Predicting Yelp Star Ratings Based on Text Analysis of User Reviews

Junyi Wang

Stanford University

junyiw@stanford.edu

Abstract

We perform sentiment analysis based on Yelp user reviews. We treat a Yelp star rating of 4 or 5 as a positive sentiment and a rating of 1, 2 or 3 as a negative one. Various language models are used to obtain feature vectors and we implement three different algorithms, namely perceptron learning algorithm, Naive Bayes and SVM to predict sentiment. The performances of these three algorithms on this binary classification problem are discussed in this paper.

I. Introduction

Yelp has been one of the most popular sites to find great local business over recent years. People write reviews and give a star rating ranging from 1 to 5 based on the service quality and their own personal preferences. The average star rating is a very important indicator of the quality and popularity of each business. On the other hand, text data are ubiquitous and play an essential role in a variety of applications. Fortunately, Yelp has provided a large dataset containing information about users and businesses. We use the user reviews in the given dataset and build prediction models for ratings. Specifically, we want to predict whether a user is conveying a positive or negative feedback simply by looking at the review texts alone.

II. Task Definition

The goal of the project is to apply machine learning algorithms to predict the Yelp star rating based on the user review. Since the star ratings are not linear, classification results are expected to perform better than regression. To simplify the problem and the implementations of algorithms, we treat a star rating of 4 or 5 as a positive feedback and a rating of 1, 2 or 3 as a negative one. Basically the input is some text of user review, and we want to output a binary value, with +1 indicating a positive feedback and -1 denoting a negative feedback. An example would be as follows.

Input: {"text": "What a cool bar/restaurant.. I will no doubt be visiting again. The service and prices were great.. and the restrooms were clean. I had the buffalo chicken sandwich and it was delicious. The menu consists of typical bar food, however; theres a few different items on there which stand out on the menu. A cool bar off the beaten path that is a worth a trip. Cheers ! !"}

Output:{"sentiment": +1}

III. Related Work

Kou and Wu tried building models to predict Yelp star rating and used the root-mean-square error (RMSE) as the error metric [1]. The best prediction performance with a 0.8223 RMSE is produced by ridge regression with TF-IDF feature selection method. Li and Zhang worked on both binary and 5-class classification problems to predict the star rating [2]. They used the mean-square error (MSE) for evaluation and concluded that the best model is to use Support Vector Regression removing stop word with a test MSE of 0.0179 on the binary classification problem. Xu, Wu and Wang also looked into the 5-class classification problem and did binary sentiment analysis [3]. Their results showed that the perceptron algorithm has better performance than multi-class SVM and Nearest Neighbor on precision and recall, and the prediction results are good for a star rating of 1 and 5, but relatively poor for a rating of 2, 3, and 4.

IV. Dataset and Preprocessing

I. Preprocessing

The dataset used in this project is provided by the Yelp Dataset Challenge [4]. The dataset includes five json files containing information about business, checkin, review, tip and user. Specifically, we use the review dataset and extract the user review and star rating information. As the original review dataset is quite large with 1569264 reviews, we collect a random sample of size 10000. Then we split the 10000 examples by randomly selecting 6000 of them to be the training data and the rest 4000 as the test set.

II. Error Metric

We use precision and recall as the error metric.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

where TP, FP, FN stand for the number of true positives, false positives and false negatives.

In terms of training error and test error, they are defined similarly as

$$\text{Training/ Test Error} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} \neq y^{i}\}}{m}$$

where m is the total size of the training/test data, $y^{(i)}$ and y^{i} stand for the predicted sentiment and the true sentiment on the i-th example, respectively.

V. Methods

We apply three supervised learning algorithms for prediction, namely perceptron learning algorithm, Naive Bayes and SVM. To obtain feature vectors, we use different language models, including basic unigram, basic bigram and character-based n-gram. We can run cross validation on different sizes of n, and find that the optimal n is 5. As a result, for examining the performance of character-based models, we use the character-based 5-gram. We have also tried removing common symbols, removing stop words and stemming. The high-level procedure is shown in Figure 1.

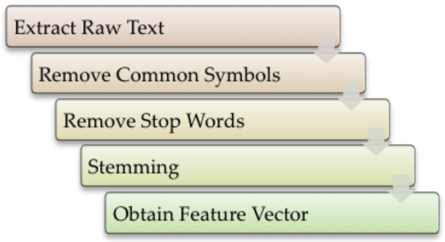

Figure 1: *Procedure of getting feature vectors*

The perceptron learning algorithm and Naive Bayes are implemented in Python, and we use the scikit-learn library to implement SVM [5]. In terms of common symbols, we define the set to be {",!.;?[]()"}. Stop words and word stems are obtained by using the Natural Language Toolkit library in Python [6].

