Sentiment Analysis on Movie Reviews

Zhuoheng Xie
Information Science
University of Pittsburgh
Pittsburgh, PA
zhx21@pitt.edu

Zhijie Yang
Information Science
University of Pittsburgh
Pittsburgh, PA
zhy38@pitt.edu

Xiujun Ling
Information Science
University of Pittsburgh
Pittsburgh, PA
xil67@pitt.edu

ABSTRACT
This paper analyzes the similarity of semantic and sentiment terms in movie reviews to estimate the polarity of movie reviews. The sentiment polarity of the reviews is classified into positive and negative. Movies with rating score >= 7 out of 10 are classified into positive, while movies with rating score <= 4 out of 10 will be classified into negative. In our experiments, we will use two types of learning methods: supervised learning and unsupervised learning. In the supervised learning, 25,000 data points are used as training set and another 25,000 data points are used as test set. In each set of data points, there are equal sizes of negative and positive movie reviews (12,500 for negative and 12,500 for positive). After tokenization of the movie reviews and removal of stop words, the model is trained with a bag of words, which comes from movie reviews, according to the relative movie ratings. The model performance is evaluated with cross validation on test set by comparing the classification of movie review polarity and rating score.

General Terms
Algorithms, Measurement, Documentation, Performance, Design, Experimentation, Security, Standardization, Languages, Theory, Validation.

Keywords
Sentiment, LSA, NMF, LDA, WordCloud, SVM, Random Forest.

1. INTRODUCTION
Movie reviews, to some extent, reflect the popularity of a movie. Whether people will go to watch a movie sometimes depends on the movie rating. Movies with higher rating tend to attract more audiences, thus having better box office. Therefore, the estimation of movie rating through movie reviews can be used to forecast whether a movie will have good box office. This will benefit the movie theaters so that they can have more reasonable arrangement on the showing slot for each movie.

In order to analyze movie review, we need take into the field of natural language processing. Generally, words are represented as indices in a vocabulary. However, this method fails to capture the rich sentiment meanings of the corpus terms.[1, 3, 4] In Potts's paper, they used vector space model to capture both semantic and sentiment similarities among the words in movie reviews. And they found out their model outperformed the LDA. In our experiments, we will use the same dataset --"Large Movie Review Dataset" from Stanford University.[2] and supervised learning is used in analysis. The main step is that we will do pre-processing on the movie reviews, including converting to lower case, removing number and stop words and stemming. In the supervised learning, after pre-processing the dataset, we use LSA (Latent Semantic Analysis), NMF (Non-negative Matrix Factorization), LDA (Latent Dirichlet Allocation) and Word Cloud, four similarity analysis methods to investigate the dataset. The dataset will be trained with a bag of words, which comes from movie reviews, according to the relative movie ratings. The model performance will be evaluated with cross validation using SVM and RF on test set by comparing the classification of movie review polarity and rating score. It will be interesting to compare the performance of these three different modeling methods.

2. Experiment
2.1 Dataset
"Large Movie Review Dataset" from Stanford University is used in this paper.[2] It contains 50,000 labeled data points for supervised learning. In the 50,000 labeled data points, we firstly process the 12,500 data points with positive rating (rating score >= 7) in training set. Each review is stored in separated txt file. Our first step is to combine all the reviews into one dataset. We use command to read the id, rating of movie, file name of each review. We get two datasets. One contains id, rate of movie and file name. The other one contains reviews which are stored in an argument x in the same order of the former one.

2.2 Pre-processing
Before applying supervised learning, pre-processing is needed to do on the movie reviews. Pre-processing is very important step in text mining because in the raw data, there are a lot of information which is unrelated with our goal or even has bad effect on the result. The pre-processing steps include setting all the terms into lower case, removing punctuations, numbers, and stop words, stemming and striping white space.

