Sentiment classification on Large Movie Review

Xinzi Sun
University of Massachusetts Lowell
xsun@cs.uml.edu

Lijian Wan
University of Massachusetts Lowell
lwan@cs.uml.edu

Xuanming Liu
University of Massachusetts Lowell
xliu@cs.uml.edu

ABSTRACT

Making sentiment prediction from short texts such as movie reviews is a challenging topic, since there will be varieties of noises in natural languages. Our goal for this work is: Given a movie review, we need to determine whether the movie review is positive or negative based on the previous reviews. In previous work, there have been many methods to solve this problem, such as *Latent Semantic Analisis*, SVM, n-gram and so on. We used Naïve Bayes as our technique to learn the model from training set. In order to improve the accuracy and avoid zero base in taking logarithm, we added a *smoothing* parameter m into the formula.

1. INTRODUCTION

Sentiment prediction has become an important topic, since it can be useful in many fields. Social network is playing an essential role in people's daily life, and different people all around the world can be connected in these social communication platform. Many times, some groups (commercial, political) may need to know people's opinion to something, so making sentiment prediction from short text becomes necessary in these cases. Whenever a new movie comes out, many people prefer to read the reviews before entering the cinema, since they do not want to waste their time and money. Under this circumstance, it will be convenient if the movie's reviews have been categorized already, then people can just choose to see the positive or negative opinions as they want, instead of figuring out which side the reviews belongs to by themselves. On the other hand, movie makers also need to know how do people like or dislike their works. So for many reasons, sentiment classification on movie reviews is an interesting topic, and that's why our team choose it as our project direction.

The goal for our project is to predict if a movie review is positive or negative, only based on the previous reviews, namely, we use the existing labelled reviews as our training set, and learn a model from it, and use this model to decide an unlabelled review's opinion (positive or negative). The challenging part is the words people use to express their opinions are very different, and there will be many unimportant words to the sentiment. Also sometimes people prefer to use phrase in natural language, and the adverbial words can emphasis or change sentimental words' direction, such as "so amazing", "not good" and so on. Therefore, if we only consider the sentimental words with high frequency, the output will not perform good enough.

Actually, since the importance of this topic, there are many previous work, and after doing some research, our team have found some methods to solve the problem. To address the challenges mentioned above, we have tried different methods, and then choose the best one according the accuracy of the result.

We have tried the counting method (CM), SVM, General Inquirer dictionary (GI), and smoothing Naïve Bayes (SNB), and in the end we choose to use smoothing Naïve Bayes as our main method, and also we use some simple method to get sentiment word list in data pre-processing part. Actually, during our trials, SVM and smoothing Naïve Bayes both can get good results, but we prefer to use Naïve Bayes because there have been many previous work that focus on SVM method, so we want to try another direction, also, this semester's Machine Learning course does not cover Naïve Bayes, so we prefer to learn something by ourselves.

2. BACKGROUND

During the research, out team found many work in this topic, and we have read some of them, and here we are going to present three of them, which give us the most useful thinkings about our project.

Sentiment Classification on Polarity Reviews: An Empirical Study Using Rating-based Features (Dai Quoc Nguyen, Dat Quoc Nquyen, Thanh Vu and Son Bao Pham, 2014) [1]

This paper focuses on document-level sentiment classification on polarity reviews. In this paper, they firstly introduce a novel rating-based feature for the sentiment polarity classification task. The rating-based feature can be seen by that the scores *which users employ to rate entities on review websites* could bring useful information for improving the performance of classifying polarity sentiment.

They applied a supervised machine learning approach to handle the task of document-level sentiment polarity classification. For machine learning experiments, besides the N-gram features, they employed a new rating-based feature for training models.

1. They consider the rated score associated to each document review as a feature named RbF for learning classification model, in which the rating-based feature RbFs value of each document review in training and test sets is estimated based on a regression model learned from an *external independent dataset* of reviews along with their actual associated scores.

2. They calculated the value of the N-gram feature ith by using term frequency - inverse document frequency ($tf * idf$) weighting scheme for the document D

3. They utilized SVM implementation in LIBSVM for learning classification models in all their experiments.

Sentiment Classification of Movie Reviews Using Contextual Valence Shifters (Kennedy, Alistair and Inkpen, Diana, 2006) [2]

This paper mentions two applications for sentiment classification; one is question answering, and another is text summarization. Our project is similar with text summarization. And the paper gives two approaches, which we can use for reference in our project. One is count positive and negative terms, and its idea is simple, which will consider a review as positive is it contains more positive terms. Another approach is to use unigrams and bigrams as features to get SVM classifiers, which we prefer in our project. There is one resource applied by this paper is very interesting, General inquirer (GI), a dictionary that contains information about English word senses, including tags that label them as positive, negative, negation, overstatement, or understatement.