I. Perceptron Learning Algorithm

Let h be a threshold function defined as

$$h(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{if } \theta^T x < 0 \end{cases}$$

where x is a vector of features and θ is the corresponding weight vector.

We apply perceptron learning algorithm and run stochastic gradient descent on the weight vector θ using hinge loss until convergence.

$$\theta \leftarrow \theta - \eta(\nabla_\theta \max\{0, 1 - \theta \cdot \phi(x)y\})$$

Table 1: *Results of Perceptron Learning Algorithm for Various Language Models*

Language Model	Training Error	Test Error	Precision	Recall
Basic Unigram	0.023	0.282	0.749	0.799
Basic Bigram	0.005	0.318	0.697	0.845
Unigram Removing Common Symbols	0.018	0.260	0.788	0.778
Unigram Removing Stop Words and Common Symbols	0.022	0.268	0.759	0.816
Stemming with Stop Words and Common Symbols Removed	0.040	0.228	0.799	0.833
Character-based 5-gram	0.000	0.237	0.774	0.858

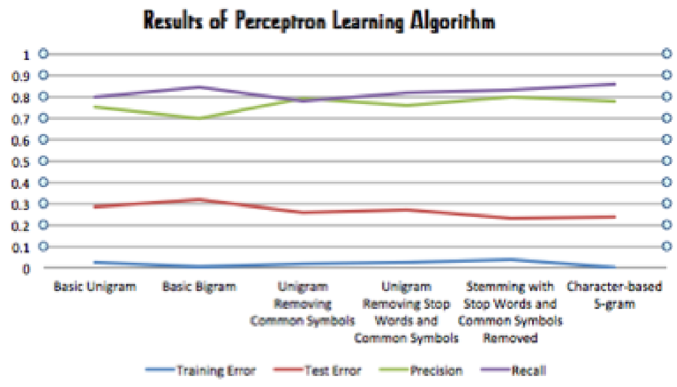

Figure 2: *Comparison of Different Language Models for Perceptron Learning Algorithm*

Specifically, we run cross validation on the learning rate η, and use $\eta = 0.001$ since it gives the lowest test error.

II. Naive Bayes

Once we get the feature vectors from the training examples, we can compute the overall probability of a positive review $p(y = 1)$ and the probability of a certain feature x_i appearing in a positive review $p(x_i|y = 1)$. Similarly for the negative case. Suppose the total feature size is n. Then we can compute the posterior probability by Bayes rule

$$p(y = 1|x) =$$

$$\frac{(\prod_{i=1}^{n} p(x_i|y = 1))p(y = 1)}{(\prod_{i=1}^{n} p(x_i|y = 1))p(y = 1) + (\prod_{i=1}^{n} p(x_i|y = -1))p(y = -1)}$$

We check whether $p(y = 1|x)$ is larger than 0.5, and give the corresponding prediction result. For the implementation of Naive Bayes, we take the logarithm of the likelihood function and use summation for computation to avoid very small number close to 0 produced by the product of a series of conditional probabilities. We also use Laplace smoothing on the prior probability for features that never appear in the training set. The prediction results are summarized in Table 2.

Table 2: *Results of Naive Bayes for Various Language Models*

Language Model	Training Error	Test Error	Precision	Recall
Basic Unigram	0.092	0.460	0.908	0.262
Basic Bigram	0.002	0.408	0.882	0.372
Unigram Removing Common Symbols	0.132	0.454	0.920	0.269
Unigram Removing Stop Words and Common Symbols	0.072	0.404	0.896	0.372
Stemming with Stop Words and Common Symbols Removed	0.116	0.424	0.868	0.349
Character-based 5-gram	0.006	0.464	0.926	0.249

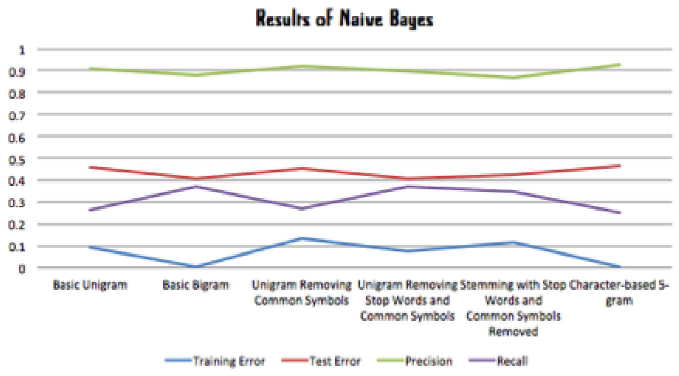

Figure 3: *Comparison of Different Language Models for Naive Bayes*

III. SVM

We define a hyperplane by

$$\{x : f(x) = x^T\beta + \beta_0 = 0\}$$

where $||\beta|| = 1$ and x is the feature vector. The optimization problem of SVM can be expressed as

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N} \epsilon_i$$

subject to

$$\epsilon_i \geq 0, y_i(x_i^T\beta + \beta_0) \geq 1 - \epsilon_i \forall i$$

where y_i is the response variable indicating the sentiment of the i-th example in the training set, and C is the penalty parameter of the error term.

