Binary Sentiment Analysis on IMDb Movie Reivews

Juntao Zhu
Computer Science and Engineering Department
University of California, San Diego
La Jolla, California
juz088@eng.ucsd.edu

Wenkang Yu
Computer Science and Engineering Department
University of California, San Diego
La Jolla, California
wey066@eng.ucsd.edu

ABSTRACT

This paper details our analysis of Large Movie Review Dataset, which contains movie reviews along with their associated binary sentiment polarity labels. Based on these reviews, it is possible to learn reviewers' words and classify their attitudes into positive ones and negative ones.

KEYWORDS

Sentiment Analysis, Word Vector, Logistic Regression, Support Vector Machine

1 INTRODUCTION

Nowadays, the booming movie industry makes it harder for audience to choose from and movie reviews plays a important role in providing references. People are willing to share their thoughts and feelings after watching movies and thus give an overview of a movie from personal aspects. Due to the huge amount of total reviews, technologies of Data mining are needed to uncover information which will both confirm or disprove common assumptions about movies based on massive reviews. It can also allow us to predict the success of a future film given select information about the film before its release.

The IMDb is an excellent resource to find detailed information about almost any film ever made. But as it has been pointed out elsewhere[1], main difficulty is that movie review mining is very domain specific and word semantics in a particular review could contradict with overall semantic direction (good or bad) of that review. For example, an "unpredictable" camera gives negative meaning to that camera model, whereas a movie with "unpredictable" plot sounds positive to moviegoers.

Therefore, for this project, we have decided to use Large Movie Review Dataset, which contains massive format suitable reviews of movies from IMDb. We cast sentiment analysis upon it and then use our training result to predict reviewer's attitude based on his/her review. And Figure 1 gives a taste about how the dataset looks like as it shows occurring frequency of unigrams/

The organization of this project report is as follows: Section 2 provides more details about the database we use and the difficulties in attempting data mining upon it. Section 3 gives an overview of the techniques we use to perform our analysis. Section 4 describes the actual analysis performed, and then presents the results and a discussion thereof. Section 5 gives the conclusions reached and a note about possible further work.

2 DATASET DESCRITION

In this assignment, we are going to use a large IMDb review dataset [5]. This is a dataset for binary sentiment classification containing

Figure 1: Unigrams Frequency

more data than other IMDb-related datasets. There are totally 50000 highly polar movie reviews in this dataset, and each review has their associated binary sentiment polarity labels, in the following paragraphs we describe basic statistics we poll from this dataset.

There are totally 50000 reviews, and the overall distribution of labels is balanced (25k train and 25k test sets), which makes this dataset relatively easy to train since we do not need to worry about the inbalanced label issues. Before our experiments, We further evenly split this dataset into 25k train and 25k test sets.

The creator of this dataset guarantees that there are no more than 30 reviews are allowed for any specific given movie because reviews for the same movie tend to have correlated ratings. In this dataset, a negative review has a score <=4 out of 10, and a positive review has a score >= 7 out of 10, reviews with more neutral ratings are not included here.

Each review is nothing more than a plain text in varied length. We scan over the whole training set, and find out that for each review,

The average length = 233.8
The max review length = 2470
The minimal review length = 10

And there are totally 280617 different words over all the reviews, the frequency distribution of different words is shown as Figure 2. From this figure we can the, most words only appear very few (one or two) times among the reviews, so we can concentrate on the words that appear most frequently.

