Predicting Star Ratings of Movie Review Comments

Aju Thalappillil Scaria (ajuts)
Rose Marie Philip (rosep)
Sagar V Mehta (svmehta)

1. Introduction

The growth of the World Wide Web has resulted in troves of reviews for products we wish to purchase, destinations we may want to travel to, and decisions we make on a day to day basis. Using machine learning techniques to infer the polarity of text-based comments is of great importance in this age of information. While there have been many successful attempts at binary sentiment analysis of text based content, fewer attempts have been made to classify texts into more granular sentiment classes. The aim of this project is to predict the star rating of a user's comment about a movie on a scale of 1, 2, 3 … to 10. Though a user self-classifies a comment on a scale of 1 to 10 in our dataset from IMDB, this research can potentially be applied to settings where a comment about a movie is available, but no corresponding star rating is (i.e. the sentiment expressed in tweets about a movie). We also explore if analyzing movie metadata (like director, actors, budget, production house etc.) and using it to augment the model built only on review text would improve the performance of the algorithm.

2. Data collection

A data set containing text-based movie review comments from IMDB and the star rating corresponding to each comment was obtained from http://ai.stanford.edu/~amaas/data/sentiment. This dataset contained reviews which had star ratings between 1-4 and 7-10 stars as it was used for binary sentiment classification. We extended the dataset with review comments having 5 and 6 star ratings from IMDB (The number of training samples per class that were added was in proportion to the existing dataset). To augment the star prediction algorithm, the following movie metadata was also collected – director name, actor names, revenue, budget, production company, number of votes on IMDB, genre, and opening week revenue.

3. Text classification from user comments

The dataset we obtained had a default dictionary containing all the words that appeared in the training samples. This dictionary had around 89500 words. Since the performance of classification algorithms on such a huge dictionary was relatively poor, we evaluated the following strategies to reduce the dictionary size:

- Eliminating stop words: Common words (like a, an, and, are, as etc.) appear with a high frequency in comments, but add very little value to the classification problem as they do not have any specific polarity.
- Identifying negation of context: The usage of terms likes 'not good', 'not interesting', 'wasn't interesting' etc. have a negative connotation. Revised dictionary combined these negation bigrams into a single feature.
- Stemming: Stemming condenses the dictionary by identifying root words.
- Eliminating features with low information: The ratio of number of occurrences of each word in positive comments (7, 8, 9, 10 star ratings) and negative comments (1, 2, 3, 4 ratings) were counted and those words with $positive\ count/negative\ count$ > 2 or $negative\ count/positive\ count$ > 2 were selected.

3.1 Models evaluated

i. **Naive Bayes** – Naive Bayes with multinomial event model was trained with a default dictionary and it served as a baseline for the project. We evaluated several options to enhance the algorithm as described in the feature selection section.

ii. **Multiclass SVM** – Apart from Naive Bayes, we used SVM in order to determine the ratings for a given movie comment. SVM is traditionally a binary classifier; however there exists methods to adapt it to the multiclass setting. The two methods used in this study were one-vs.-all and one-vs.-one using a linear kernel and the same bag of words frequencies from Naive Bayes as our features. All SVM models were generated using the liblinear package.

Suppose we have k classes $C_1, C_2, \dots, C_k$, the one-vs.-all method trains k models. Each model trains a binary classifier considering all points in a class C_i as positive examples and all points in the remaining $k-1$ classes $C_k, k \neq j$, as negative examples. During testing, each sample is tested against each of these models, and the class C_i which creates the greatest separation from the decision boundary for that data point is chosen as its class.

An alternative method is one-vs.-one. Here $k * (k-1)/2$ models are trained and during testing, each sample is tested against each other. Concretely, for the case of IMDB ratings on a scale of 1-10 we train 45 SVMs, one for each pair of ratings. Each time a test point is placed into a class k, it is considered a vote for that class. The training sample is then classified as belonging to the class with the most votes (or in the case of ties, the mean of the star ratings of the classes rounded to the nearest class).

