Predicting Usefulness of Yelp Reviews

Xinyue Liu, Michel Schoemaker, Nan Zhang

1 Introduction

Yelp offers users with a myriad of reviews and ratings of businesses all over the world. It also provides users with the ability to "up vote" a review as useful, funny or cool, with some particular reviews being heavily up voted as useful by the Yelp community. Yelp uses a proprietary algorithm to determine the display order of reviews for a particular business. The particular details of this algorithm are kept under wraps, though Yelp acknowledges it is based on "various measures of quality, reliability, and activity on Yelp". A quick analysis seems to reveal that these usefulness scores do not seem to heavily influence the display order of reviews. Based on this fact, our team ponders the question of whether the words used in a review correlate to its usefulness, and whether we can predict if a review will be deemed useful by Yelp users (in particular, within the top 25% of votes). Using the Yelp Dataset Challenge Database, we choose 1000 random reviews to perform an analysis. After performing numerous natural language processing techniques on our data set, we obtain a feature matrix on which to predict our binary response. We apply feature selection methods to our matrix, build five different models, calculate the training and test errors and analyze our results. Finally, we extract the top 10 indicative words of a useful review.

2 Prior Work

In the past, Yelp has posed the question of "How many 'useful' votes will a Yelp review receive?" as a recruiting competition aimed at engineers interested in working at Yelp [1]. Though most submissions are kept private, we find two relevant papers that attempted to answer his question. The first paper is titled "Predicting the Number of "Useful" Votes a Yelp Review Will Receive" [2], and uses multiple features such as star rating, length of review and a small 19-word feature matrix for an error of approximately 0.52. The model is good at predicting votes up to the number of 8, but falters after that. The second paper, "Exploring the Yelp Data Set: Extracting Useful Features with Text Mining and Exploring Regression Techniques for Count Data" [3] uses a myriad of features and models and had a RMSLE ranging from 0.45 to 0.73. All these analyses, however, fail to account for the fact that reviews for restaurants in highly populated and urbanized areas will overall have higher numbers of useful votes. Our approach is different in that it looks at usefulness as a binary response, in which the true value accounts for the top 25% useful votes within a particular business.

3 Data

We obtained our data from Yelp on the Yelp Dataset Challenge website [4]. The data is a collection of .json files with information on Yelp users, businesses, reviews etc. Since we are interested in review text, we obtained all of our data from the file that contains data on reviews (named "yelp_academic_dataset_review.json"). It is formatted as follows:

```
{
   'type': 'review',
   'business_id': (encrypted business id),
   'user_id': (encrypted user id),
   'stars': (star rating, rounded to half-stars),
   'text': (review text),
   'date': (date, formatted like '2012-03-14'),
   'votes': {(vote type): (count)},
}
```

The file contains 1,125,458 reviews, each with the text string and "useful" vote count. It should be noted that The Challenge Dataset only includes data from Phoenix, Las Vegas, Madison, Waterloo and Edinburgh.

3.1 Feature Matrix

For the purpose of our project, we randomly choose 1000 reviews out of all reviews to perform our analysis. After examining the reviews that were randomly chosen, we discover that 3 reviews contain no text in them or only contain punctuations in them. Therefore we choose to discard these 3 reviews. Next, we attempt to clean the data up to get a dictionary of words that will be used as our features. We extract all the words that appeared in the 997 reviews, and count the number of times each word appeared in total. After taking a look at the words we obtained, we see that many words lack spaces between them (e.g. "potatosquishy", "chickenjoy"). We split them by comparing the Levenshtein distance between words, and obtain a second dictionary with correctly spaced words. Because there exist many non-useful "stop words" (such as "the", "a") in the dictionary that are not desired features for text classification, we use an NLTK package [5] to delete all stop words from our dictionary. Since different variations of a word sometimes have the same meaning (e.g. "eat", "ate", "eaten"), we use an NLTK package to lemmatize the words in our dictionary. Finally, we discard words that appear less than 10 times, since words that appear infrequently are less useful for prediction. The final version of our dictionary contains 170 words.

