Twerating: Pooling Twitter Tweets to Provide Accurate Movie Rating

Charles Wang
cvwang@umich.edu

Chen Luo
luochen@umich.edu

Zhiyi Fan
fanzy@umich.edu

Hang Yu
hangy@umich.edu

Chengyu Yang
cyuyang@umich.edu

Abstract

As the number of movies released each year increases, the task of choosing a movie becomes increasingly hard for the audience. Established movie rating services, like IMDb and Rotten Tomatoes address this problem by collecting peoples ratings on their sites. Our team has created a new service, Twerating, that is a machine learning based service that aims to provide more accurate movie rating based on massive compiling and analyzing massive Twitter tweet sets relating to movies.

1 Introduction

Movies have always been a popular form of entertainment, most audiences tend to judge the movies quality based on ratings from websites like IMDB and Rotten Tomatoes, and decide whether a movie is worth watching. However, the ratings from these websites are limited in several ways. Even the most popular movies only amass several hundred written reviews. And most of these ratings may have considerable bias due to limited amount of reviews and the more opinionated reviewers that tend to write reviews. Most people do not write reviews on these websites, instead they tend to show opinions on twitter, but these opinions based on a much larger number than reviewers on IMDB are ignored as sources of ratings. Twerating addresses this issue by creating an unbiased rating system based on a much larger group by analyzing tweets on Twitter to do with movies.

2 Related work

From the Machine Learning point of view, our work can be classified as sentiment analysis/classification problem. More specifically, movie review sentiment analysis problem. Pang, Lee [6] gave a very good introduction on how we could start working on the sentiment analysis problem. They stated the importance of the application of sentiment analysis on the review-based website, which indirectly proves the usability of our work. Pak, Paroubek [5] used Twitter as a source for gaining data on the opinions on the product. Although they used a different approach than us, they demonstrated the viability of using Twitter as a massive opinion-based pool. Oghina et al [4] actually focused on the same topic as us. They did the analysis using linear regression, but we tried to use more approaches to explore which algorithm performs the best. In order to start the work, we searched on how feature engineering works in the sentiment analysis problem. There are several methods used in other papers:

1. **Tf-idf** Tf-idf stands for Term frequency-Inverse document frequency. Sebastiani [8] concludes that tf-idf has the following properties: 1) The more often a term occurs in a document, the more it represents the content. 2) The more documents a term occurs in, the less discriminating it is. However, it ignores any semantic relationship within the document.

2. **Word2Vec** Word2Vec finds the continuous vector representation of words from a very large data sets. It measures the representation by using word similarity. It is developed by Mikolov et al [3]. They concluded that using the model in word2vec will increase the accuracy dramatically.

Also, there are lots of algorithms used by various papers:

1. **Naive Bayes** Naive Bayes is based on the assumption that the words are independent in the context. Lewis [2] gave a brief review on how Naive Bayes can be applied to information retrieval. He states that even if Naive Bayes is less favorable in the text classification problem, it still shows some effectiveness.
2. **Ranking SVM** Ranking SVM aims to map documents into several ordered categories. The total order must exist for the problem that Ranking SVM applied. Cao et al [1] adapted the Ranking SVM to the information retrieval problem. They ran the algorithm on web search data and showed the increase in accuracy.
3. **Convolutional Neural Network** Santos, Gatti [7] applied neural network on the sentiment analysis of short texts. They tried to extract character- to sentence- level features. Their algorithm achieves 85.7% accuracy on the Stanford Twitter Sentiment Corpus.

3 Proposed method

We propose a pipelined workflow to achieve the goal, as shown in Figure 1. It is further broken down into 4 sub-tasks: Data collection, Data Preprocessing, Feature Generation, and applying machine learning algorithms. We evaluate seven different learning algorithms including kNN, Linear Regression using Newtons method, Naive Bayes, Gaussian Kernel Regression, SVM Ranking, Linear SVM, and Polynomial-kernel SVM.