In the second approach, the author uses three methods to get feature sets. First one is to use the unigrams that appears more the three times in one review as feature set, and the second is to select the unigrams that are in GI as feature set, and the last one is to select the unigrams that are in GI, CTRW, and Adj as the feature set, where CYRW and Adj are both dictionaries that contains label negative and positive terms, and the difference between GI, CTRW and Adj is only the different characteristics or properties of a certain term, by which one can determine what part of speech the term belongs to.

At last the paper mentions that we can combine these two approaches and get a better experiment results, which we are willingly to have a try in our project

Movie Review Mining and Summarization (Li Zhuang, Feng Jing, Xiao-Yan, and Zhu, 2006) [3]

This paper gives the definitions of: Movie Feature: A movie feature is a movie element or a movie-related people that has been commented on. According to IMDB, feature classes are divided into two groups: ELEMENT and PEOPLE. Relevant opinion of a feature: The relevant opinion of a feature is a set of words or phrases that expresses a positive (PRO) or negative (CON) opinion on the feature. The polarity of a same opinion word may vary in different domain.

Feature opinion pair: A feature-opinion pair consists of a feature and a relevant opinion. An explicit F-O pair: both the feature and the opinion appear in sentence. An implicit F-O pair: the feature or the opinion does not appear in sentence. Figure 1 just illustrates the approach.

It uses Keyword list generation to build a keyword list to capture main feature/opinion words in movie reviews, and divide the list into two classes: features and opinions. For the Mining Explicit F-O Pairs: In a sentence, use keyword list to find all feature/opinion words.

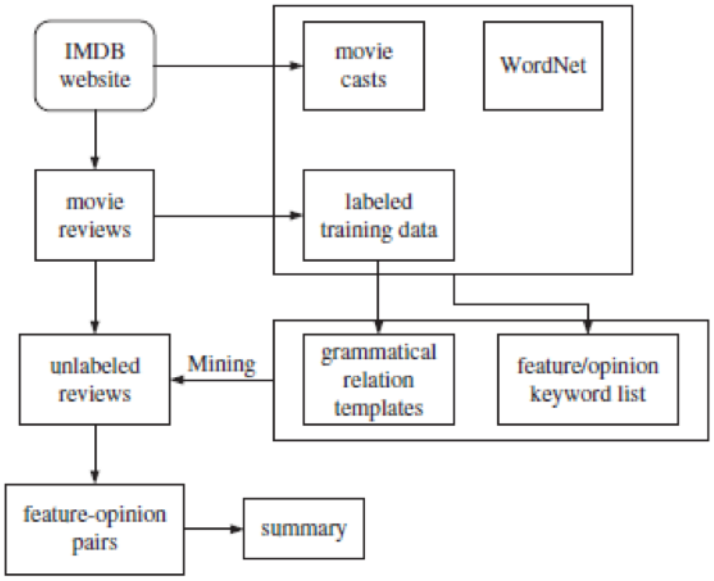


Figure 1: Architectural overview of multi-knowledge based approach

Use dependency grammar graph to detect the path between each feature word and each opinion word.

For the Mining Implicit F-O Pairs:

This problem is difficult, so only deal with two simple cases with opinion words appearing.

Very short sentences that appear at the beginning or ending of a review and contain obvious opinion words.

Ex: "Great!" 11movie-great" or "film-great" Specific mapping from opinion word to feature word

Summary Generation

Collect all the sentences that express opinions on a feature class.

The semantic orientation of the relevant opinion in each sentence is identified.

List the organized sentence as the summary.

3. APPROACH

In this part, we will present our SNB approach in details.

3.1 Sentiment List

First of all, we need to build positive and negative vocabularies dictionary. In some of previous work, they use some existing source dictionaries, such as GI, as their vocabularies dictionaries, but we don't think this is a good idea for our project, since the external dictionaries are not specific for our training set, so there will be many redundant words, which will affect our result and slow down the running time, thereby we decide to get the sentiment list from the specific training set.