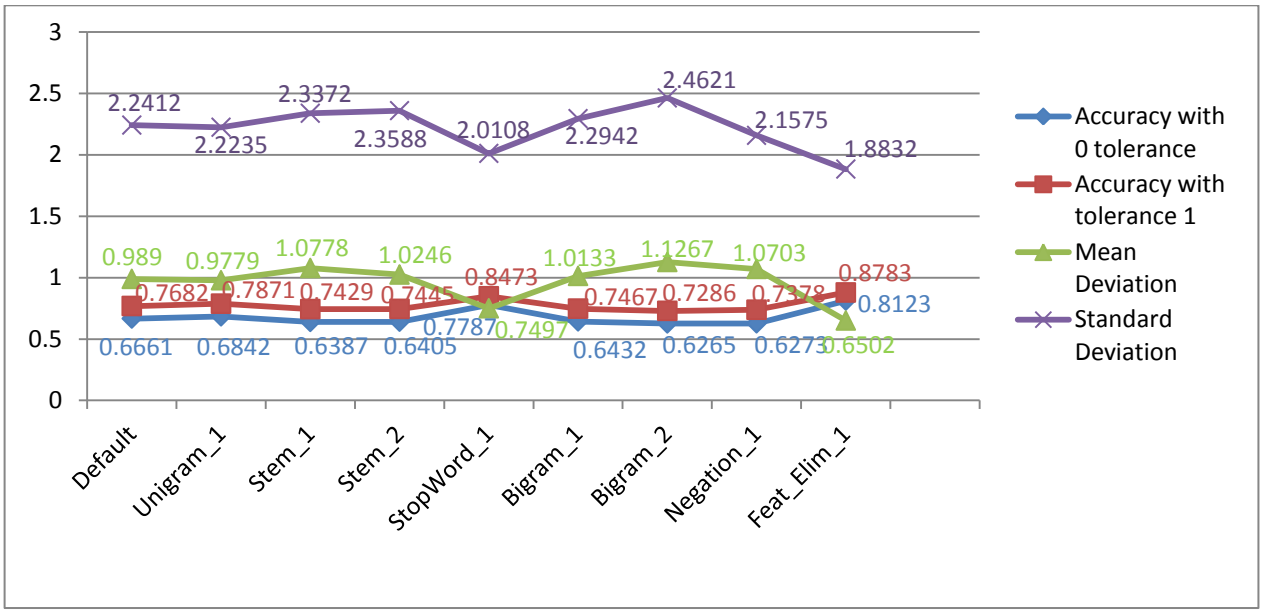
3.2 Features evaluated

Naïve Bayes and SVM were evaluated with different dictionaries to identify the best feature set (features being words):

- Default: The default dictionary that came with the dataset.
- Unigram_1: Default dictionary with cleaned up words from default dictionary.
- Stem_1: Dictionary with stemmed words.
- Stem_2: Dictionary with stemmed words by treating degree differentiators like 'er' and 'est' separate from root words.
- StopWord_1: Dictionary used in Unigram _1 with stop words removed.
- Bigram_1, 2: Dictionaries used in Unigram _1, Stem_1 augmented with top 500 bigrams, respectively.
- Negation_1: Dictionary used in Unigram_1 augmented with negated words.
- Feat_Elim_1: Dictionary in Unigram_1 after removing words having low information.

3.3 Feature and model selection

Feat_Elim_1 gave the best results on training and cross validating using Naïve Bayes classifier. 20% of the training data was used for cross validation. The metrics obtained are as follows ('Accuracy with 0 tolerance' is the fraction of cross validation samples whose star ratings were predicted exactly correct. Mean deviation is the average absolute value deviation of predicted star rating from the actual star rating of comments.)



The best cross validation metrics obtained after training SVM with the different feature sets is as follows:

Method	Accuracy with 0 tolerance	Accuracy with tolerance 1	Average Deviation	Standard Deviation
One vs. All	28%	56%	1.89	2.77
One vs. One	20%	42%	3.1	3.9

As evident from the table above, both the multiclass methods provided worse results than the original Naïve Bayes approach. In an attempt to improve the result, we tried various parameters for the L2-normalization constant and also

tried to modify the feature set (i.e. different settings for stop words, stemming, etc.); however this did not prove to be useful.

3.4 Analysis of cross validation results

i. **Naïve Bayes:** Word stemming reduced performance. This is because different tenses/usages of word have different semantics attached. For example, the word 'recommend' can be used in both positive and negative ways like in 'I do not recommend the movie' and 'I recommend the movie'; but, the word 'recommended' can be used only in positive context like 'I recommended the movie' and not like 'I did not recommended the movie'. Removing stop words improved performance by around 10%. Identifying negation of context did not improve performance. This may be because since there are 5 different classes each within positive and negative comments, the negation of context couldn't accurately capture the difference in usages of words like 'not' in comments that have similar ratings (either positive or negative). Eliminating words that have very less information resulted in improvement in performance by around 4%.

ii. **Multiclass SVM:** An interesting observation is that the one-vs.-one with voting performed considerably worse than the one-vs.-all method. This is possibly due to the fact that the voting model used was too simplistic. In particular, in simply taking the frequency of a classification we ignore the actual separation between the test data point and the decision boundary. A better approach would have been to have a probabilistic interpretation for the actual margin created between the test point and the decision boundary for each of the SVMs. This would allow for a better estimate of the posterior probability that the point belongs to that class. In contrast the one-vs.-all method does pick the point which has the greatest margin, which could explain why it performed significantly better.