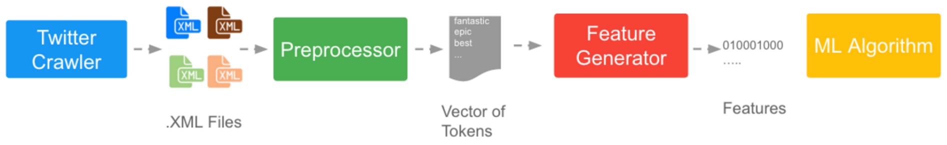

Figure 1: Flow diagram of Tweratings entire pipeline process for predicting movie ratings

3.1 Data collection

Data collection pipeline consists of the following steps:

1. **Movie list collection** For each movie rating class, we need a large number of movies for training and testing. To automate the movie selection process, a python script is used to send Http request to IMDB with specified movie number. By IMDB advanced searching API, we are able to get most popular movies. For this project, we chose the top 50 movies in each rating for training data.
2. **Tweets retrieval** We built a tweet crawler script by an external library which can send http requests to twitter with a high frequency. Since retrieving tweets takes a long time, the crawler script supports multi-threading. With the crawler, we managed to get 2000 tweets for each movie.

3.2 Data pre-processing

Unlike the formal literature, Twitter's language is very short and has lots of malformed words. In order to avoid the error caused by other irrelative information, we did pre processing on the data we retrieved from Twitter.

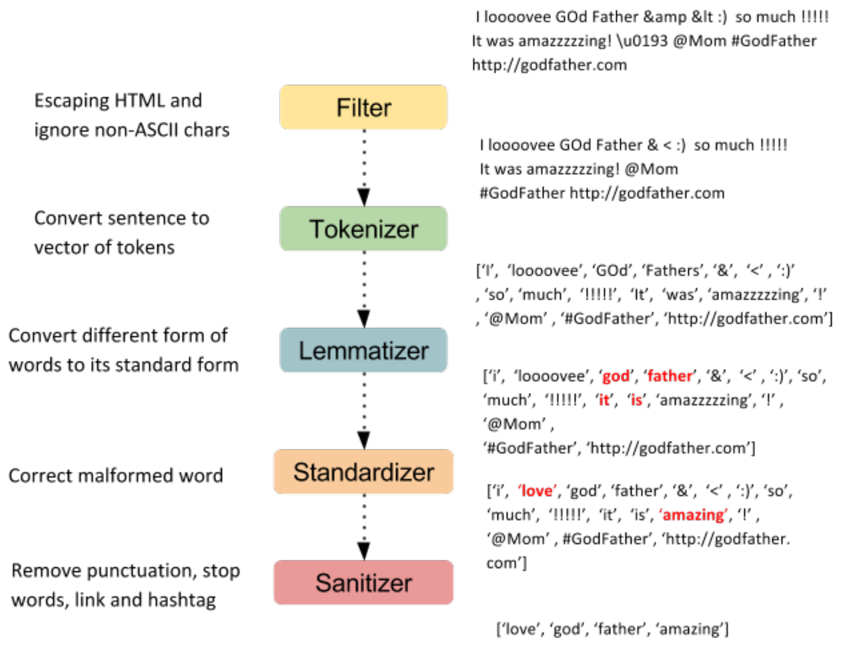
I loooovee GOd Father &amp &lt :) so much !!!!!
It was amazzzzzing! \u0193 @Mom #GodFather
http://godfather.com
Escaping HTML and ignore non-ASCII chars
Filter
I loooovee GOd Father & < :) so much !!!!!
It was amazzzzzing! @Mom
#GodFather http://godfather.com
Convert sentence to vector of tokens
Tokenizer
['I', 'loooovee', 'GOd', 'Fathers', '&', '<', ':)'
, 'so', 'much', '!!!!!', 'It', 'was', 'amazzzzzing', '!'
, '@Mom', '#GodFather', 'http://godfather.com']
Convert different form of words to its standard form
Lemmatizer
['i', 'loooovee', 'god', 'father', '&', '<', ':)', 'so',
'much', '!!!!!', 'it', 'is', 'amazzzzzing', '!',
'@Mom',
'#GodFather', 'http://godfather.com']
Correct malformed word
Standardizer
['i', 'love', 'god', 'father', '&', '<', ':)', 'so',
'much', '!!!!!', 'it', 'is', 'amazing', '!',
'@Mom', #GodFather', 'http://godfather.
com']
Remove punctuation, stop words, link and hashtag
Sanitizer
['love', 'god', 'father', 'amazing']

Figure 2: Pipeline of preprocessing. Left of the picture demonstrates the function of each step. On the right side, an example is provided for reference.

