CIS 520 Final Project Report

pleyelp
Naicheng Zhang, Zimeng Yang, Ye Yang

Dec 10th, 2013

1 Introduction

For the final project of machine learning class, we used Yelp review dataset and additional features to developed and tested various learning algorithms to predict the star rating of restaurant review.

2 Features

2.1 Stop Word Removal

We implemented a set of stop words[1] and removed them. The removal reduces RMSE across almost all of the classifiers.

2.2 Low Document Frequency Term Removal

Removing words with low document frequency (i.e. those words present in less than 3 documents overall) may reduce RMSE. In the implementations, we removed words that don't exist in the training data set. This affects performance positively.

The above two removal steps reduced the size of dictionary from 56835 tokens to 44368 tokens.

2.3 Word Stemming

We implemented word stemming using the Matlab porterStemmer function[2]. The stemmer was used on the removed dictionary. This process effectively reduced the size of the dictionary from 44368 tokens to around 30674 tokens. However, the stemmer does not reduce the RMSE and takes too long to generate the new feature vector for about 0.2s per review. The technique was not used in our final implementations.

2.4 Mutual Information

More features doesn't mean better prediction. Lots of features are redundant while time consuming. Mutual information is a widely used technique for feature selection which measures the dependence between two variables. We applied MI to select a small subset of features that carries as much information as possible. MI is calculated in[4]:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right),$$

Where Y denotes the restaurant rating labels and X denotes the word counts.
To select the number of features(K) which can minimize the error, we varied K from 200 to 2000 with step size 10 and measured the errors using different classifiers. Please refer the table 1 to check the specific K value we applied in each different classifier.

2.5 Clustering the Words

Based on the results of feature selection, we clustered the words from vocabulary and used this way to produce new features. First we selected a certain number of the most important features, and looked at each word as an observation, while using its occurrences in each review as its features. Then we gathered the words into k clusters. By adding together the features of the words in the same cluster, k new features were produced.

To visualize how this is working, we chose the top 300 words and clustered them into 10 groups. Figure 1 and figure 2 show the most representative word clusters among them. The font size is corresponding to each word's distance to the centroid of the cluster. It can be seen that clustering words is an effective way to gather similar features.

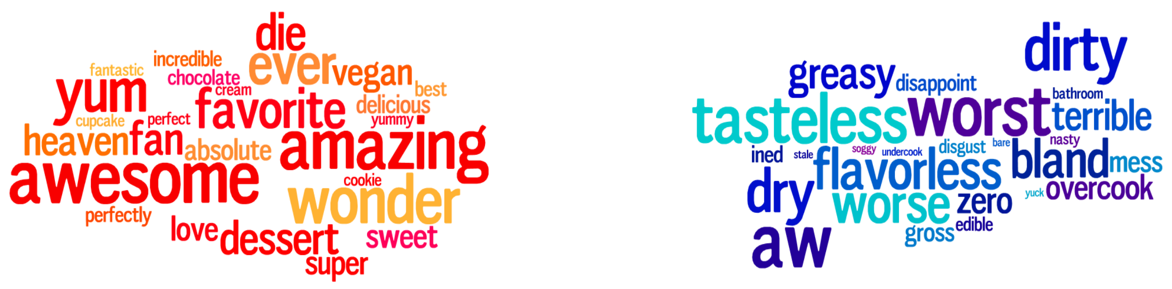

Figure 1 Cluster Samples

2.6 Bigram

In text based analysis, bigram is very useful for its ability to capture some combinations of modified verbs and nouns. We can improved the predicting performance by introducing bigram as an additional feature. But to find the bigrams in the texts, we have to scan the texts every time which takes lots of time. Due to the time limitation, we gave up the bigram.

2.7 Additional Features

We used the votes on cool, funny and useful and restaurant ID to generate new features. For votes on cool, funny and useful, we generate a 1x3 feature vector for each data. The entries are the number of votes. For restaurant ID, given a test data, we examine the restaurants in the training set whose IDs are the same as the test data and average those ratings. In addition, we count the number of words of each review using the given counts matrix as an additional feature.

3 Implementations

3.1 Naïve Bayes

We first tested using the built-in Naïve Bayes function. The built-in function gives the argmax prediction. We rewrite our own Naïve Bayes function which turned out to be the same as the built-in function and can output the probabilities of each rating.