2.2.1 Lower Case
It always happens that the upper or lower case of letters exit in a single sentence. However this situation would make the result of word-counting algorithm wrong. For example, algorithm may consider "Word" and "word" as two different words which are

actually the same words. So converting all letters into lower case is needed. Here is a sample result after lowering case:

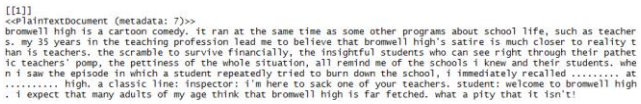

Figure 1. Corpus in lower case

2.2.2 Remove Punctuations

Punctuations are unrelated information of this project. We also need to remove them first:

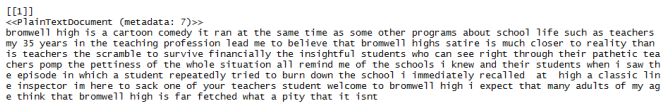

Figure 2. Corpus after removed punctuations

2.2.3 Remove Numbers

The reason of removing them is the same as removing punctuations:

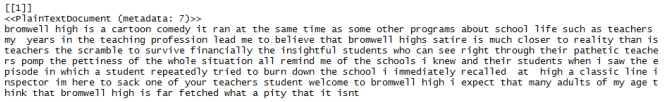

Figure 3. Corpus after removed numbers

2.2.4 Remove Stop Words

Stop words are frequently occurring and insignificant words. They help construct sentences but do not represent any content of the documents. So they should be removed before documents are indexed:

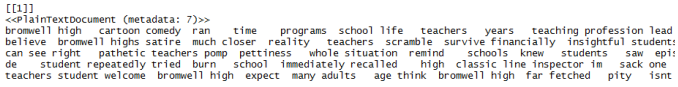

Figure 4. Corpus after removed stop words

2.2.5 Stemming

In many languages, a word has various syntactical forms. But different forms will cause low recall for a document retrieval system because a relevant document may contain a variation of a query word but not the exact word itself. Stemming is the process of reducing words to their stems or roots:

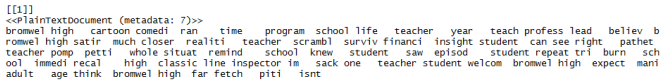

Figure 5. Corpus after stemming

2.2.6 Strip White Space

One white space is enough to separate words. In order to make documents look clean, extra white spaces should be removed:

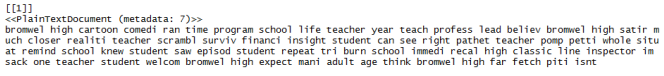

Figure 6. Corpus after removed spaces

2.3 Similarity Analysis

Similarity analysis provides a way to test statistically whether there is a significant difference between two or more groups of sampling units. In this project, we need to find the similarity between terms and documents. In other words, we need to figure out whether these terms can represent the polarity of movie reviews or not. In this part, we used LSA, NMF, LDA and Word Cloud for similarity analysis. Before starting similarity analysis, we should build a term-document matrix. Considering the configurations of our computers, we decided to randomly choose 2500 documents in positive and negative reviews respectively and terms which show up at least 1000 times in documents to build this matrix.

2.3.1 LSA

Latent semantic analysis (LSA) is a technique in natural language processing, in particular in vectorial semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.[5] LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Words are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

We applied LSA method on both positive and negative reviews of train set. Here are two different MDS maps of similarity analysis results:

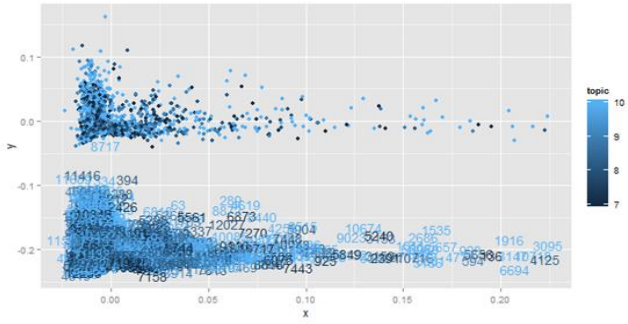

Figure 7. MDS map for Term Document Matrix with LSA (positive).