3.5 Test Result

Naïve Bayes gave better results as compared to SVM, hence it was chosen to predict star rating of a user comment on the test data. The performance is as below:

Accuracy with 0 tolerance	Accuracy with tolerance 1	Mean Deviation	Standard deviation
52.61%	67.82%	1.1934	2.5342

4. Movie Star prediction from metadata

This section attempts to recognize how movie metadata affects individual user comments. Our intuition was that even though the text review would reflect a person's opinion of the movie, his rating might be influenced by factors like the general opinion of the public. We try to incorporate this into our prediction by including the average movie rating (similar to the one predicted by IMBD), which is a general indicator of public opinion towards a movie. Since the average rating might not be available for all movies, we try to predict this based on the metadata available, like gross revenue, number of votes on IMDB, director, genre, production company, opening weekend revenue, budget, and main actor.

4.1 Models evaluated

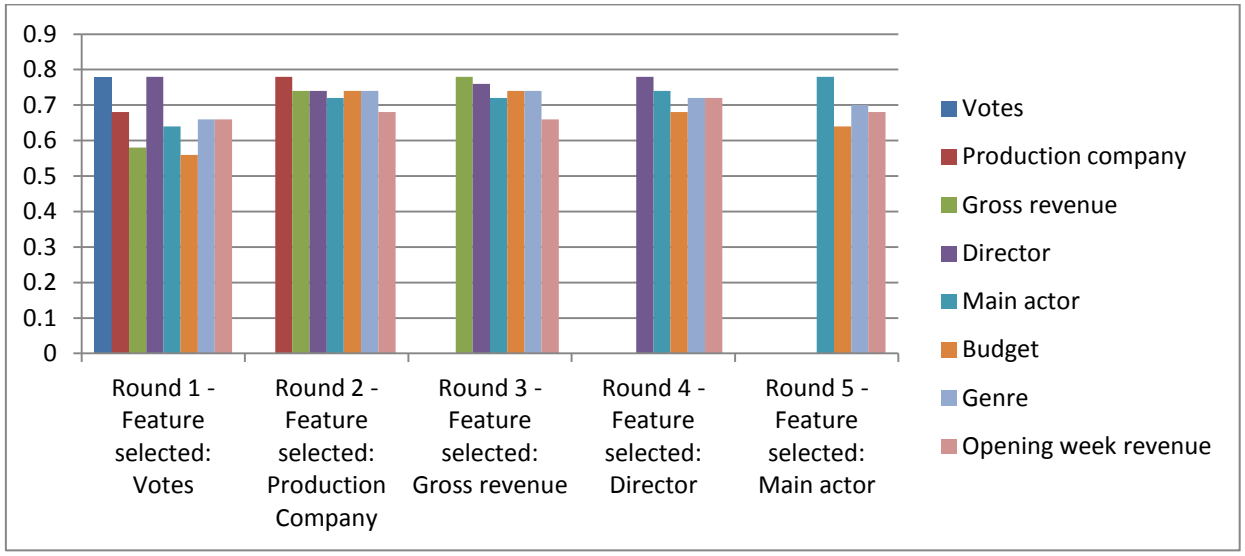
K Nearest Neighbor algorithm was used to predict overall rating of movies with k=25 and using Hamming distance to estimate the distance between two data points. Hamming distance is the percentage of features that are different. These handle nominal features and the continuous features like revenue were discretized by grouping them based on their frequency distribution. From the comments that we collected, we used only those movies (763 movies) for which we could obtain most of the features from IMDB. We used 550 samples for training, and used five-fold cross validation for feature selection, taking 100 movies out at a time.

4.2 Features evaluated

The movie features evaluated were budget, genre, opening week revenue, main actor, director, gross revenue, production company, and number of votes.

4.3 Feature selection

Forward search was used for selecting five best features which came out to be number of votes, Production Company, gross revenue, director, and main actor. Since the prediction of movie star rating based on metadata with 0 tolerance gave poor results (accuracy less than 50%), features were evaluated with a tolerance of 1. The results for cross validation are shown below:



After selecting the best 5 features, cross validation gave 78% accuracy with a mean deviation of 1.06.

4.4 Analysis of Cross Validation Results

We believe the high error observed for 0 tolerance was mainly due to other features that affect a movie's goodness value like its plot which cannot be quantized and hence not captured in our feature list. During cross validation, genre did not give good results probably because we included only one genre for each movie. But a movie can be classified under more than one genre at the same time (like humor and drama). Budget and opening weekend revenue details were not available for all movies. We assigned the mean value of all other movies to these features, which might have resulted in them being rejected. While selecting features, actor was one of the last features to be selected. We had selected only the first actor in the credits which might not be the best strategy. Future work should include obtaining a more exhaustive data set for budget, revenue and actors.

