Sreedevi Gangarapu
Tatiana Romanchishina

Predicting a Yelp Review Rating from its Text Alone

Abstract

Online user reviews are a central part of many web services where users can post their opinions about businesses, products and services. A review usually consists of free-form text and a numeric star rating, usually out of 5 or 10. The problem of predicting a user's star rating of a product, given the user's text review for that product, is called Review Rating Prediction and has lately become a well-known problem in machine learning. In our project, we use the dataset from the Yelp Dataset Challenge to build and evaluate several classifiers for this multi-class classification problem. We experiment with combining multiple models to improve the prediction accuracy. We evaluate the prediction accuracy of a model in terms of precision and recall for all classes.

Introduction

Yelp has been one of the most popular sites for users to rate and review local businesses. Their data are available through the Yelp Dataset Challenge, which consists of approximately 1.6 million reviews for 61 000 businesses. Businesses organize their own listings while users rate the business from one to five stars and write text reviews. Users can also vote whether reviews written by other users are helpful, funny or cool. Using this enormous amount of data, we would like to learn to predict the rating of the review based on the text of the review and some additional information, specifically the three sets of votes indicating how many people found this review funny, useful or cool.

Related Work

Ganu et al. (2009) focus on improving recommender systems using the text of the reviews. They add text analysis to a recommendation system and demonstrate that the more detailed textual information improves the prediction quality. They also show that the textual features of the review are more informative than the star rating.

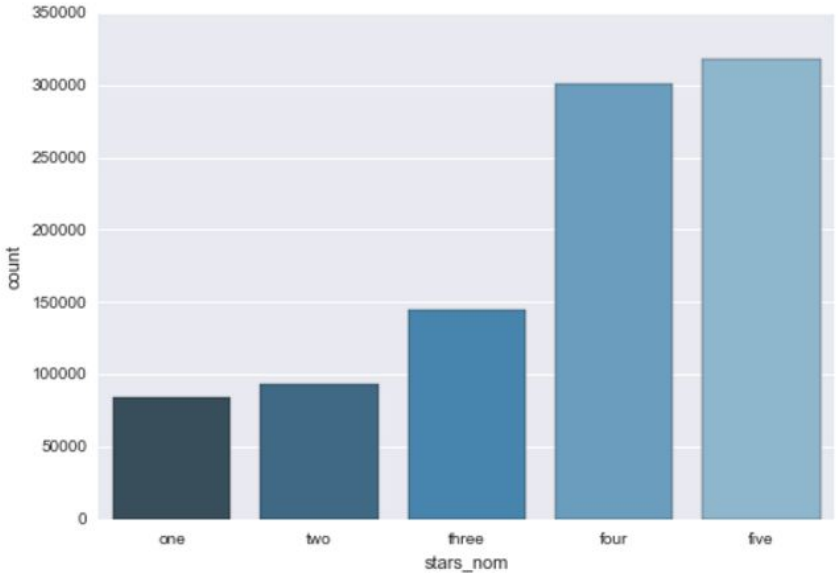
Most of the more recent work related to review rating prediction relies on sentiment analysis to extract features from the review text. Qu et al. (2010) tackle this problem for Amazon.com reviews, by proposing a novel feature extraction method called bag-of-opinions, which extracts opinions from the review corpus, computes their sentiment score, and predicts a review's rating by aggregating the scores of opinions present in that review and combining it with a domain-dependent unigrams model.

Li et al. (2011) proposes incorporating the information about the reviewer and the product into a text based learner to improve the review rating prediction accuracy. They perform a number of experiments to demonstrate the effectiveness of their model. This especially becomes important when considering new and unpopular products or inactive users.

Data set

The dataset we will use in this project is from the sixth round of the Yelp Dataset Challenge. It consists of over 1.6 million reviews by more 360 000 users for 61 000 businesses. The dataset includes a variety of business types, but in this project we decided to focus on restaurant reviews only. Due to the size of the dataset and our limited processing power, we had to work with a subset of all restaurant reviews.