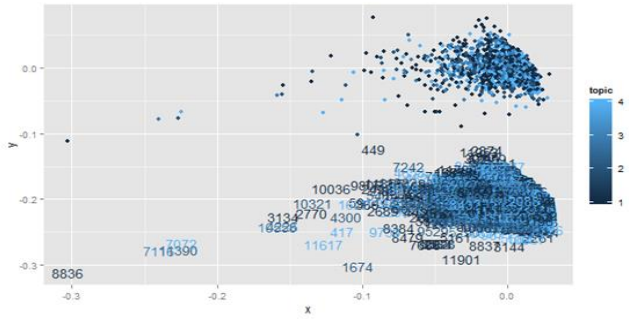

Figure 8. MDS map for Term Document Matrix with LSA (negative).

We can see from these two plots that most points gather around the area of coordinate (0, 0). It indicates that there is high similarity between terms and documents which means these high-frequency terms generated from train set can really represent the polarity of movie reviews.

2.3.2 NMF

Non-negative matrix factorization (NMF), also non-negative matrix approximation is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into (usually) two matrices W and H, with the property that all three matrices have no negative elements.[6] This non-negativity makes the resulting matrices easier to inspect. Also, in applications such as processing of audio spectrograms non-negativity is inherent to the data being considered. Since the problem is not exactly solvable in general, it is commonly approximated numerically.

Just as what we did in the last part, we also applied NMF method on both positive and negative reviews of train set. Figure 3 and Figure 4 are MDS maps of similarity analysis results:

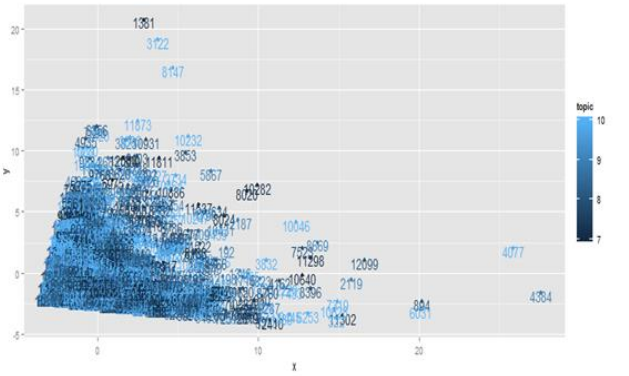

Figure 9. MDS map for Term Document Matrix with NMF (positive).

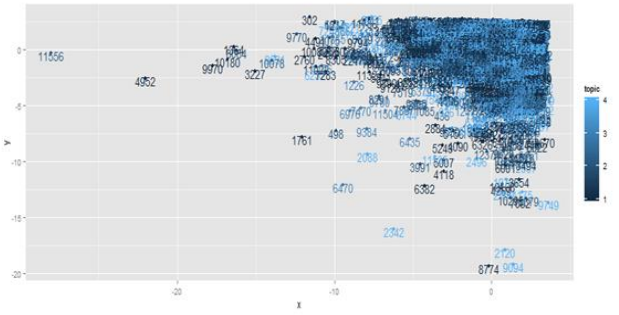

Figure 10. MDS map for Term Document Matrix with NMF (negative).

As figure 3 and figure 4 present, most points gather around the area of coordinate (0, 0). It indicates that there is high similarity between terms and documents. So these terms have a high representativeness on the polarity of movie reviews.

2.3.3 LDA

In natural language processing, latent Dirichlet allocation (LDA) is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.[7]

We built two topic model using LDA on positive and negative reviews respectively. The results are as following:

```
> terms(lda)
Topic 1 Topic 2 Topic 3 Topic 4
 "2271"  "2179"  "2057"  "2131"
> topics(lda)
   also    can charact    even    film     get    good   great    just    like    love
      2      3       4       2       2       3       1       4       3       3       3
   make    movi     one    play  realli     see    show   stori    time   watch    well
      2      1       2       4       3       2       4       3       3       3       3
   will
      3
```

Figure 11. Topic Model result with LDA (positive).

```
> lda = LDA(td.mat, 4)
> terms(lda)
Topic 1 Topic 2 Topic 3 Topic 4
 "1791"  "1167"   "941"  "2073"
> topics(lda)
    act     bad     can charact    dont    even    film     get    good
      3       2       1       4       1       4       3       4       2
   just    like    look    make    movi    much     one  realli   scene
      1       1       1       2       2       4       4       1       4
    see   stori    time   watch
      1       3       4       2
```

Figure 12. Topic Model result with LDA (negative).