3.3 Feature generation

In this project, term frequency and word embeddings are used to generate feature vectors. We aggregate both 1000 and 2000 tweets to generate one feature vector for each movie. Based on the test result, word embedding vectors outperform term frequency vectors.

1. **Term frequency** Firstly, we preprocess all the tweets to generate a vocabulary set. Words which occurs more than 5 times are selected into the vocabulary set. The final vocabulary set has size of 9500 words. Secondly, term frequency vector for each movie is generated according how many times each word occur in the tweets. The term frequency vectors are specially useful for naive bayes algorithm because the probability of each in different ratings can be calculated.

2. **Word embeddings** To better capture the linguistic regularities of the text document, we use Word2Vec library to generate the feature vectors for all the movies. It uses techniques including K-Means word clustering to map text to a high dimensional real number vector. Compared with term frequency vector, W2V feature vectors are able to reduce the dimensionality from 9500 to around 300 and thus greatly reduce the sparsity of the feature vector. We use all the training and testing data to train the W2V feature generator and set the minimum term frequency to be 15 (terms which occurs less than 15 wont be considered).

3.4 Machine learning algorithm

We tested our feature vectors on seven different learning algorithms: KNN, Linear Regression using Newtons method, Naive Bayes, Gaussian Kernel Regression, SVM Ranking, Linear SVM, and Polynomial SVM.

3.4.1 kNN

KNN is a simple algorithm that gave a good baseline accuracy to surpass. Depending on the value of K, the algorithm may be impractical as NLP problems usually deal with a large database of training text samples.

- Applying L2 norm as the distance function, we get k nearest neighbors:

  $$kNN(x) = \{(x(1)', t(1)'), ...((x(k)', t(k)')\}$$

  The class label for testing data is predicted by majority vote

  $$y = \arg\max_t \sum_{(x',t') \in kNN(x)} 1[t' = t]$$

- When setting k = 1, kNN achieves the best testing accuracy: 67%

3.4.2 Linear regression using Newtons method

Newtons method of linear regression is simple to implement by adapting the Moore-Penrose pseudo-inverse equation. However, the complexity to perform a matrix inverse grows cubically with the number of features, so it is inefficient on high dimensional vectors. Also, computing the inverse of sparse matrices (such as bag-of-words data sets) can lead to error in computation.

- $w_{ML} = (\Phi^T \Phi + \lambda)^{-1} \Phi^T t$
- The best accuracy result was achieved using a lambda regularizer value of 1.

3.4.3 Naive Bayes

Naive Bayes was not intended to be applied on word embedding vectors created from Word2Vec. This is unfortunate since Word2Vec extracts lots of contextual information from text.

- $\frac{P(C1|x)}{P(C2|x)} = \frac{P(C1) \sum_{j=1}^{M} P(x_j|C1)}{P(C2) \sum_{j=1}^{M} P(x_j|C2)}$, where C2 represents the rating classification 2.
- We only used a bag-of-words vector for each pool of movie reviews. We extended a spam/non-spam classification algorithm to perform multi-classification for each rating label.
- We did not use Word2Vec produced feature vectors on Naive Bayes as this algorithm is only meaningful with respect to frequency of words.

3.4.4 Kernel regression

RBF kernels should not have a great performance on frequency text or embedded text vectors as they may lead to overfitting in NLP problems where similar categories can have large variation of text.

- $$\frac{\sum_i K(X, X^{(i)}) t^{(i)}}{\sum_i K(X, X^{(i)})}$$
- Gaussian kernel: $K(x, z) = \exp(-\frac{|x-z|^2}{w\tau^2})$
- The best accuracy result was achieved using a tau value of 0.05.
- Kernel regression computations made on Matlab using tau values below 0.05 usually resulted in loss of precision leading to NaN values.