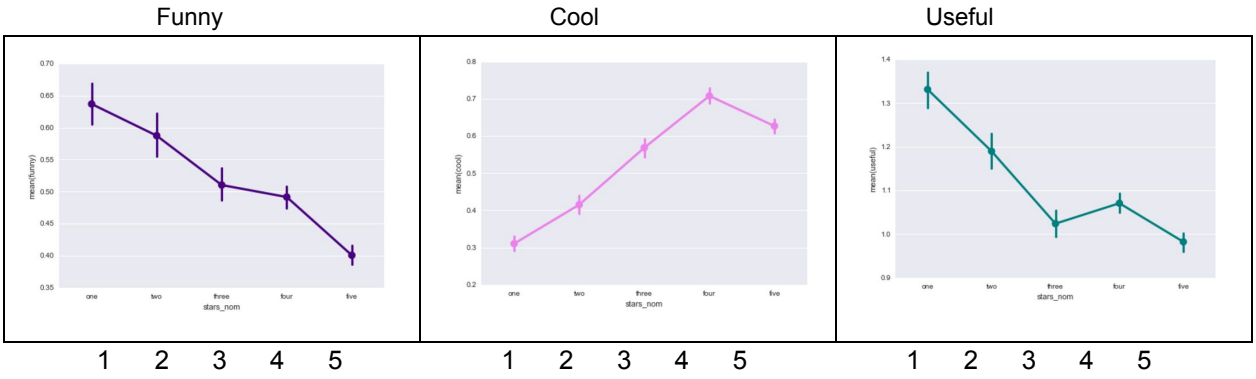
Figure 1: Distribution of restaurant reviews by the number of stars.



We decided to include the three sets of votes indicating how many people found the review funny, useful or cool. When we were exploring the data, we plotted the average count per star for each set of votes. Count data can be difficult to interpret, and as part of future work we want to scale these counts. Simple visual assessment suggests (Figure 2) that these votes have a correlation with the number of stars the review has.

Dataset is originally compiled JSON, which we have expanded to reduce and edit, and then convert to a matrix-based CSV format for classification and prediction.

Figure 2: Average counts of votes.



Project Goal and Outline

Our plan is first to extract features from the text of the reviews. We will use unigrams as a feature extraction method. When we have all the features ready, we plan to build classification models that will use the extracted features to predict the number of stars. We will use cross-validation and other methods to evaluate the accuracy of the models. Finally, we will choose the best model(s) for prediction.

We will use the Python library *nltk* - Natural Language ToolKit (Bird, 2009) - to perform some of the text preprocessing. We will also use this library to filter out the so-called “stopwords”. These stopwords are usually the most frequently used words, but they do not convey any significant information, for example, the articles “a” and “the”. The library *nltk* contains a list of stopwords for the English language.

We will extract the features using a unigrams model, also known as a bag of words. This bag of words is created from the top frequent words in all raw restaurant reviews. We plan to use the features with a frequency of at least 800. Again we will use *nltk* to perform these tasks.

When we have all the features, we will build models using different machine learning algorithms, that we learned from James et al. (2009) and Alpaydin (2014). We specifically want to try K Nearest Neighbor (KNN), Multinomial Logistic Regression (MLR), Naive Bayes (NB) and Support Vector Machines (SVM) with a linear kernel. We will attempt combining multiple learners to improve the results of their individual predictions.

Experimental Setup

Evaluation Metric

We use precision and recall as the evaluation metrics to measure our rating prediction performance. Precision is the number of retrieved and relevant instances divided by the total number of retrieved instances, or in other words, the proportion of the retrieved instances that

are relevant. Recall is the number of retrieved relevant instances divided by the total number of relevant instances, or basically the fraction of relevant instances that are retrieved. Precision and recall are calculated as follows (Alpaydin, 2011):

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \qquad Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Precision and recall seemed necessary because we found the traditional accuracy and error scores not very informative. Some classifiers will maximize their accuracy by predicting every star rating as 4 or 5, because those are the majority of the reviews. In that case the precision and recall for each star level will give a better picture of the model's performance. We also suspected that some of the levels might be classified with a higher accuracy than others, and we wanted to be able to see that.