We have divided all terms into four topics. We also know which terms are in the same topic from the results above. It means that the terms which are in the same topic have the same sentiment meaning. For example, in figure 5, "well", "like" and "love" are in topic 3, so these three terms have the same sentiment meaning of "like".

2.3.4 Word Cloud

Word Cloud is a visual representation for text data, typically used to depict keyword metadata on websites, or to visualize free form text.[8] Tags of Word Cloud are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence.

Figure 7 and Figure 8 are two Word Cloud generated from positive and negative reviews respectively. From Word Cloud, we can directly know what these high-frequency terms are and the similarity between terms (The closer terms are, the higher similarity they have).

Figure 13. Word Cloud (positive).

Figure 14. Word Cloud (negative).

2.3.5 Evaluation

After similarity analysis, we also applied an evaluation on test set to test whether terms which represent the polarity of movie reviews, like positive, in train set still have a high frequency in test set of the same polarity. In "Large Movie Review dataset", there is a file which contains the information of how many times each term shows up in every review. So we just need to calculate total frequency of keywords using this file. For instance, we use terms "good" and "like" as keywords which generated from train set and do a search in all positive reviews of test set. It turns out that two terms show up over 4500 times in 12500 reviews. So they also have a high frequency in test set, which is the same as what we get from train set.

2.4 Classification

Classification is popular in data mining projects. It assigns items in a collection to target categories or classes. It is widely used in today's business. For example, banks can classify customers into high, medium or low credit levels based on their profiles, finance status and debt history. In our project, we aim to classify each review into positive and negative sentiments according to the text itself. Different from other classification tasks in data mining where each item has many attributes and variables, classification in text mining mainly based on the content where there is no concrete attribute. We used RTextTools package to apply algorithms in classification task.[9]

2.4.1 Dataset

Datasets are separated as positive and negative reviews. In order to do classification, we combine both positive and negative reviews and label them to 1 and 0. Positive reviews whose ratings are 6, 7, 8, 9 and 10 are labeled as 1. Negative reviews whose rating are 4, 3, 2, 1 and 0 are labeled as 0. The combined dataset has 12500 reviews in total. And 5000 reviews are randomly selected in each trial when models work.

2.4.2 Preprocessing

Same as similarity analysis, we transform corpus into lower case, removed punctuations, numbers, stop words and step words. And we select terms whose frequencies are higher than 1000 to build Term-Document Matrix.

2.4.3 Algorithms

There are many algorithms for classification but not all are appropriate to do text classification. For example, linear classifiers including logistic regression and Naïve Bayes classifier usually use observed characteristics of the subject. The words are very sparse in reviews and its characteristics are not apparent though we have transformed them into Term-Document Matrix. A good algorithm can analyze and recognize patterns. Thus, we choose Support Vector Networks (SVM) and Random Trees that can do non-linear classification to train and test datasets.

SVM has scalable memory requirements and can handle problems with many thousands of support vectors efficiently. It present examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.[10]

Random Forest constructs a multitude of decision trees at training period and output the class that is the mode of the classes. It combines bagging and random selection of features to control variance.[11]

To implement the algorithms, we use RTextTools package in RStudio. After building the Term-Document Matrix we created a container. In container, we set the training size equal to 4500 where the first 4000 documents were used to train the machine learning model and the last 500 documents were set aside to test the model.

2.4.3 Result

We generate the results after classification using trained models. The result are composed of three indicators: precision, recall and F-score. We did 4-fold cross validation to validate the results. The performance is not so good at first several trials. Here is the result table.

	PRECISION	RECALL	FSCORE
SVM	0.595	0.595	0.59
RANDOM FOREST	0.605	0.605	0.605

Table 1. Early trials performance

2.4.4 Improvement

To improve performance, we looked back the processes above and found in our word cloud map, there are many neutral words and nouns. These words have no emotions and they are not useful when learning the reviews orientation towards positive or negative. Also, these words enlarge the sparsity of the Term-Document Matrix and increase the difficulties when algorithms learning the training dataset.