4.5 Test Result

Accuracy on test data was measured at two tolerances. Allowing a deviation of 1 from actual value obtained an accuracy of 82.00% and a deviation 2 from actual value gave an accuracy of 93.69%. In the first case, the mean deviation was 0.9915 and standard deviation 1.36.

5. Combining Naïve Bayes and movie metadata

As a final step, we attempt to combine the prediction from the Naive Bayes (which provides the rating for an individual comment) and the metadata (which provides the overall rating for the movie) to provide a new prediction for an individual comment. The motivation for doing so is that Naive Bayes assumes prior overall rating classes, which we attain from the frequencies of classes in the training data. However, it is also true that not all movies are created equal. Movies which are considered "good" overall will, in theory, tend to have better individual comments as well. In mathematical terms, the decision to combine metadata results with Naive Bayes was:

$$\mathrm{p}(\mathrm{rating_i} = \mathrm{C} \,|\mathrm{overall\ rating}) \neq \mathrm{p}(\mathrm{rating_i} = \mathrm{C})$$

The data was combined linearly in the following fashion:

$$\mathrm{Prediction_{Combination}} = \mathrm{Prediction_{Naive\ Bayes}} * \mathrm{k} + \mathrm{Prediction_{KNN}} * (1 - \mathrm{k}), \qquad \text{where } \mathrm{k} \in [0,1]$$

That is, the new prediction for a given comment i is given by the Naive Bayes prediction times some factor k plus the overall movie prediction times $(1 - \mathrm{k})$. The coefficients were chosen as k and $1 - k$ so that the new value will be bound

by the earlier two predictions and would tend towards the better predictor. The parameter k was chosen to minimize the sum of the absolute differences between the true value and the predicted value using a simple grid search with .01 step size. Using this model, we shade our prediction upwards if the metadata suggests that a movie is better than as suggested by the Naïve Bayes predictor, and vice-versa for a bad movie. The result of testing the new model is as follows:

Model	Accuracy with 0 tolerance	Average deviation
Naïve Bayes	51.23%	1.1987
Naïve Bayes + Metadata	49.78%	1.2675

Incorporating metadata into the Naive Bayes predictor actually resulted in roughly similar to slightly worse results overall. The optimal value for k determined from the training set was .92 indicating that a high weight was placed on the Naive Bayes model and low weight on metadata. This might be due to the fact that most of the movies had similar overall ratings. The information gleaned from the metadata did not provide better insight into an individual reviewers comment as we thought it might.

6. Conclusion

Naïve Bayes with a feature set obtained after removing stop words and features with low information content gave the best performance. It predicted the star rating with 52.61% accuracy from amongst 10 different classes on the test data. The mean deviation of 1.1934 stars indicates that the algorithm on the average does a good job of identifying the polarity of the user comment.

We presume that some of the reasons for not getting higher accuracy while predicting individual rating could be the following. The star rating assigned by a reviewer on IMDB can be affected by other factors like general impression of that movie among the public. It is possible a person liked the movie for its uniqueness, or plotline, or prejudice of the reviewer towards an actor or a genre. These would vary from person to person and hence would not show up as a pattern during text classification. It is also possible that a person seeing an extreme rating for a movie does not agree with it and tries to contribute towards altering this by giving an individual rating in the opposite extreme even though he found the movie to be of average value. For example, if a movie was rated 9 and a user thinks it should be a 7, he might give a 3 or 4 just so that it brings down the overall rating.

We also tried to find a direct relation between individual rating and general perception of a movie as indicated by its average star rating. It was observed that the linear combination of parameters that we evaluated did not establish any relationship between the two. However, it is possible that they are related in some higher degrees, maybe, second or third degree polynomial functions of the parameters. Further work could be done in this area, probably using SVM.

7. References

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

[2] Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A. Smith. Movie Reviews and Revenues: An Experiment in Text Regression. Proc of the North American Chapter of the Associatoin for Computational Linguistics Human Language Technologies Conference. (2010.)

[3] Tsutsumi K, Shimada K, Nedo T. Movie Review Classification Based on a Multiple Classifer. The 21st Pacific Asia Conference on Language, Information, and Computation (PACLIC) (2007)

8. Credits

The data set used in the project was obtained from Andrew Maas. We extend our sincere gratitude for his support and guidance during the course of the project. We would also like to thank Dr. Andrew Ng. for providing us with the opportunity to work on this project. The material covered in the lecture and the notes gave us very good insight about machine learning concepts that helped us to come up with algorithms to complete this project.