Implementation Details

Data provided by Yelp is in json format. To convert it to csv, we used Python libraries *json* and *pandas*. We also used the library *numpy* to do some necessary data transformations. Formatted data is stored in the data structures (particularly dataframe) provided by *pandas*, which allows us to convert a dataframe to and from csv. We used the library *nltk* for text cleaning and feature extraction.

To build the KNN, MLR, NB and SVM models and validate them, we used Python *scikit-learn* machine learning library. Also built on top of *numpy*, it was easy to seamlessly incorporate library with our existing code.

We have also used R *core* libraries for some minor data transformations, the library *jsonlite* for parsing json files and libraries *tm* and *wordcloud* for plotting word clouds.

All our code and a copy of this write up can be found in this repository: https://github.com/TatianaRoma/Yelp_Review_Prediction .

Data Preprocessing

Before we can proceed with the analysis, we first need to prepare the data. The primary file we will work with - yelp_academic_dataset_review.json - is ~1.5GB. Even after removing some unnecessary attributes, the table remains very large. It is possible to work with, but most of the processes take a long time. We use a much smaller file yelp_academic_dataset_business.json to select all restaurant reviews.

Yelp allows users to write text reviews in free form. This means that a user may excessively use capital letters and punctuation marks to express their intense emotion and also slang words within a review. Moreover, stop words, like 'the', 'that', 'is' etc, occur frequently across reviews and are not very useful. Therefore it is necessary to preprocess the reviews in order to extract meaningful content from them.

To do this we used the *nltk* library and preprocessed the text by tokenizing, removing stop words, punctuations and digits, and converted all letters to lowercase to reduce redundancy in subsequent feature selection. We create a unigram features vector from the preprocessed data.

Feature Selection and Extraction

We analyzed the raw data from all reviews to choose the words with the frequency greater than 800. We do this by first creating a dictionary of all the words occurring in the review corpus. Then we extract a sparse training matrix, where the rows are reviews and the columns are tokens. The cell of this matrix represents the number of occurrences of the corresponding token in the corresponding review. This array of words became our unigram feature vector.

Machine Learning Techniques

To train our prediction models, we choose four learning algorithms: (i) KNN; (ii) Logistic Regression; (iii) Naive Bayes and (iv) SVM with a linear kernel.

We use 10 fold cross validation and run our algorithms on a random sample with the size of 50000. We randomly split this sample set into training (70% of the data) and test (the remaining 30%) sets.

KNN

K Nearest Neighbors is a non-parametric method used for both classification and regression. In our case we use it for classification. This means, that given a positive integer K and a test observation, the KNN classifier will identify the K points in the training data that are closest to the observation. It will then estimate the conditional probability for each class as the fraction of the identified K points whose response values equal to that class. Finally, the KNN algorithm will apply Bayes rule and classify the test observation to the class with the largest probability (James, 2009).

Logistic Regression

Logistic Regression models the probability that an observation belongs to a particular class using a linear combination of the observed features and some problem-specific parameters(James, 2009). In our case, we will model the probability a review has a certain star rating, for example 5, given the features we extract from the text of the review, as follows:

$$Probability(star\ rating = 5 \mid features)$$

Naive Bayes

Naive Bayes classifier is called naive, because it ignores possible dependencies, or correlations, among the inputs and reduces a multivariate problem to a group of univariate problems (Alpaydin, 2011). It is a very simple classifier that assigns each observation to the most likely class, given its predictor values (James, 2009). We will assign a certain star rating, for example 5, to a review that has the highest conditional probability that it is 5 star review.