We use the SVC function in sklearn library with a linear kernel and experiment with various values of C. We find that the test error for $C = 0.1$ is the smallest, so we use $C = 0.1$ throughout the process of trying different language models and get the following result.

Table 3: *Results of SVM for Various Language Models*

Language Model	Training Error	Test Error	Precision	Recall
Basic Unigram	0.005	0.315	0.687	0.781
Basic Bigram	0.0033	0.3675	0.624	0.826
Unigram Removing Common Symbols	0.00667	0.3025	0.698	0.790
Unigram Removing Stop Words and Common Symbols	0.00833	0.2925	0.695	0.831
Stemming with Stop Words and Common Symbols Removed	0.01	0.2675	0.730	0.813
Character-based 5-gram	0	0.265	0.722	0.840

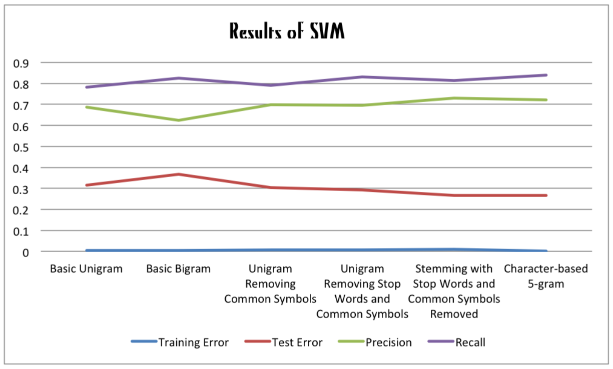

Figure 4: *Comparison of Different Language Models for SVM*

VI. Results and Discussion

We can compare the results in Table 1, 2 and 3. Generally the perceptron learning algorithm has the best performance on various language models. SVM has comparable performance with perceptron learning algorithm, while Naive Bayes does not produce good prediction results. The precision for perceptron learning algorithm typically ranges from 0.7 to 0.8, and the recall from 0.78 to 0.86. For comparison, the precision for SVM is around 0.7, and the recall varies from 0.78 to 0.84.

Notice that the recall for using Naive Bayes is really low, while the precision is very high. This means that the total number of false negatives is high. One possible reason can be that $p(y = -1)$ is high, meaning that the overall probability of getting a negative feedback is higher than the probability of getting a positive one. Another possible explanation is that people who give a negative feedback tend to write longer reviews, so for each feature in the test set, the probability that it appears in a negative review is higher than the chance that it appears in a positive one.

In terms of all the language models that we have experimented, stemming with stop words and common symbols removed gives the best prediction result. It produces the smallest test error of 0.228 if we apply the perceptron learning algorithm. The result is expected as stemming and removing common symbols reduce dimension and the chances of multicollinearity by removing sets of similar features. Removing stop words improves the model by getting rid of non-significant features. In addition to perceptron learning algorithm, SVM with features generated by character-based 5-gram also produces a reasonably good test error of 0.265.

One thing worth pointing out is that for all the three algorithms that we have discussed, the training error is much lower than the test error. This suggests that we might have overfitted the data. To further improve the performance on the test set, we need to add regularization terms and run cross validation on the regularization parameter.

VII. Conclusion and Future Work

In this paper we experimented perceptron learning algorithm, Naive Bayes and SVM to predict sentiment as reflected by the Yelp star rating. Various language models have been used to extract the features from the text of Yelp user reviews. To sum up, the perceptron learning algorithm generally produces the best prediction results and the best language model is using stemming and removing stop words and common symbols.

For future work, we would try regularized perceptron learning algorithm and Naive Bayes to see if that will reduce the test error. We would also explore the 5-class classification problem, other classification methods and more advanced language models. As mentioned in the section of related work, Xu, Wu and Wang did not get high precision and recall for the 5-class classification as it is a much more complicated problem than binary classification that we have discussed in this paper. As a result, it is difficult to compare the our results and theirs.

References

[1] Jiyou Kou, and Luyan Wu. *Building Prediction Models on Yelp Dataset*. 2014.

[2] Chen Li, and Jin Zhang. *Prediction of Yelp Review Star Rating using Sentiment Analysis*. 2014.

[3] Yun Xu, Xinhui Wu, and Qinxia Wang. *Sentiment Analysis of Yelp's Ratings Based on Text Reviews*. 2014.

[4] Data source: `http://www.yelp.com/dataset_challenge`

[5] Scikit-learn library documentation: `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

[6] NLTK library documentation: `http://www.nltk.org/`