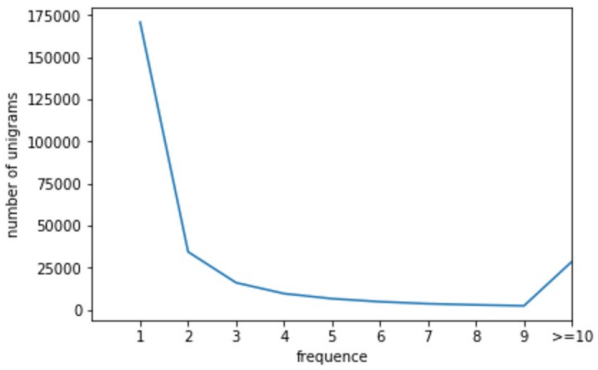

Figure 2: Word Frequency Distribution

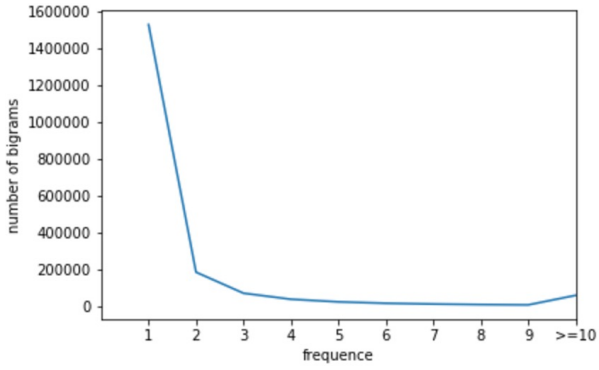

Figure 3: Bigram Frequency Distribution

3 PREDICTIVE TASK

Basically, the predictive task we are going to solve in this dataset is to predict binary rating labels base on plain review texts, and the measuring metrics for this task is the classification accuracy on test set. In the following part of this section, we are going to discuss the baseline of this task, as well as the different input features we are going to use as input of classification model.

3.1 Baseline

Since our task is binary classification and our input is plain text with varied length, there is not much choice on the baseline algorithm. What's more, this dataset is evenly distributed in positive and negative patterns, so choose label with majority in training set will certainly not work in this task. Therefore, we tried two other different baseline algorithm

3.1.1 Regression Based on Review Length. At first, we tried to perform logistic regression based on the length of reviews, but both the training accuracy and test accuracy turn out to be near 0.5, which means that the label is independent to review length, and the review length is not a good feature in this task.

3.1.2 Naive Bayes Based on Words. Then we tried to perform Naive Bayes analysis on this dataset. We treat the review text as a set of words $\{w_1, w_2, ..., w_n\}$, and assume different words are conditionally independent given a review is positive or negative, then the possibility on the reviewś label given its text can be expressed as

$$P(lb|w_1, w_2, ..., w_n) = \frac{p(lb) * p(w_1|lb) * ... * p(w_n|lb)}{p(w_1, ..., w_n)}$$

Where labelś value can be chosen from $\{positive, negative\}$, and the prior possibility is equal between two kind of labels, the posterior possibility $p(w_i|lb)$ can be calculated by

$$p(w_i|lb) = \frac{|\{d \in D \text{ and } lb_d = lb|w_i \in d\}|}{|\{d \in D|w_i \in d\}|}$$

Where d is review in training set.

Using above Naive Bayes Model, we can achieve accuracy 0.638 on test set.

3.2 Feature Construction

Before we can use review text to generate feature vectors, we need to perform several regularization on the raw reviews. First, we keep only characters, spaces from the raw reviews, dropping out other special symbols or punctuations. Second, we transform all the characters into lower case. After that, we can derive a word sequence split by space.

There are a few ways to generate numerical input vectors for later classification task based on above word sequence, we list and discuss different feature construction models as below.

3.2.1 Bag of Words. The bag of words model[4] is a simplifying representation used in natural language processing and informative retrieval. In this model, review text is represented as the bag of its words, disregarding word order but keeping multiplicity.

To use this model to generate input features, we have to first build a dictionary. And for each review, we need to count the appearance time of each word in dictionary, use it as the input feature. However, since we have more than 29000 different words among all the reviews, which can be even larger considering our dataset is still relative small, we need to limit the maximal size of dictionary. In our experiments, we are going to use 300 most frequently appeared words to construct a dictionary, in comparison with 300 dimensional global word vector feature model mentioned below.

In addition to the unigrams-based bag of words model, we can further build a richer predictor by using bigrams-based, or even n-grams-based bag of words model. However, this extension does not perform well in our task because the limitation on review corpus and restraint on dictionary size. The Figure 3 shows the frequency distribution of bigrams over all of the reviews. Compare this figure to Figure 2, we can see that there are far more different bigrams than unigrams, and it also has a less average frequency (more sparse) compared to unigrams, which requires a increasing dictionary size (feature dimension) to make use of this representation.