SVM (with a linear kernel)

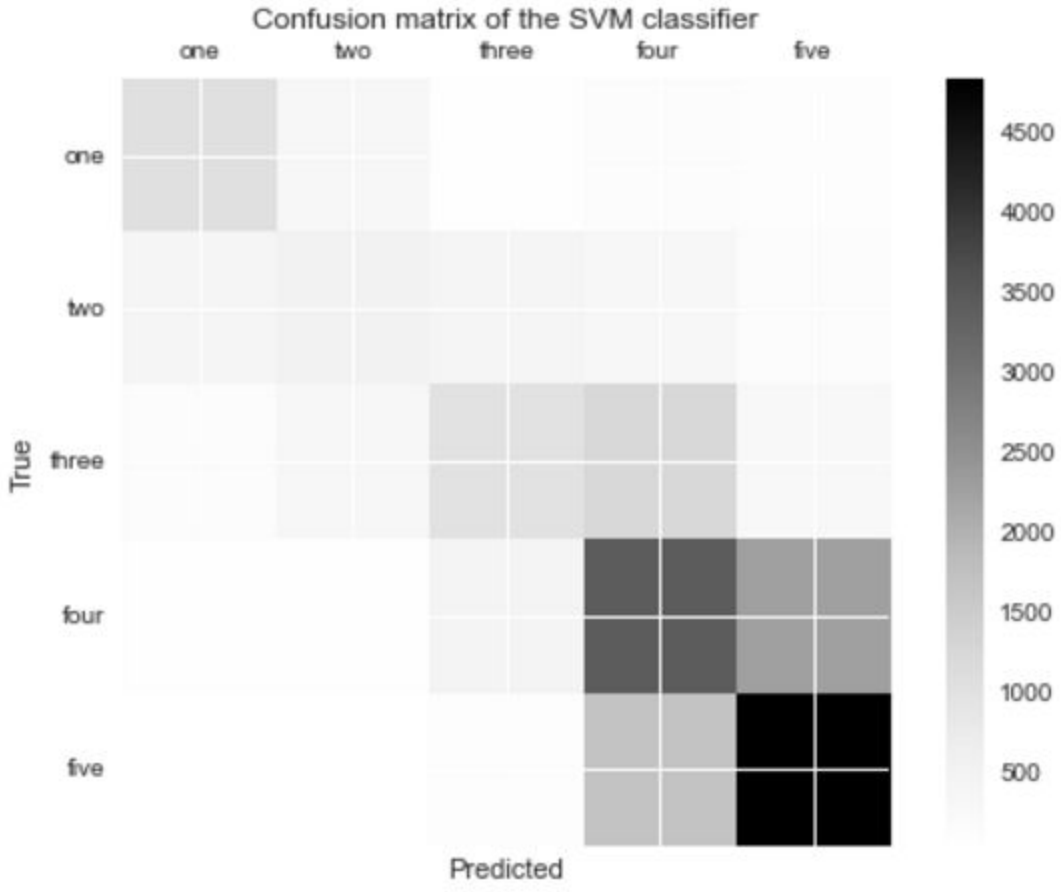
Support Vector Machines (James, 2009) are learning models that create a representation of the data as points in space, where each point is an observation, which is a

review in our case. The algorithm maps them in such a way that the points are grouped into categories corresponding to classes and are separated by a gap. New observations are classified based on where they are mapped in this space.

Results

We evaluated our models in terms of precision and recall because we expected the algorithms to have a higher accuracy when predicting some star ratings than when predicting some other. Figure 3 shows the confusion matrix for the results of our best SVM model. It is clear that “5” star reviews are classified with a higher accuracy than others, but it is also the majority of reviews. “1” star reviews are also classified with a high accuracy, though their number is much smaller. “4” star reviews are classified with a lower accuracy, and it looks like they get confused with “5” star reviews a lot. “2” and “3” star reviews are classified as almost any star rating.

Figure 3: SVM confusion matrix.



This finding that some star ratings are more distinguishable than others can be confirmed by simple visual analysis of the most frequent meaningful words associated with each class. To do that, we plotted word clouds for each star rating, using the R libraries *tm* and *wordcloud*. Figure 4 shows an example of the wordclouds created from this dataset. If we compare “1” and “5” star reviews, they are very different from each other. But “4” and “5” are very close, and so are “1” and “2”. “3” star reviews are odd, because they can be positive or negative, so they are similar to “2” and “4” star ratings.