Our Naïve Bayes uses multinomial event model and Laplace smoothing[4]. Given a set of m independent observations {z(1), ... , z(m)}, the maximum likelihood estimates are given by:

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + k}.$$

where k is the number of words in the vocabulary. In the multinomial event model, we assume that the way a review is generated is via a random process in which its rating is first determined. Then, the user writes the review by first generating x1 from some multinomial distribution over words (p(x1|y)). Next, the second word x2 is chosen independently of x1 but from the same multinomial distribution, and similarly until all n words of the email have been generated. Computing the expected rating, the RMSE of Naïve Bayes is 0.9683 which is much smaller than 1.0526 by taking the argmax.

We also tested Naïve Bayes with feature selection. Here we selected k features and plot the RMSE using Naïve Bayes with respect to k. The number of features k with minimum RMSE is around 300.

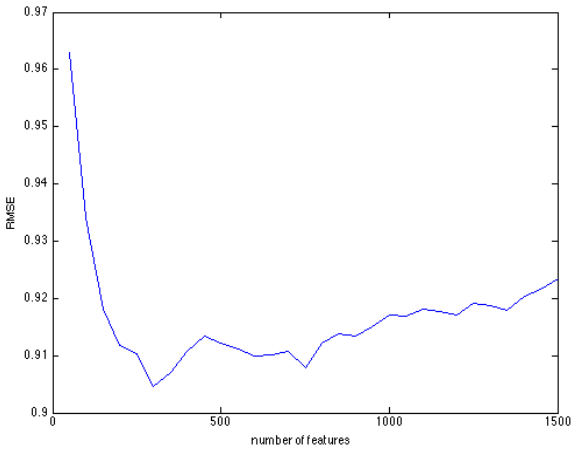

Figure 2 RMSE with different number features using Naïve Bayes

3.2 SVM

We used Liblinear to implement SVM because Liblinear is much faster than LibSVM on this dataset. The Gamma parameter is determined by the cross-validation. We trained Gamma on each of feature selected counts matrix. The Gamma is around 0.005 for no feature selection and 0.1 for selecting 700 features. Unfortunately, Liblinear doesn't support probability output for SVM. So we cannot compute the expected rating as the prediction.

3.3 Logistic Regression

We also used Liblinear to implement logistic regression. We also trained Gamma for logistic regression. Because Liblinear supports probability output for logistic regression, we ca

3.4 Linear Regression

We also implemented linear regression on the dataset of the counts matrix, which turned out to be efficient, and the RMSE could be reduced to 0.9379.

By doing cross validation we computed RMSE on different number of features and chose the dataset yielding best results. Figure 4 and figure 5 each shows the RMSE respect to k features without using word stemming and using word stemming. The number of features k with minimum RMSE is around 800 on the dataset without using word stemming.