3.4.5 SVM

SVM with linear kernel is recommended for text classification. This is because text is often linearly separable. Linear kernel also performs well when there is a lot of features which is the case with NLP problems that have large features and data points. Since linear kernel SVM does so well, mapping it to a higher dimension in the case of polynomial kernel should give the classifications a slight improvement.

- Linear SVM

$$\text{minimize:} Q(w, b, \xi_i) = \frac{1}{2}||w||^2 + C\sum_i \xi_i$$

$$\text{subject to:} y_i(w \cdot x_i - b) \leq 1 - \xi_i, \forall(x_i, y_i) \in D, \xi_i \geq 0$$

- Kernelized SVM

$$w = \sum_{n=1}^{N} a^{(n)} t^{(n)} \phi(x^{(n)})$$

$$y(x) = w^T\phi(x) + b = \sum_{n=1}^{N} a^{(n)} t^{(n)} k(x, x^{(n)}) + b$$

- Polynomial kernel: $K(x, z) = (x'z + c)^M, c \geq 0$
- SVM algorithms were ran using Matlab SVM library.

3.4.6 Ranking SVM

While ranking SVM provides orderings with decent correlation to the actual ranking labels for text classification, since the values it returns are relative and not indicative of any solid classification, the feedback cannot be used to give a definitive rating but only a relative rating with respect to other movies.

-

$$\text{minimize:} Q(w, b, \xi_i) = \frac{1}{2}||w||^2 + C\sum_i \xi_i$$

$$\text{subject to:} \forall\{(x_i, x_j) : y_i < y_j \in \mathbb{R}\} : w \cdot x_i \geq w \cdot x_j + 1 - \xi_{ij} \forall(i, j) : \xi_{ij} \geq 0$$

- Cornell's SVM$^{\text{rank}}$ is used in our system.

4 Experimental results

4.1 Feature selection

The quality of feature vectors are affected by factors including number of tweets aggregated per movie and the vector size generated by W2V. We perform the following tests with three ML algorithm (linear SVM, polynomial SVM, KNN) to select the best feature:

- Fix W2V vector size, compare the accuracy for different number of tweets aggregated per movie (shown in Figure 2). 50, 500, 1000, 2000 tweets per movie are selected for testing.
- Fix the number of tweets aggregated per movie, compare the accuracy among different W2V vector size (shown in Figure 3). Vector sizes of 50, 150, 225, 300, 400 are selected for testing.
- Fix the number of tweets aggregated per movie, and compare the accuracy of raw vector and W2V feature vector (shown in Figure 4).

As shown in Figure 3, aggregating more tweets for movies will generally achieve better performance for the three algorithms. Taking the average of more word vectors can make the feature less-biased. Figure 4 shows that when the tweets number is fixed, feature vector of size 300 achieves the best

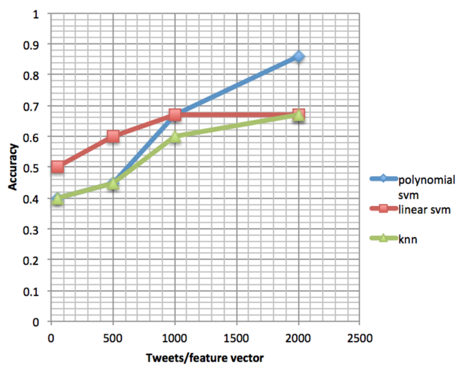

Figure 3: Line graph comparing rating accuracy vs. the number of tweets per movie

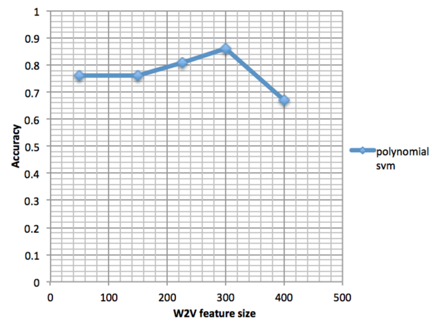

Figure 4: Line graph comparing rating accuracy vs. Word2Vec feature vector size

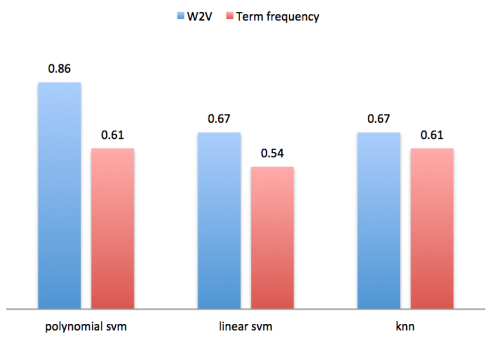

Figure 5: Bar graph comparing the accuracy of Word2Vec feature vectors vs. term frequency feature vectors.