Figure 4: Wordclouds for each star rating.

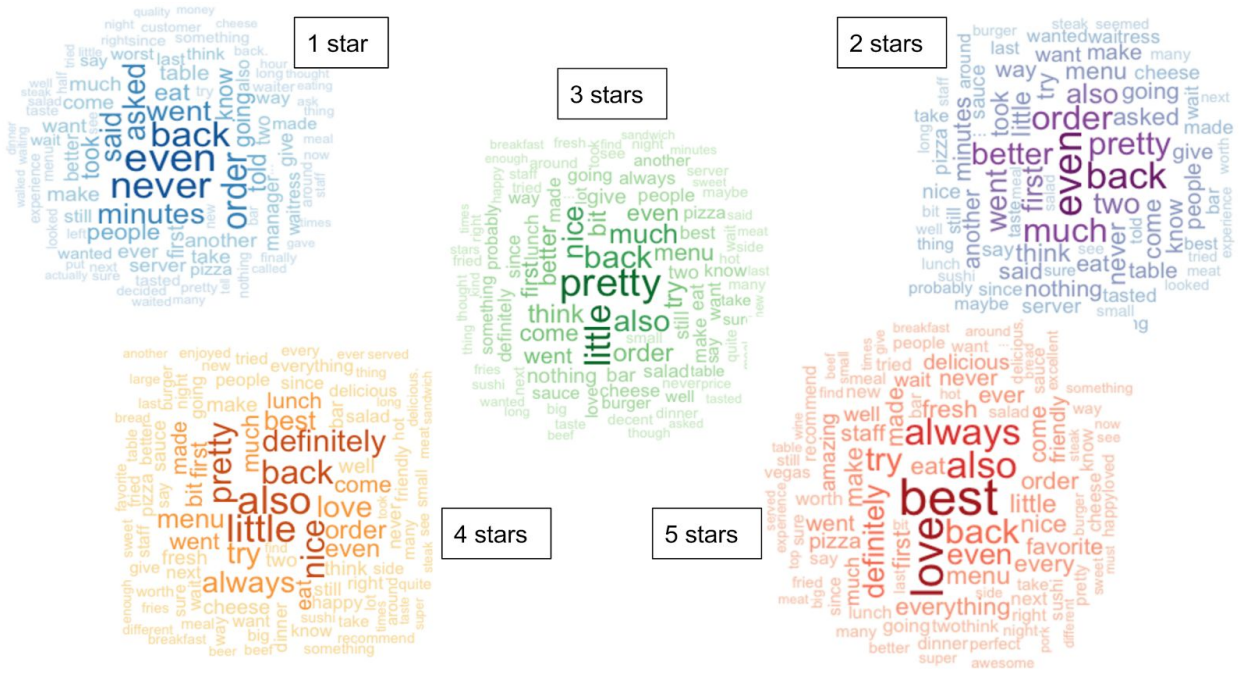


Table 1 summarizes the precision and recall for the best individual models we built. Logistic Regression, Naive Bayes and SVM with a linear kernel performed well in general, achieving overall accuracy rate of ~56-56%, but there were slight differences in how they performed for different star ratings. Our next step is to combine these learners to improve the prediction accuracy.