To solve this problem, we removed these words before forming the Term-Document Matrix by adding words in stop word list. The most frequency words including “movie”, “film”, “one”, “just”, “character”, “just”, etc. are added into new stop word list. Then we redid the pre-processing steps and the following steps. The performance turned out much better as follows.

	PRECISION	RECALL	FSCORE
SVM	0.63	0.63	0.625
RANDOM FOREST	0.725	0.72	0.72

Table 2. Performance after improvement

2.4.5 Evaluation

Both SVM and Random Forest performed better after improvement. The precision of SVM increases from 59.5% to 63% and of Random Forest, from 60.5% to 72.5%. The recall and F-score also increase about 10%. We believe the sparsity is an essential problem in text mining task. In order to do sentiment classification, the more oriented words are, the better classification results turn out.

3. Conclusion

Our project contains similarity analysis and reviews classification. Doing the sentiment analysis gives us a basic and comprehensive structure of our dataset and enlighten us what are the main difficulties in classification. Here are some points we learn from the project.

3.1 Unsupervised learning

In classification process, we also did unsupervised learning using hierarchical clustering and K-means clustering. However, the results are not good. Reviews are clustered mainly by words’ frequency but not words’ orientation. It means to do unsupervised learning, we need at least two dimensions including similarity and orientation.

3.2 Data pre-processing

Similar to other data mining tasks, data pre-processing is significant for later steps. In text mining, we need to do corpus and form a Term-Document Matrix. And in classification, we revised our stop word list to improve the performance.

4. Future Work

For the future work, we can add more neutral words into our stop word list and reduce sparsity in reviews.

Also, in unsupervised learning, we can corporate the similarity analysis into reviews as one new dimension. We also consider transforming each word into concrete number that reflects how much its orientation is towards positive and negative.

Finally, similarity analysis can also be used to modify supervised learning algorithms.

5. Work Assignment

This project is completed by Zhuoheng Xie, Zhijie Yang and Xiujun Ling. The following is the specific work assignment for each member.

Data collection: Zhuoheng Xie, Xiujun Ling;
Data process: Zhuoheng Xie, Zhijie Yang;
Algorithm and Program: Zhuoheng Xie, Zhijie Yang;
Result Analysis: Zhuoheng Xie, Zhijie Yang, Xiujun Ling;
Paper Word: Zhuoheng Xie, Zhijie Yang and Xiujun Ling.

6. ACKNOWLEDGMENTS

Our thanks to the dataset source and also ACM SIGCHI for allowing us to modify templates they had developed.

7. REFERENCES

[1] Maas, A. L.; R. E. D., Pham, P. T.; Huang, D.; Ng, A. Y. and Potts, C. 2011. *Learning Word Vectors for Sentiment Analysis.* The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[2] "Large Movie Review Dataset" from Stanford University http://ai.stanford.edu/~amaas/data/sentiment/.

[3] Alm, C. O.; Roth, D. and Sproat, R. 2005. *Emotions from text: machine learning for text-based emotion prediction*. In Proceedings of HLT/EMNLP, pages 579–586.

[4] Andreevskaia, A. and Bergler, S. 2006. *Mining Word Net for fuzzy sentiment: sentiment tag extraction from Word-Net glosses*. In Proceedings of the European ACL, pages 209–216.

[5] LSA- Wikipedia https://en.wikipedia.org/wiki/Latent_semantic_analysis/.

[6] NMF- Wikipedia

https://en.wikipedia.org/wiki/Non-negative_matrix_factorization/.

[7] LDA- Wikipedia https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation/.

[8] Word Cloud- Wikipedia https://en.wikipedia.org/wiki/Tag_cloud/.

[9] Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, and Wouter van Atteveldt, R. 2013. *RTextTools: A supervised Learning Package for Text Classification*. The R Journal, Volume 5/1, June 2013.

[10] Support Vector Machine – Wikipedia

http://en.wikipedia.org/wiki/Support_vector_machine

[11] Random Forest – Wikipedia

http://en.wikipedia.org/wiki/Random_forest