performance. it indicates increasing the dimensionality of feature vector won't have much performance gain. We can also conclude from Figure 5 that Word2Vec feature representation outperforms the term frequency vector. Based on the testing results, we decide to aggregate 2000 tweets per movie and use Word2Vec feature with size of 300 for the following experiments.

4.2 Performance analysis

To evaluate the performance of Twerating, we measured the testing accuracy and correlation between prediction ratings and IMDB scores. The predicted rating is considered accurate if it falls in IMDb Rating ±1. Correlation is another important metric because it shows how well the prediction results fit the labels. When analyzing different algorithms, we used training accuracy to illustrate whether an algorithm is appropriate for our use case.

For the 21 testing movies, we achieved the best accuracy using polynomial kernelized SVM on our Word2Vec vectors compiled from 2000 tweets and represented in 300 features. We achieved an accuracy of 86% and a correlation of 0.87.

The training accuracy of polynomial-kernel SVM reaches 99%, which indicates the model is appropriate. Setting the polynomial degree as 3 or 4 achieves the best testing accuracy. The accuracy decreases significantly when the degree is greater than 5. Polynomial-kernel outperforms linear-kernel SVM because it extracts higher degree features. It also outperforms gaussian-kernel because gaussian kernel might result in overfitting in NLP problems.

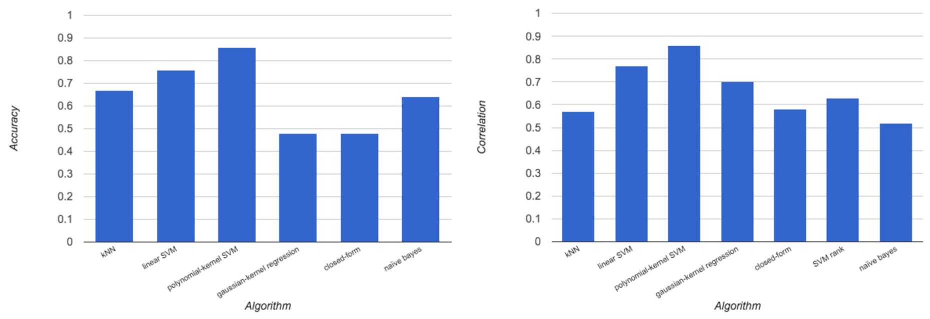

Figure 6: Left:Bar graph comparing the accuracy of different algorithms. Note: naive bayes was performed on term frequency feature vectors; all other results were obtained using Word2Vec feature vectors; SVM rank does not provide labels.
Right:Bar graph comparing the correlation of different algorithms. Note: naive bayes was performed on term frequency feature vectors; all other results were obtained using Word2Vec feature vectors.

From Figure 6, we observe that polynomial-kernel SVM is 10% more accurate than the next accurate algorithm, linear SVM. Algorithms which perform poorly has low training accuracy, showing underfitting problem. For example, closed-form linear regression has 40% training accuracy. The scatter plot shown in Figure 7 illustrates the correlation between IMDBs ratings and Tweratings ratings found using the aforementioned algorithm and feature vector set and the following table details the movie test set shown in the scatter plot. Table 1 records the detailed prediction results for the testing set.

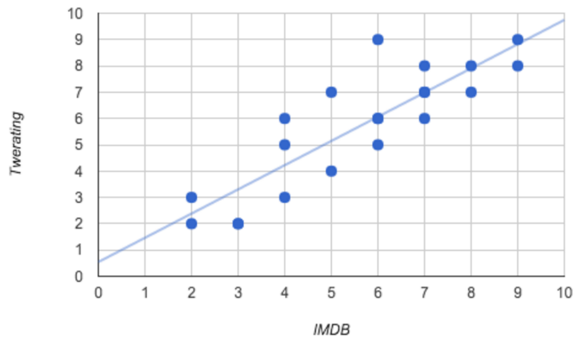

Figure 7: Scatter plot of IMDBs movie ratings vs. Tweratings predicted movie ratings that were obtained using polynomial kernelized SVM on Word2Vec vectors compiled from 2000 tweets and represented in 300 features.