In our experiment, the unigram model does perform better than multi-gram model at a same and small dictionary size, so we are not further covering multi-gram in later discussion.

3.2.2 Tf-idf. Tf-idf[2], in short for term frequency–inverse document frequency, is a kind of extension of bag of words model just discussed. It is numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus, instead of the pure frequency of one word in a document which the bag of words model will compute.

The definition of tf-idf is shown below, since the computation still relies on the term frequency, we need to construct the dictionary first so that we can generate input features with fixed length, and we also constrain this size as 300, for comparison to different feature model.

$$tf(t,d) = \texttt{frequency of term tin the document d}$$
$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|}$$
$$tf - idf(t,d,D) = tf(t,d) * idf(t,D)$$

Both tf-idf and bag of words representations are invariant to the document length, while in tf-idf representation, the feature vectors are now determined by the words that are most "characteristic" of the document.

3.2.3 Global Word Vector. There is another text representation called Global Word Vector[6]. The basic idea of this representation is that, we can learn the relations between any two words based on very large corpus crawled from Internet, and then map each word (or common ones) into a fixed dimension space, where correlation between two words can be reflected by two vectors in that space.

The details of this representation will be further discussed in Related Literature section. In our experiment, we download a dictionary of well trained global word vectors which contains more than 2.2 million common words trained by 840 billion tokens, and each word is mapped into a space with 300 dimensions.

For each review text, we are going to sum up the global word vectors of each word in this text, and use it as input feature in our task. In fact, the global word vectors can be used in a better way by applying some sequential learning models like LSTM on that, but since we haven't covered those material, we only use the sum vector as our input feature for regression models discussed in next section.

4 CLASSIFICATION MODELS

In this section, we are going to discuss the classification models we use to predict binary labels for review text take the features described in last section as input. Basically, we have tried two different classification models learned from class, Logistic Regression and Support Vector Machine, to perform this classification task.

4.1 Logistic Regression

First, we apply Logistic Regression Model to solve this classification task, and use gradient descent algorithm to optimize this model into a local minimum.

The main idea of Logistic Regression is to learn a weight w which minimize the cross entropy loss function with regularization

$$Loss = \sum_n (t_n \log y_n + (1 - t_n) log(1 - y_n)) + C\|w\|_2$$

Where t_n is the target label, C is regularization parameter, and y_n is predicted by the function

$$y(x) = \begin{cases} 0 & \text{if } w \cdot x < 0 \\ 1 & \text{if } w \cdot x \geq 0 \end{cases}$$

While applying this simple logistic regression model, we need to tune several hyper-parameters in order that this model does not overfit training set.

The first parameter we need to take care of is learning rate η of gradient descent. An overly large learning rate will cause oscillating in learning procedure, and will make it hard to converge into a "deep" optimum.

Another parameter is the regularization penalty C. If we set C to zero, the model will end up overfitting training set as long as the input features are linear separable, but if we set C too large, the classifier will become plain and has low accuracy.

After grid searching many possible combinations of learning rate and regularization penalty, we find out that $\eta = 0.001$ and $C = 1.0$ is a fairly good solution for all types of input features discussed above.

4.2 Support Vector Machine

Support Vector Machine(SVM) is an alternative classification model to logistic regression. Given a set of training examples, each marked as a positive label or negative label, a SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier by optimize the soft-margin formulation:

$$\arg \min_{\theta, \alpha, \zeta_i > 0} \frac{1}{2}\|\theta\|_2^2 + C \sum_i \zeta_i$$

such that

$$\forall_i y_i(\theta \cdot X_i - \alpha) \geq 1 - \zeta_i$$

And we can further efficiently perform a non-linear classification with SVM using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

During our experimenting, we find that because our feature dimension is relatively large (300 dimension) while the training dataset is small (25000 patterns), it becomes really easy to overfit training set by reaching 1.0 accuracy, especially with "rbf" kernel which maps input feature into a even larger space.