Table 1: Individual Model Results

Star Rating	Algorithms							
	KNN		LR		NB		SVM(lin)	
	precision	recall	precision	recall	precision	recall	precision	recall
★	0.34	0.28	0.61	0.60	0.54	0.63	0.58	0.60
★★	0.22	0.04	0.42	0.25	0.38	0.28	0.39	0.29
★★★	0.29	0.08	0.48	0.31	0.45	0.38	0.47	0.32
★★★★	0.37	0.33	0.49	0.53	0.51	0.48	0.50	0.54
★★★★★	0.41	0.71	0.61	0.74	0.63	0.73	0.63	0.71

We experimented with combining the predictions of three learners (Logistic Regression, Naive Bayes and Support Vector Machines) to improve the prediction accuracy. Each of our learners outputs a class label, which is the predicted star rating of this review. Our first attempt was to look at the three predicted labels and pick the one with the highest frequency. If there was a conflict, it would either pick the prediction produced by SVM, since we were getting the highest accuracy from it, or it would pick “2” as the class label, since two-star reviews are almost equally likely to be classified as “2” as any other star rating. However, neither of these combined models improved the prediction accuracy.

Next we tried computing the average of the three predictions, and rounding it to the nearest integer to produce the final prediction. This strategy showed a significant improvement. Figure 5 shows a comparison of the confusion matrix produced by SVM and by the combined model. The shading along the diagonal became darker, indicating that the overall prediction accuracy has improved.

The overall prediction accuracy has indeed improved to 60%, but that was not the only exciting discovery. Figure 6 shows the precision (left) and recall (right) for our three best individual models and two blended models for every star rating from “1” to “5”. The rightmost light-blue bar for each star rating represents the blended model that used the average of three predictions. As highlighted by red squares in Figure 6, one can see that precision has greatly improved for “1” and “5” star ratings, while recall has significantly improved for “2” and “4” star ratings.

Figure 5: SVM Vs. Blending Individual Models

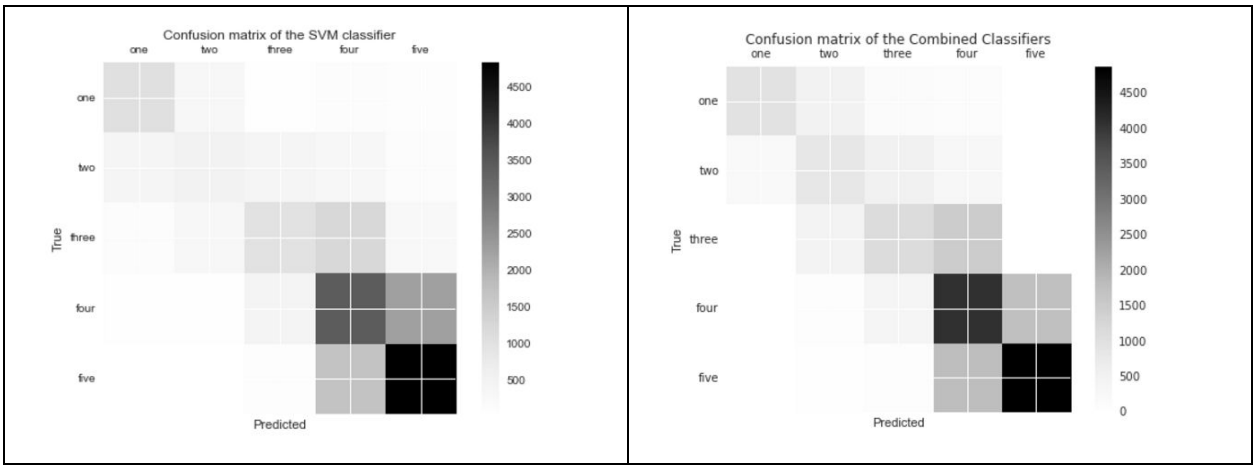
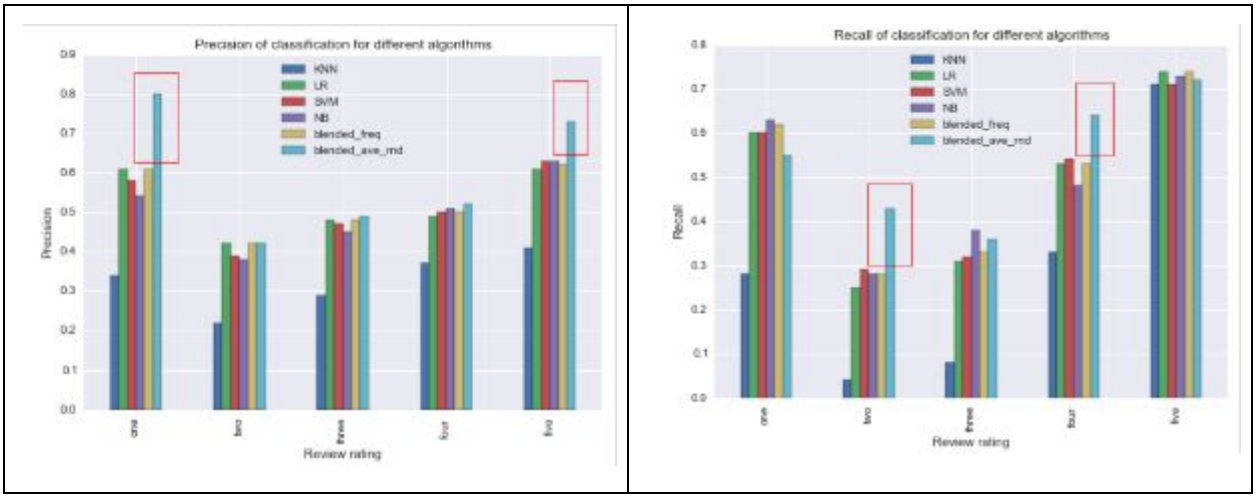


