CIS520 Final Project Report

Team **Stella** {Sung Joon Kim, Soohyun Lee, Seunghoon Park}

Department of Computer and Information Science
University of Pennsylvania

{ksung,soohyunl,seupark}@seas.upenn.edu

1 Introduction

After we beat the baseline Naive Bayes model Quiz score of 1.4171 RMSE, and reached the minimum Quiz score threshold of 1.05 RMSE using liblinear library, we tried out several different approaches we learned from class: SVM, random projection, bigrams, and generative approach with different combinations of parameters. Through these different combinations of runs, we have had diverse range of RMSE values and accuracies, and we analyzed the behavior of our approaches according to these various results. More detailed explanation and analysis can be found in the upcoming sections.

2 Methods

2.1 SVM

2.1.1 Approach

For this approach, we used the liblinear library on the basic data X composed of 62,771 reviews and 89,259 word counts on each review, different solvers and used 5-fold Cross-Validation to find the best parameters.

2.1.2 Analysis

Type of Solver	RMSE from the leadearboard
L2-regularized L2-loss support vector classification (dual)	1.1147
L2-regularized L1-loss support vector classification (dual)	1.0231
multi-class support vector classification by Crammer and Singer	1.2089
L1-regularized L2-loss support vector classification	1.08

The solver 'L2-regularized L1-loss' had better (lower) RMSE compared to that of 'L2-regularized L2-loss support vector classification (dual)'. We came to this conclusion because we tried several different distance measures, and this happened to be the best fit for these bag-of-words features that are just the numbers of word occurrences.

2.2 PCA using Random Projection

2.2.1 Approach

First of all, it should be noted here that if we use PCA method to reduce the feature dimension, the data X, a N-by-M matrix where N is the number of examples and M is the number of features, becomes a non-sparse matrix. This is a big problem from the point of view of implementation, since we used the property of sparsity from the basic bag-of-words features to minimize the usage of memory and to speed up using liblinear library instead of libsvm library. For example, a basic X for training data using bag-of-words features is a 62,771x89,259 sparse matrix using 64MB. Let's suppose we reduce the number of features to 9,000 which is about the one tenth. Even in this case, we need to store a 62,771x9,000 non-sparse matrix that needs 62,771 * 9,000 * 8 bytes = 4.2GB which is too large to deal with it on our BIGLAB machine and it also takes too long to learn a model since we can't use liblinear anymore. Thus, we decided to reduce both the number of training examples and the number of features.

To reduce the number of examples, we incorporated the 'helpful' data given in the training set. Only 26,719 reviews had a ratio of 1,the number of positive votes divided by the number of total people who voted. This is fine from the point of view of memory usage, but we realized that it takes at least a few days even after extremely reducing the feature dimension to 300. Thus, we applied one more heuristic which is to sample a half of them based on the same distribution of the ratings of 26,719 examples.

Before applying the random projection algorithm directly to the matrix with the reduced number of examples, we selected the first 6,900 features that appear at the beginning of the list in decreasing order based on the weight values that can be achieved by learning a model using SVM on the basic matrix.

To reduce the feature dimension further, we applied Random Projection algorithm to the matrix X achieved after the above two pre-prosessings. A brief explanation for the procedures is the following: we compute the production XR, 13,360x300 matrix, where R is a random 6,900-by-300 matrix by sampling numbers randomly from the normal distribution with the mean of 0 and the variance of 1 and normalize each column to have unit length.

2.2.2 Analysis

The result of this approach, RMSE=1.2206 from the leaderboard, was worse than learning a SVM model on the basic 62,771x89,259 that gives RMSE=0.9422. To figure out why this didn't improve the performance, we inspected the data deeply and made some charts to visualize it.

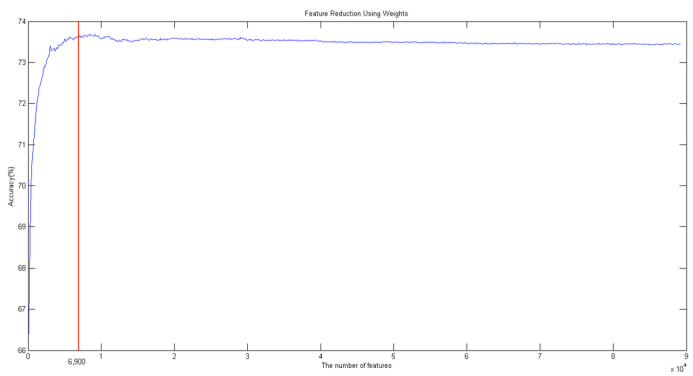

Figure 1(a)

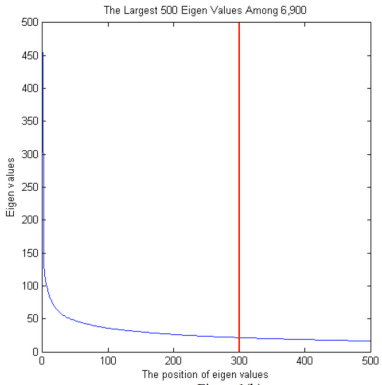

Figure 1(b)

Figure 1

The Figure 1(a) shows the accuracy from the 5-fold Cross-Validation using only a few top features with the largest weights from the model learned by SVM on the basic matrix X. As you can see, about 80,000 words after the first 6,900 words don't improve accuracy so much. This is why we discarded the rest of words. The Figure 1(b) shows the largest 500 eigen values of 6,900 words from SVD. Similarly, 300 is enough number of features to capture the principle components.