The method for getting sentiment list is simple, first define two sets, *posword* and *negword*, and then we just scan the positive and negative review set separately, and each time a

Table 1: Notation used in the report

Notation	Decription
posword	A set that contains all the word vectors from positive training set
negword	A set that contains all the word vectors from negative training set
[*word, count*]	A vector with the actual word and its count
c_p	# of different words appearing in *posword*
c_n	# of different words appearing in *negword*
n_p	# of words appearing in *posword*
n_n	# of words appearing in *negword*
ec_p	# of each word appearing in *posword*
ec_n	# of each word appearing in *negword*
pos_c	words' positive coefficients
neg_c	words' negative coefficients
m	smoothing parameter
tpos_c	summation of words' positive coefficients in a test review
tneg_c	summation of words' negative coefficients in a test review

new word is detected in a positive (negative, respectively) review, then a vector will be created in *posword* (*negword*, respectively), with the format [*word, count*], obviously, whenever a new word is found, the vector is initialized as [*word, 1*]. If a word that is already in the list is detected, the count of this word will be incremented by 1.

Now we have two large sets, *posword* and *negword*, containing all the vectors indicating words that appearing in positive and negative training data, respectively, and their counts. Note, the same word can appear in both of the sets. And then we calculate the cardinality for the two sets, c_p and c_n, which indicate the total number of the different words in both sets. Also, we need to get the summation of the *count* in both sets, n_p and n_n, which presents the total number of words that appear in both sets. Table 1 gives all the notions that will appear in this report and their explanations.

After getting the raw list, we need to prune it. First of all, there are many non-sentimental words should be removed, such as "I", "the" and so on, because they contribute nothing to opinions, but they have large fractions of the whole data, then we just consider such words as noise. And before all the scan part, we have already change all the letters to lower case, so both sets are not case sensitive.

3.2 Positive and Negative Coefficient

Now we have got the final dictionary for our project, then we need to calculate the positive and negative coefficient for each word in the list.

In our method, for each word, we define two coefficients, *pos_c* and *neg_c*, which indicating the "contribution" each word makes to positive reviews and negative reviews, respectively.

Definition 1. (Positive Coefficient, *pos_c*). A coefficient for each word that indicates its proportion in *posword.*

$$pos\_c = \log \frac{ec_p + m}{n_p + m \times c_p} \tag{1}$$

Definition 2. (Negative Coefficient, *neg_c*). A coefficient for each word that indicates its proportion in *negword.*

$$neg\_c = \log \frac{ec_n + m}{n_n + m \times c_n} \tag{2}$$

Here we can notice a parameter m, which is the smoothing parameter, and we will discuss it in next subsection. Now let's look at the two equations. Since for each word, we add m, so for the whole set, we have added c_p (c_n respectively) m, therefore we add this term in denominator, which can guarantee the summation of each word's *pos_c* (*neg_c* respectively) is equal to 1.

The reason we take logarithm is because the proportion of each single word in its set will be very small, like 0.000002, and after taking logarithm, we can get a negative number with a large absolute value, which is convenient for our following process.

Using these two formula, we can get positive and negative coefficients for each word in our list.

3.3 Choosing Smoothing Parameter

Now let us show why we need a smoothing parameter. Just imagine, when we need to predict a review using our model, we should consider positive and negative coefficients of the each word in the review. But some words in this reviews may only appear in one of our sentiment list, namely, some words' ec_p or ec_n are equal to 0. Since we need to take logarithm, if ec_p or ec_n are equal to 0, and we don't add m, then there will be a problem. To avoid this, we add a parameter to each words, just pretend we have seen each word more than 0 time in each list.

Obviously, m cannot be very large, otherwise, it will largely effect the real data. Actually, m should be between 0 and 1. We cannot fix the exact value of m at first, so we just use a loop to choose m, and each iterate m is incremented by 0.01. And then we choose the m that can perform the best result.

At last, we have fix m to 0.84, which can give the highest accuracy.

3.4 Prediction

Finishing all the procedure above, now we can predict sentiment for new reviews as following.

For each word in the test review, we know its positive and negative coefficient, so now we add up all the words' positive and negative coefficients separately, getting *tpos_c* and *tneg_c*. If *tpos_c* is larger than *tneg_c* for this review, then we predict it is a positive review, and vice versa. Since the value of *pos_c* and *neg_c* before logarithm are very small, so the probability for *tpos_c* is equal to *tneg_c* of a review is very low. When this draw happens, we will regard this review as positive.

The whole approach for our method is illustrated in figure 2.

