataset - http://ai.stanford.edu/~amaas/data/sentiment/x

[2] WEKA - http://www.cs.waikato.ac.nz/ml/weka/

[3] Python Stop Words List - https://pypi.python.org/pypi/stop-words/2014.5.26

[4] Scrapy web crawler - http://scrapy.org/

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/ cjlin/liblinear

[6] Christian S. Perone, "Machine Learning: Text feature extraction (tf-idf)", http://pyevolve.sourceforge.net/wordpress/?p=1589

[7] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*