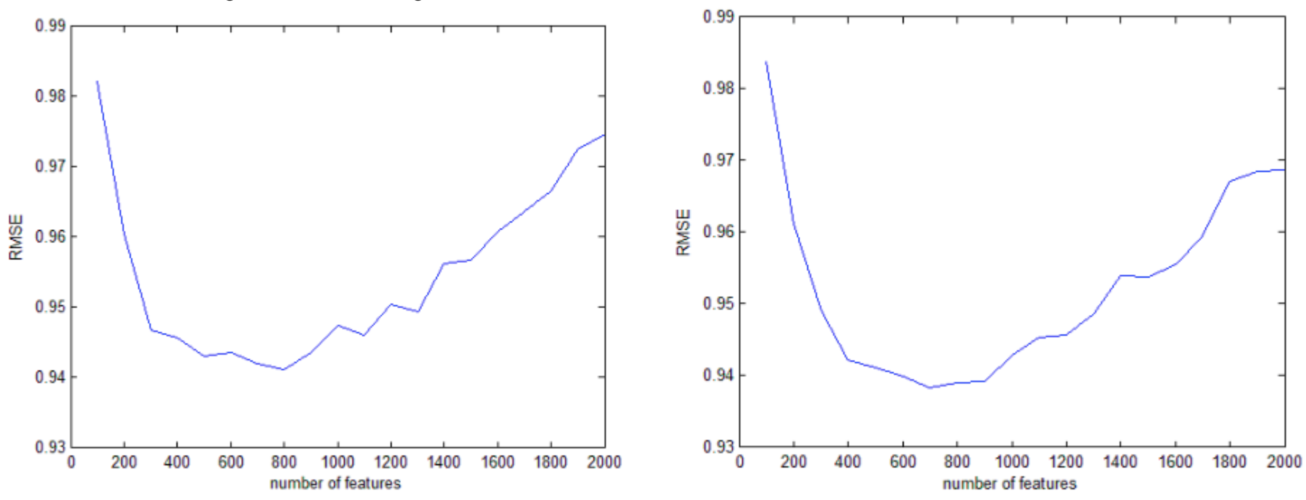

Figure 3 RMSE with different number features using Linear Regression with and without word stemming

3.5 k-NN

We also used k-nearest neighbors to classify each test data by averaging the labels of the k closest training data. For the distance measurement, we tested with euclidean, cosine and correlation. However the results were not satisfactory. It is probably because neither of these distance metrics could represent the real model nicely. Thus we did not include this model in the final implementation.

3.6 K-means

We implemented K-means by running this method on the training set and the test set together and gathered them into 5 clusters, in corresponding to 5 labels. For each test data, find which cluster it belonged to and decide its label by the voting of the training data in the same cluster. However this model did not work well. After looking into the data, we found that the occurrences of label 4 were remarkably larger than label 1, 2, 3, and 5. Thus all 5 clusters were labeled as 4 in this way, which leaded to wrong classifications for most of the test data.

4 Result and Analysis

We compute the *expected rating* by computing a weighted average according to the distribution of the possible ratings 1, 2, 3, 4, and 5 stars. This more accurately represents the guess of the classifier than taking the argmax, and reduces RMSE.

Classifiers+Features	RMSE
NB+BW (argmax)	1.0526
NB+BW (expected rating)	0.9683
NB+WR (expected rating)*	0.9583
NB+FS 310 (expected rating)*	0.9032
NB+CFU (expected rating)*	1.1969
SVM+BW (argmax)	0.9857
SVM+WR (argmax)*	0.9986
SVM+FS 700 (argmax)*	0.9438
SVM+CFU (argmax)	1.5432
LOGISTIC+BW (expected rating)	0.9875
LOGISTIC+WR (expected rating)*	0.8846
LOGISTIC+FS 1400 (expected rating)	0.8593
LOGISTIC+CFU (expected rating)	1.5419
LR+RL*	1.2116
LR+FS 800*	0.9379
K-NN	1.3901
K-means	1.3566

Table 1. Quiz set test result. NB - Naive Bayes, SVM - Support Vector Machine, LOGISTIC - Logistic Regression, LR - Linear Regression, BW - Bag of Words, WR - Words Removal, FS 310 - Best 310 features selected using MI, CFU - Votes on cool, funny and useful, RL - Review Length (# of words), * - Included in our final implementation

The most interesting outcome of our project was the words clustering. While clustering did not decrease our test set error, its word associations proved interesting in their own right and worthy of further study. Naive Bayes and SVM are always two baseline methods for text classification. Without computing the expected rating, SVM performs better than Naive Bayes in our dataset. Our hypothesis for the reason is that the Naive Bayes's assumption that the probability of each word appearing is independent of whether other words are in the text may not be true in reality and affect the performance. The expected rating reduced the RMSE significantly. The stop words removal reduced RMSE across almost all of the classifiers. The reason for that may be that the stop words in the text are irrelevant to the rating of the review and affect the performance negatively.

Our final submission was a linear combination of the predictions by some of the methods above. We trained two parameters for the linear regression. One is for that including business ID and the other not including business ID. Once we get the test data, we check its business ID and search through the training set for the same ID to decide whether to use the linear combination of methods including business ID and which parameter to use.

References

[1] https://amplab.cs.berkeley.edu/2012/05/05/an-nlp-library-for-matlab/
[2] http://tartarus.org/~martin/PorterStemmer/
[3] http://en.wikipedia.org/wiki/Mutual_information/
[4] http://cs229.stanford.edu/notes/cs229-notes2.pdf