To solve this problem, we repeat many experiments on "linear" and "polynomial" kernel functions, and keep the error parameter C relatively small. In addition, we also constrain the maximal iteration number during SVM fitting process.

5 RELATED LITERATURE

Trying to figure out semantics of reviews is a well studied topic and it often referred as opinion mining. These opinions could be summarized in order to give users statistics information. And movie review mining is a special branch of opinion mining which also attracts lots of attention.

The machine learning approach applied to this problem mostly belongs to supervised classification in general and text classification techniques in particular for opinion mining. Supervised classification tends to be more accurate because each of the classifiers is

trained on a collection of representative data known as corpus. In contrast, using semantic orientation approach to opinion mining is "unsupervised learning" because it does not require prior training in order to mine the data. Instead, it measures how far a word is inclined towards positive and negative.[1]

There are several excellent works have been done on the same dataset Large Movie Review. Learning distributed representations for phrases and sentences as a whole[7] is one of them. This method only utilizes parallel data and eschews the use of word alignments. Another approach, overcoming the drawbacks of bag-of-words, proposed using Paragraph Vector[3], an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. These two approaches all achieved satisfying results that the error rate reduced to below 4%.

The optimal approach of our search results is using Global Vectors for Word Representation[6]. As global matrix factorization and local context window methods are two major models, this global log-bilinear regression model combined the advantages of the former two. The main idea is based on that word-word co-occurrence is more reasonable than simply counting. For example, if a review contains word "steam", it's more likely it will also contain "gas" rather than "ice". Built on this idea, the author introduced a global log-bilinear regression models that achieved error rate of 5.496%, which is a improvement of former two methods.

In our study, we combine data mining technologies we learned from class with state-of-art technologies and compare their features and results. More details are included in the next section.

6 RESULT AND CONCLUSIONS

Using two different classification models and three different feature representations described above, along with Naive Bayes algorithm that has been performed on this task, we can derived results as shown in Table 1.

Classification Model	Feature Type	Training Accuracy	Test Accuracy
Naive Bayes	words	0.9812	0.6381
Support Vector Machine	Bag of Words	0.9264	0.6753
	TF-IDF	0.9146	0.6685
	Word Vector	0.8819	0.7009
Logistic Regression	Bag of Words	0.8069	0.7964
	TF-IDF	0.8122	0.8043
	Word Vector	0.8591	0.8466

Table 1: Experiment Result

We may further derive following conclusions by analyzing experiments result above:

Feature Comparison. We can see that three different representations of features all have capability to binary classify review texts. While Bag of Words (term frequency) representation is quite similar in effect to TF-IDF representation, Word Vector is a better representation compared to them. But the length of review does no good in this task.

Model Comparison. When it comes to different classifier that has been tried in experiments, we can see logistic regression performs better than support vector machine regardless of the choice of feature representations, but both of them show improvements compared to Naive Bayes baseline. As discussed in section 4, this may due to the relatively large number of features versus small number of training patterns. SVM can easily overfit training data, and result in poor performance on test set.

In conclusion, although our result is not as good as state-of-art results which use mores complex sequence learning model like LSTM and takes much more computing resource, we have improved the accuracy by a lot compared to non-trivial baseline algorithm. We also try many different classification models as well as text feature representation method, discussing the causing of different performance on them. Thanks to this assignment, we have a deeper understanding of course material, and also have a peek of modern Neuro-linguistic programming(NLP) topics.

REFERENCES

[1] Pimwadee Chaovalit and Lina Zhou. 2005. Movie Review Mining: a Comparison between Supervised and Unsupervised Classification Approaches. In *Hawaii International Conference on System Sciences*. 112c.

[2] Thorsten Joachims. 1997. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In *Fourteenth International Conference on Machine Learning*. 143–151.

[3] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *Computer Science* 4 (2014), 1188–1196.

[4] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research* 2, 3 (2002), 419–444.

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. http://www.aclweb.org/anthology/P11-1015

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[7] Hieu Pham, Thang Luong, and Christopher Manning. 2015. Learning Distributed Representations for Multilingual Text Sequences. In *The Workshop on Vector Space Modeling for Natural Language Processing*. 88–94.