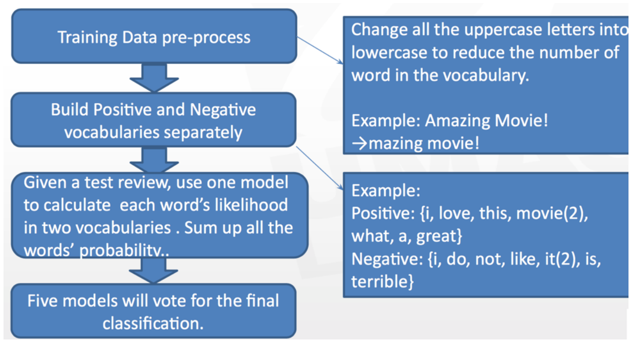

Training Data pre-process
Build Positive and Negative vocabularies separately
Given a test review, use one model to calculate each word's likelihood in two vocabularies . Sum up all the words' probability..
Five models will vote for the final classification.
Change all the uppercase letters into lowercase to reduce the number of word in the vocabulary.
Example: Amazing Movie! →mazing movie!
Example:
Positive: {i, love, this, movie(2), what, a, great}
Negative: {i, do, not, like, it(2), is, terrible}

Figure 2: Approach for Smoothing Naïve Bayes

3.5 Example

For a better explanation of our approach, we will give a simple example here to make it clear.

Assume now we have finished the scan and prune, getting two sets, *posword* = {[best,3], [love,5], [stunning,1], [amazing,4], [extraordinary,4], [oh,6], [good,6]}, and *negword* = {[horrible,3], [embarrassing,5], [pathetic,2], [bad,10], [oh,3] }. In this case, $c_p = 7$, $c_n = 5$, $n_p = 29$, $n_n = 23$, and $ec_p(\text{oh}) = 6$, $ec_n(\text{oh}) = 3$. Assume we choose $m = 0.5$, then $pos\_c(\text{oh}) = \log \frac{6+0.5}{29+0.5\times 7} = -0.699$, and $neg\_c(\text{oh}) = \log \frac{3+0.5}{23+0.5\times 5} = -0.862$. So we have got the two coefficients for word "oh", and from the value we can see in our example, "oh" contributes more to positive reviews more than negative reviews, since $pos\_c(\text{oh}) > neg\_c(\text{oh})$. In the same way, we can calculate all the other coefficients for other words. Whenever, we want to predict a test review, we should add up all the positive and negative coefficients, and compare them to get the label.

4. DATASET

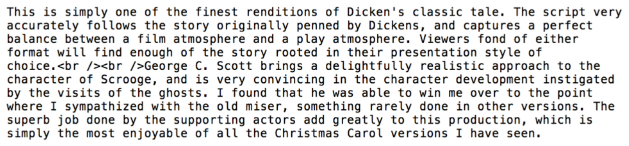
The dataset we use is from "Kaggle" website. It provides the training data, test data and output sample.

In training data, the reviews are classified into two folders, postive and negative, and there are 12501 positive reviews and 12501 negative reviews in total.

For test data, there are 11000 reviews going to be predicted.

And from the sample output, we know our result should be with format [id, label].

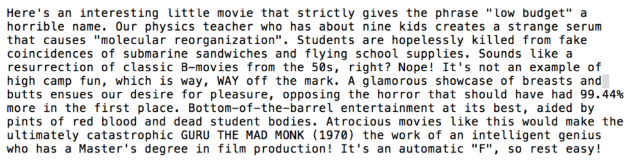
Figure 3, 4 and 5 give the examples of our dataset.



```
This is simply one of the finest renditions of Dicken's classic tale. The script very
accurately follows the story originally penned by Dickens, and captures a perfect
balance between a film atmosphere and a play atmosphere. Viewers fond of either
format will find enough of the story rooted in their presentation style of
choice.<br /><br />George C. Scott brings a delightfully realistic approach to the
character of Scrooge, and is very convincing in the character development instigated
by the visits of the ghosts. I found that he was able to win me over to the point
where I sympathized with the old miser, something rarely done in other versions. The
superb job done by the supporting actors add greatly to this production, which is
simply the most enjoyable of all the Christmas Carol versions I have seen.
```

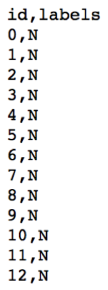
Figure 3: Example for positive training data