Movie Title	IMDB	Twerating	Movie Tile	IMDB	Twerating
Battlefield Earth	2	2	The Last Witch Hunter	6	6
Epic Movie	2	3	War Room	6	9
Catwoman	3	2	Furious 7	7	8
Jack and Jill	3	2	Brooklyn	7	6
The Canyons	4	5	Goosebumps	7	7
The Ghost Dimension	4	3	Hotel Transylvania 2	7	7
Super Girl	4	6	Bridge of Spies	8	8
Our Brand is Crisis	5	4	The Peanuts Movie	8	7
Batman Forever	5	7	Schindler's List	9	8
Burnt	6	5	The Dark Knight	9	9
Miss you already	6	6			

Table 1: Preview comparison of IMDB vs. Tweratings predicted movie ratings on a sample set of films obtained using polynomial kernelized SVM on Word2Vec vectors compiled from 2000 tweets and represented in 300 features.

5 Conclusion

Nowadays, commercial movies tend to be promoted by industries and thus their movie ratings and reviews on traditional movie website like IMDB may be largely biased. Also, due to the limited number of reviews for a specific movie, the influence of spoilers on the final movie scores is very large on IMDB and Rotten Tomatoes. To address these issues, we have presented Twerating, an innovative movie rating system where we use Twitter as our reviews source to get a more accurate movie rating based on users tweets. By utilizing the large amount of tweets about a specific movie, we managed to get a score that reflects users actual preferences. Another benefit of our Twerating is that users dont have to go to IMDB or Rotten Tomatoes to write a review. They could just put their feelings in the tweets which can be used by our rating system later on. In all, our rating system has a very high rating accuracy (86%) allowing users to easily find more correct and unbiased film scores by using Twerating.

5.1 Future work

1. **Generating better features:** Current Word2Vec model is trained by all the training and testing data. The model might obtain better quality if being trained by a larger text data set. We generate Word2Vec feature by taking the average of all the word vector in the aggregated tweets. Each word doesnt show any relation with other words in the sentence. To address the problem, we can use better feature to represent the sentence syntax and semantic characteristics of a document.
2. **Applying more algorithms:** Recurrent Neural Network and Convolutional Neural Network shows good performance in the related works [9]. We can also implement CNN and RNN as our predictive model.

Acknowledgments

We would like to thank Prof. Honglak Lee for his guidance that was crucial to this projects success. We would also like to thank GSIs Changhan (Aaron) Wang and Renhan Zhang. They gave lots of helpful suggestions that improved our project.

References

[1] Yunbo Cao et al. “Adapting ranking SVM to document retrieval”. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2006, pp. 186–193.

[2] David D Lewis. “Naive (Bayes) at forty: The independence assumption in information retrieval”. In: *Machine learning: ECML-98*. Springer, 1998, pp. 4–15.

[3] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[4] Andrei Oghina et al. “Predicting imdb movie ratings using social media”. In: *Advances in information retrieval*. Springer, 2012, pp. 503–507.

[5] Alexander Pak and Patrick Paroubek. “Twitter as a Corpus for Sentiment Analysis and Opinion Mining.” In: *LREC*. Vol. 10. 2010, pp. 1320–1326.

[6] Bo Pang and Lillian Lee. “Opinion mining and sentiment analysis”. In: *Foundations and trends in information retrieval* 2.1-2 (2008), pp. 1–135.

[7] Cıcero Nogueira dos Santos and Maıra Gatti. “Deep convolutional neural networks for sentiment analysis of short texts”. In: *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*. 2014.

[8] Fabrizio Sebastiani. “Machine learning in automated text categorization”. In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[9] Aliaksei Severyn and Alessandro Moschitti. “Twitter sentiment analysis with deep convolutional neural networks”. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2015, pp. 959–962.