After obtaining our dictionary of words, we create a sparse matrix with 997 rows and 170 columns. Each row represents a unique review, and each column represents a word in our dictionary. The (i, j)th entry in the sparse matrix is the number of times the jth word appeared in review i. The matrix is saved in a .mat format that can be readily loaded into Matlab.

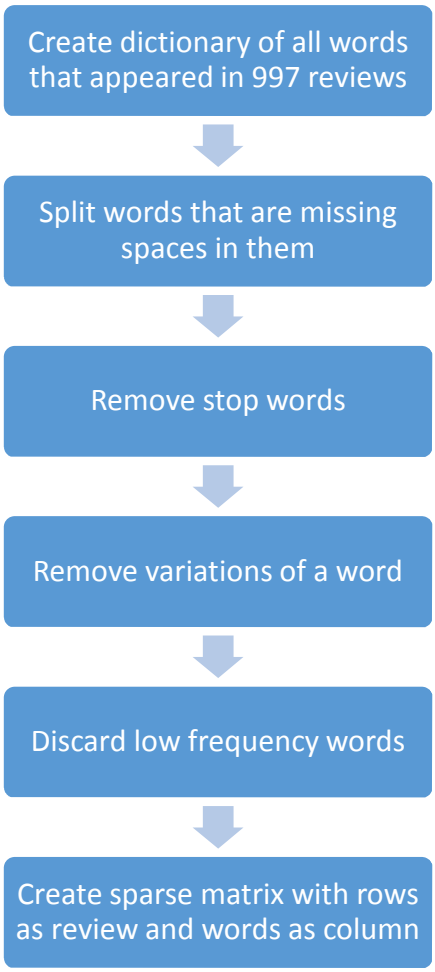

Figure 1. Process of creating feature matrix

3.2 Response Vector

Each review on Yelp receives a number of "usefulness" votes. We wish to turn the number of "usefulness" votes a review receives into a binary variable with 1 indicating the review is useful and 0 otherwise. For each business, we order the reviews by how many "usefulness" votes they received, and we consider the top 25% reviews as useful, and the remaining 75% reviews as not useful.

4 Feature Selection

We use holdout cross validation and randomly split our data into 70% training set and 30% test set. To reduce the dimension of the feature matrix, we apply principal component analysis (PCA) to the inputs of both the training set and the test set. We choose 85%, 90%, 95%, 100% to be the percentage of total variance explained by the principal components. The number of principle components for 85%, 90%, 95%, 100% are 32, 50, 81, 170 respectively. Thus we have four pairs of training set and test set with reduced dimension. We train and test our models on these training sets and test sets in order to see how the number of features influences the results for different learning algorithms.

5 Models

As mentioned before, our response variable is a binary and determines whether a review will be voted "useful" or not. We apply five methods to the training sets attained by feature selection: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Naïve Bayes, Logistic Regression, and Support Vector Machine (SVM). For each method we calculate both the training error and test error, which are defined as:

- training error = misclassified training sample/total number of the training sample
- test error = misclassified test sample/total number of the test sample

5.1 Linear Discriminant Analysis (LDA)

We use a MATLAB built-in linear classifier that fits a multivariate normal density to each category with a pooled estimate of the covariance matrix. Specifically, we choose the following MATLAB function:

```
output = classify (sample, training data, training category, 'diaglinear')
```

We use 'diaglinear' as the type of classifier instead of 'linear', because 'diaglinear' is similar to 'linear' but gives an estimate of the covariance matrix that is diagonal. This can solve the problem of "ill-shaped" covariance matrices. The output is the category of the sample predicted by this classifier.

5.2 Quadratic Discriminant Analysis (QDA)

We use a MATLAB built-in quadratic classifier that can separate two classes of events by a quadratic surface. It fits multivariate normal densities with covariance estimates stratified by each category. Similar to what we did for LDA, we choose the MATLAB classify function with type 'diagquadratic' instead of 'quadratic' to avoid "ill-shaped" covariance matrices. We calculate the binary output and the corresponding generalization error from fitting this model.

5.3 Naïve Bayes

We train the model by using a multinomial event Naïve Bayes classifier with Laplace Smoothing. Later we will use this model to find the most predictive words for useful reviews.