5. EVALUATION



```
Here's an interesting little movie that strictly gives the phrase "low budget" a
horrible name. Our physics teacher who has about nine kids creates a strange serum
that causes "molecular reorganization". Students are hopelessly killed from fake
coincidences of submarine sandwiches and flying school supplies. Sounds like a
resurrection of classic B-movies from the 50s, right? Nope! It's not an example of
high camp fun, which is way, WAY off the mark. A glamorous showcase of breasts and
butts ensues our desire for pleasure, opposing the horror that should have had 99.44%
more in the first place. Bottom-of-the-barrel entertainment at its best, aided by
pints of red blood and dead student bodies. Atrocious movies like this would make the
ultimately catastrophic GURU THE MAD MONK (1970) the work of an intelligent genius
who has a Master's degree in film production! It's an automatic "F", so rest easy!
```

Figure 4: Example for negative training data



```
id,labels
0,N
1,N
2,N
3,N
4,N
5,N
6,N
7,N
8,N
9,N
10,N
11,N
12,N
```

Figure 5: Example for sample data

The idea of our method is to find "how much" a word "contributes" to each side, saying positive and negative, and then add up all the "contribution" to get "how much" the target review "contributes" to each side, and then we decide this reviews belongs to the side with more "contribution".

Actually, our idea is very straightforward and sensible, but how well it will perform? So this part we will give the evaluation results coming from our experiments.

As we don't have the correct labels for test data, so we cannot get accuracy using it. Therefore, we use Cross-Validation to evaluate our method.

First, we use the first 90% training data as our training data, and the rest 10% labelled data as our test data, to get our prediction, and then compare with the real label with test data to get our accuracy.

In this process, one problem is to choose m. As we mentioned above, we used a iterate to get the m which can achieve the highest accuracy. At last, we found when $m = 0.84$, the result is optimal. Figure 6 illustrate the process of choosing m.

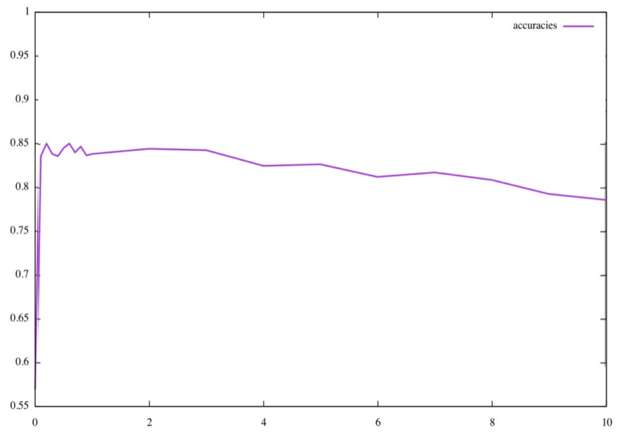


Figure 6: Process for choosing m

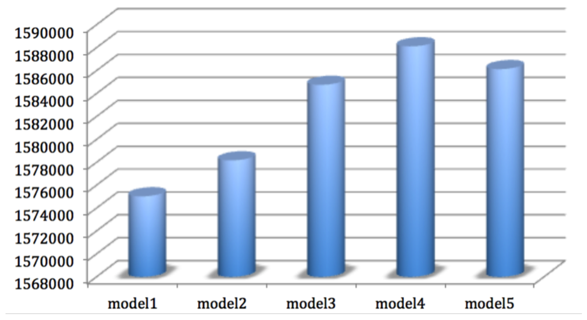


Figure 7: n_p for each model

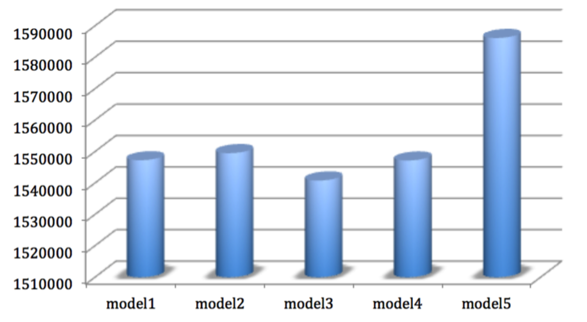


Figure 8: n_n for each model

In our first Cross-Validation process, we get 86.044% for accuracy, and our train error is 9.28%.

For further evaluation, we did five more experiments. For the fist one, we chose the first 20% training data as test data, and the rest 80% as training data. For the second time, we chose the second 20% training data as test data, namely, the 20% - 40% data, and the rest data as training data, and so on, we did five models of Cross-Validation experiments. Figure 7 - 10 give the data for each model experiments.