'Excellent' 'Great' 'Best' 'stars' 'Good' 'waste' 'Perfect' 'worst' 'Awesome' excellent' 'Works' 'Other' 'best' 'Love' 'Poor' 'poor' 'amazing' 'terrible' 'junk' LOVE' 'complaint' 'Otherwise' 'Overall' 'love' 'loves' 'Disappointed' 'Horrible' 'However' 'returned' 'return' 'Not' 'useless' 'awesome' 'Outstanding' 'perfect' 'Fantastic' 'Amazing' 'fantastic' 'GREAT' 'Wonderful' 'great' 'disappointing' 'disappointment' 'returning' 'Highly' 'disappointed' 'horrible' 'Thank' 'Easy' 'pleased'

The above word list shows the top 50 features with the largest weight values among those 6,900 features. The words seem to be reasonable to represent the good or bad reviews. One thing found from the list is that there are multiple words in the list meaning that the basic vocabulary is case-sensitive, so we could merge these words into one feature if the more time is given to us.

2.3 Bigrams

2.3.1 Approach

Since we are dealing with reviews, which are basically texts, we thought using the information about relationships between adjacent words would be helpful when it comes to classifying arbitrary (unseen) reviews.

Our first and straightforward approach was to use the liblinear library and sparse matrix just like we did before, only by replacing the word_count and word_idx with those of bigrams. With this approach we got 38.12% accuracy with 1.06 RMSE. This result was somewhat better than what we had expected, but it seemed we can improve this with more finer tuning of the data.

The next approach we took was to use bigrams information along with word_count. The reason we decided to incorporate word_count is to improve the RMSE because we already had better RMSE only using the word_count and word_idx for the checkpoint. Using this information and combining with bigrams information seemed promising because we are using extra information based on the already sound approach. The result was quite similar to the approach without using the word_count.

Based on these, we thought we needed some more information to boost up the performance. The next idea we had was to use the title information because title can be seen as a summary of the content, and this will certainly be helpful when the classifiers are trained. So, by then, we used bigrams, word_count, and title. The accuracy turned out to be 38.19%, which is slightly higher (but almost the same) as the previous approach and the RMSE turned out to be 0.9916, which was significantly better (lower).

In this approach, we did not normalize each column, and thought normalizatoin itself may affect the result. So we came up with a pseudo-normalization approach, by dividing each value of that column with the largest value of that column. The result was a slight improvement of RMSE of 0.9243.

2.3.2 Analysis

The first approach of replacing word information with bigram information was the base case for this approach. Unfortunately, the second approach of using additional word_count based on the first approach did not boost the performance. The accuracy was almost the same, as well as the RMSE. We were expecting the results to be better, but then realized there were reasons behind the results. We looked into the data and soon realized that most of the occurrence of pair of words are usually one. And even the ones that occur more than once had nothing to do with the positive/negative words; the most common words were ‘this is’, ‘this product’ or category-specific word pairs. We concluded that we over-estimated the impact of word pairs.

However, we thought having some additional information may boost up the performance, either accuracy or RMSE. The next thing we thought to be reasonable was using the title as part of the training data, maybe as an extension of the context. This is because people tend to summarize their reviews in the title, and the words that will play a critical role in classifiers is highly likely to be included. We believed that even if we don’t extract useful bigram information from reviews, we may have some good/useful words included in the title. This happened to be on the right track of improving the performance, and we obtained much better (lower) RMSE.

2.4 Generative

2.4.1 Approach

Both the data with and without the bigrams were used separately to generate predictions for ratings of the reviews.

A unique part of this approach was to reconsider the multinomial classification into a binary classification. Instead of predicting 1, 2, 4, or 5, we determined if the

reviews were rated good(4 or 5) or bad(1 or 2). Then we gave good reviews a prediction of 4.5 and bad reviews a prediction of 1.5. In this way, if we can perfectly determine if a review is good or bad, our RMSE will be 0.5, which is way below the RMSE of the leaderboard.

However, using bigrams only gave about 85% ~ 90% accuracy on determining bad/good side. Since the cost of mistake is high, we tried to reduce the error by combining generative approach with the result from SVM.

Also, to correct where original generative approach made mistakes, we used some of the testing data as a training set for better prediction.

To further reduce the high cost of mistakes made by original approach, we used quadgram to determine roughly where our classifier made error with bigram.

2.4.2 Analysis

Original generative approach gave RMSE of 1.12. This meant 80 ~ 90% accuracy. When SVM replaced some of the results of generative approach, about 3900 reviews were changed in their sides, and it produced RMSE 0.944. This meant that about 2500 reviews were changed correctly, and the remaining 1400 reviews were not supposed to be changed.

To correct only the reviews that are supposed to be corrected, we tried using some of the testing sets as training sets to learn more bigrams along with the predictions on those testing sets. This way, we were able to correct about 1000 reviews that were supposed to be corrected, which was equivalent to about 2% of entire errors on the test set because there are about 5000 reviews that we made error on.

To seek out the remaining errors, we used quadgram. Running quadgram was slow so it was best for seeking out little portions of the test sets. We averaged the results of the quadgram to that of SVM. Also we just guessed ‘3’ when quadgram was really unsure to avoid making high RMSE because we saw that SVM was unsure on those reviews, too. The total number of changed ratings was about 4000. Some of them included ratings of 3. Few of them were incorrectly changed.

We tried using the test set as training set to seek out the remaning errors. This effort, however, failed miserably maybe due to the errors in the ‘training set’ made from the test set. Almost all the changes made by such approach outside the portion of 1000 reviews were incorrect.

The final RMSE of the combined methods was 0.9226, which meant that we reduced the cost of the mistakes from the original generative approach. The benefits of looking at the problem as a binary classification was gaining insights on how much we were good at classifying the data into good or bad side, and also being able to hypothesize on what RMSE we are going to get just using calculations and without any cross-validation.