5.4 Logistic Regression

For the Logistic Regression model, we select the corresponding coefficients in the following ways:

1. Apply the MATLAB built-in function of Generalized Linear Regression (GLR for binomial responses) to the selected training sets and get coefficient estimates b1.
2. Apply the Lasso algorithm to the training sets and use 5-fold cross validation to select the best regularization parameter lambda. Gain the coefficient estimates b2 under the value of lambda.

We compute predicted outputs for the Logistic Regression with coefficient estimates b1, b2, and attain corresponding training and test errors for comparison.

5.5 Support Vector Machine (SVM)

We use MATLAB's "fitcsvm" function to train a binary support vector machine classifier, using both a linear kernel $K(x_1,x_2) = x_1'x_2$ and a Gaussian (RBF) kernel $K(x_1,x_2) = \exp(-\|x_1-x_2\|^2)$. Then we predict the categories by calling the "predict" function on our trained SVM models.

6 Results

6.1 Training and Test Errors

For all the models, the training and test errors with regard to the percentage of total variance explained by the principal components are presented in Table 1 and plotted in Figure 2.

Model	Error	85% variance explained	90% variance explained	95% variance explained	100%variance explained
LDA	Training	0.2927	0.2841	0.2224	0.1220
	Test	0.3043	0.2642	0.2341	0.2542
QDA	Training	0.4175	0.3099	0.1679	0.0976
	Test	0.4649	0.3512	0.2642	0.1773
Naïve Bayes	Training	0.4261	0.4290	0.4319	0.3859
	Test	0.4214	0.3913	0.3980	0.4013
Logistic	Training	0.0732	0.0717	0.0717	0
	Test	0.0803	0.0803	0.0936	0.2074
Logistic(lasso)	Training	0.0746	0.0746	0.0746	0.0746
	Test	0.0702	0.0702	0.0702	0.0702
SVM (linear)	Training	0.0746	0.0746	0.0746	0.0244
	Test	0.0702	0.0702	0.0702	0.1338
SVM (RBF)	Training	0.0029	0.0014	0	0
	Test	0.0702	0.0702	0.0702	0.0702