Figure 6: Precision and Recall for different algorithms (red squares indicate improvement of the blended model).



Problems/Issues

Training most learners took along time. We did not have enough memory and processing, due to the scale of this project. Even though we considered just 5% (50 000 reviews) of the original data few algorithms, like SVM, with 10 fold cross validation took 2 days to run.

Conclusions and Discussion

In this paper, we work with the Review Rating Prediction problem for restaurant reviews on Yelp. We treat it as a 5-class classification problem, and examine unigrams feature extraction and both supervised and unsupervised learning methods to construct prediction systems. Performance evaluation is done through 10-fold cross validation. We use precision and recall as main evaluation metrics.

We analyzed the results of individual learners and observed that the accuracy of prediction is higher for “1” and “5” star ratings and very low for “2” and “3”. We then combined three learners (Logistic Regression, Naive Bayes and SVM) and were able to improve the precision for “1” and “5” and recall for “2” and “4” star ratings. We think it is because “2” and “4” star reviews were often misclassified as “1” and “5” star reviews, respectively. Our blended model improves the prediction accuracy by averaging the individual predictions, “shrinking” some misclassified labels back to the true value.

Future Work

There are many avenues for improvements and future work. We list some below:

1. We did the analysis only on 1/20th of the data (50 000 reviews) because using all 1 million reviews turned out to be very expensive in terms of time and memory. We are sure that if we did the analysis on entire data, accuracy would have been at least 10% more than we have. To deal with the time and RAM bottleneck, we could try parallel processing over clusters, using Hadoop, Spark.
2. We experimented with limited feature extraction techniques and algorithms. We can try more elaborate experimentation for feature extraction like bigrams, trigrams, POS (Parts-of-Speech) tagging, etc.
3. We could consider this problem as a two-layer classification problem. We could first build a model to separate, for example, “1” and “2” star reviews from the others, creating a positive and a negative group. Then we would try to further separate the star ratings within those groups.

References

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Bird, S. et al. (2009). *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O’Reilly Media. http://www.nltk.org/book/

Ganu, G. et al. (2009) *Beyond the Stars: Improving Rating Predictions using Review Text Content.* Twelfth International Workshop on the Web and Databases (WebDB 2009), Providence, Rhode Island, USA

James, G. et al. (2009) *Introduction to Statistical Learning Using R.* Springer Texts in Statistics. http://www-bcf.usc.edu/~gareth/ISL/index.html

Li, F. et al. (2011) *Incorporating Reviewer and Product Information for Review Rating Prediction*. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, pages 1820-1825.

Yelp Public data: http : //www.yelp.com/dataset challenge