After the Cross-Validation experiments, we tried to submit our result to the competition website to see the accuracy. Actually, the accuracy we got from the website was lower than our Cross-Validation results. This is not surprising because in our process, we get the sentiment list from our training data, so in the Cross-Validation process, our sentiment list will cover almost all of the words that appear in test data. However, for the test data the website gives us,

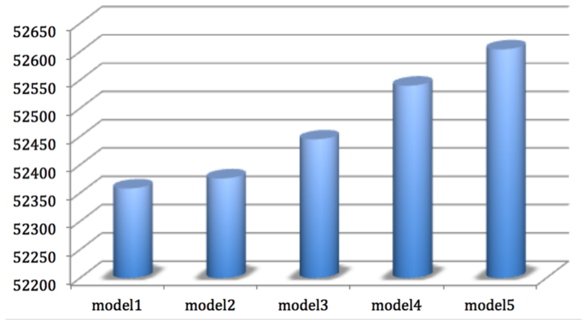


Figure 9: c_p for each model

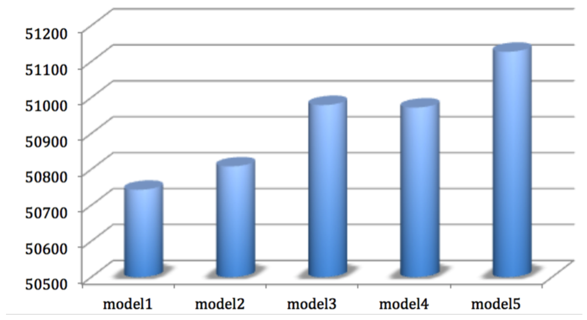


Figure 10: c_n for each model

there will be some words that are not in our training data, as a result, they will not appear in our sentiment list, so we cannot give a sensible positive and negative coefficients to them, and because of smoothing parameter m, they will get misleading coefficients. Hence, the accuracy will be lower in website.

At last, we got our final accuracy 83.291%, with the smoothing parameter $m = 0.84$

6. CONCLUSION

Through this project, we have got familiar to a new model of Naïve Bays classification - smoothing Naïve Bays. With adding a smoothing parameter m, we can avoid error in taking logarithm and hold the accuracy in the meanwhile.

Actually, from other teams' results, we learn that if we choose to use SVM, we may get a better result, but even this, we don't regret because our method is a new thing for us. And the good thing is our result is only a little lower than the top-1 team's.

The reason our method is not good enough is because our formula for positive and negative coefficients is not complex enough, if we can add more useful parameter into it, the result may be more accurate.

Another reason is we just add up the coefficients for each test review, but this process will not make sense sometimes. For instance, from training set, word "not" may get a larger *pos_c* than its *neg_c*, and word "good" is the same. When the phrase "not good" appears, which is a obvious negative comment, using our method, we will regard it as positive because its *pos_c* is larger. So if we want to improve our performance, we need to alter this part, using a more sensible process instead of simply adding up the coefficients.

Another potential way to solve the phrase problem mentioned above may be to get sentiment list every two or more words, then we will have a more complex dictionary, and the result will be better. However, this will cause a higher cost time for running.

7. TEAM ROLES

During the research part, all of our team mates participated into research. Each of us have found some useful related work to support our project.

Since we have enough time to finish the project, so we first tried some models. After the research, Xinzi Sun is responsible to implement Counting method, Lijian Wan is responsible to SVM, and Xuanming Liu is responsible to Logistic regression.

After comparing all the method we know, we decided to use smoothing Naïve Bays, which is interesting and new to us.

Lijian Wan did the pre-process to the data, and Xinzi Sun did the implement of the main formula, and Xuanming Liu did the test to get results.

For the presentation and report part, Xinzi Sun is responsible to prepare for the slides, Lijian Wan gave the presentation, and Xuanming Liu is responsible to finish the final report.

All of us have learnt some thing from this teamwork, and we all have had a great time during the cooperation.

8. REFERENCES

[1] Thanh Vu Dai Quoc Nguyen, Dat Quoc Nguyen and Son Bao Pham. Sentiment classification on polarity reviews: An empirical study using rating-based features. In *roceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 128–135, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[2] Alistaira Kennedy and Diana Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125, 2006.

[3] Li uang, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 43–50, New York, NY, USA, 2006. ACM.