Table 1. Summary of all training and test errors

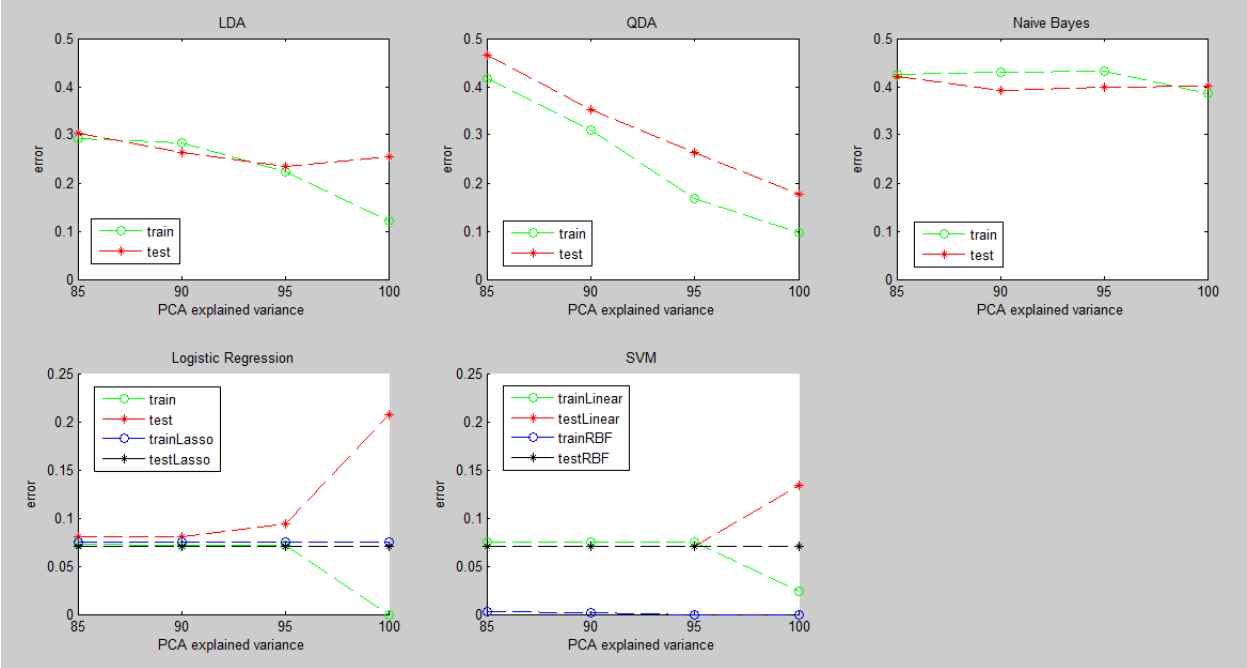

Figure 2. Plotting errors against percentage of variance explained

A rough comparison of the test errors produced by different methods shows that Logistic Regression and SVM do better than LDA, QDA and Naïve Bayes. Both the Logistic Regression (with Lasso) and SVM methods have the least test errors around 7%. We also examine the relationship between errors and percentage of total variance explained by principle components for each model:

- *LDA:* Both the training error and test error go down from 85% to 95% explained variance, with the test error close to training error. However, the training error continues to go down after 95% explained variance while test error goes up, with a large gap at 100% explained variance. This may be caused by high variance when training in the original high dimensional feature space.
- *QDA:* Both the training error and test error go down as the explained variance increases. The large decrease in test error from 85% to 100% explained variance implies that reduced dimensions may lead to underfitting.
- *Naïve Bayes:* Both the training error and the test error are relatively high (around 40%) regardless of the dimensions of feature matrices. It indicates that Naïve Bayes may not be a good model choice for this data set.
- *Logistic Regression:* Using the Lasso method to train the model does better than Generalized Linear Regression (GLR) by keeping a constant test error (0.0702) for all the situations. The original GLR will overfit data in high dimensional feature spaces (e.g. at 100% explained).
- *SVM:* the RBF kernel has lower training error than the linear kernel, while their test errors are the same after reducing the number of features to 95% explained variance or lower. This implies that by properly reducing the number of features with PCA, the type of kernel may not influence the test error.

6.2 Indicative Words for Useful Reviews

By applying our multinomial event Naïve Bayes model to the original feature matrix (997*170) and corresponding response, we found 10 most indicative words for useful reviews: day, show, over, star, thing, free, look, chicken, say, lunch.

7 Conclusion

From our analysis, Logistic Regression (with Lasso) and SVM algorithms can be used to predict the usefulness of Yelp reviews, because they have much lower generalization errors. We also find that most of the time using PCA or the Lasso method to reduce the dimension of the feature space helps solve the problem of overfitting, and thus enhances prediction accuracy. Finally, we find the 10 most indicative words of a useful review. Future work could involve the following parts:
1. Adding more features to our models and analyzing how these extra features influence our prediction of review usefulness. For example, we could consider adding the length of the review or other types of votes the review received from users.
2. Using higher-order regularization methods to train the Logistic Regression model.
3. Using the k-means method to group the reviews into several classes and train corresponding models of each class. To predict the usefulness of a new review, we can first decide which class it belongs to, then using the models for this specific class.

8 References

[1] Kaggle. 2013. "*How many "useful" votes will a Yelp review receive? Show off your skills to land an interview for a position on a Yelp data mining team!"* <https://www.kaggle.com/c/yelp-recruiting>

[2] Luther, Akshay. 2013. "*Predicting the Number of "Useful" Votes a Yelp Review Will Receive*" <http://akshayluther.com/2013/09/08/predicting-the-number-of-useful-votes-a-yelp-review-will-receive>

[3] Anonymous. *"Exploring the Yelp Data Set: Extracting Useful Features with Text Mining and Exploring Regression Techniques for Count Data"* < http://www.cs.ubc.ca/~nando/540-2013/projects/p9.pdf>

[4] Yelp. 2014. *Yelp Dataset Challenge*. Retrieved October 15, 2014. <http://www.yelp.com/dataset_challenge>

[5] NLTK.org. 2014. *NLTK Documentation 3.0*. Retrieved October 15, 2014. <http://www.nltk.org/api/nltk.html>