Yelp Project Report

Joseph Simonian
UC Berkeley

Yiyuan Li
UC Berkeley

Atul Lanka
UC Berkeley

Nikhil Sharma
UC Berkeley

May 2017

1 Introduction

Yelp thrives on the numerous reviews that are left by users for local businesses and restaurants. It is of considerable value to analyze this data and find out whether they help in directing the performance of restaurant or whether restaurant performance is indeed dictated by other factors. The purpose of the project is to extract features from the data to predict the star rating given by the customer as well as other information of the restaurant and predict the average star rating for each one. The analysis indeed serves as an experiment to see whether the performance of restaurant is predictable as a function of its information and customers' feedback. The results from this analysis may provide a solution for further processing of huge volumes of data beyond the bounds of purely human analysis.

2 Exploratory Data Analysis

The Yelp business data includes the the position (state, city, neighborhood, longitude, latitude, zip code, etc.), working time, number of reviews, average star rating, and category of cuisine offered for a variety of restaurants. The 2500 restaurants in the dataset are clustered in 6 states across United States.

All the restaurants are located in 10 cities in these 6 states (though there are only a total of 6 metropolitan areas) of 4 categories: Mexican, Coffee&Tea, Chinese and Pizza restaurants.

The corresponding review data maintains the star rating on a scale of 1-5, review text of each user for each reviewed restaurant, time of the review, and whether or not the review was rated as cool, funny, and/or useful by other Yelp users.

There are 116474 reviews in total from 2510 restaurants. Out of this large corpus of reviews, roughly 35% of the reviewers gave a rating of 5 stars, 27% gave 4 stars, and 13.5% gave 3 stars. Of the 2510 restaurants, approximately 37% are based in Arizona, 30% are based in Nevada, 12% from Philadelphia, 10% from North Carolina, 6% from Ohio and 5% from Wisconsin.

These two datasets are what we utilized for the bulk of our analysis, presented in this report.

3 Model Pipeline

We ended up performing a two-stage pipeline for our final model. First, we processed data for each review, using the text data of the review as well as the "Useful", "Funny", and "Cool" tags as features. We then tried a variety of models, attempting to predict the rating (1-5) for each review. We then aggregated these values for each restaurant in the test data set, and computed the average star rating for each restaurant. This composes the "first stage" of our pipeline.

In the second stage of our pipeline, we added in additional features for each restaurant, such as its price range and geographic location. We then fit a second model on that data plus the average star rating data from the previous stage, in order to produce a final prediction for each restaurant. In this case, we treated the problem as one of regression as opposed to classification, both because the variable we were attempting to predict now took non-integer values such as 3.5 or 4.5, and because we were allowed to submit predictions such as 4.25. In the case where we are unsure of whether a rating is 4.0 or 4.5, submitting 4.25 carries a lower risk than than submitting 4.0 or 4.5 and having a half chance of being wrong.

4 Text Mining of Review Data

4.1 Natural Language Processing Model

We considered a variety of preprocessing models for the review text. Starting from simple bag-of-words, we tried various adjustments. These included:

1. Tf-idf weighting. This performed poorly, achieving a far higher RMSE for most models than simple bag-of-words. We believe poor performance is partly due to overfitting, by relatively increasing the weights of terms that occurred infrequently.

2. Bag-of-words on the top N most frequent words. This gave us somewhat better results for all models, though the precise choice of N varied between models.

3. Bag-of-words on the top N most frequent words, followed by singular value decomposition and principal components analysis. This was in part to reduce the time necessary to train classifiers, and ended up giving somewhat better performance on Random Forests. Its effect on other classifiers was minimal.

4. Bag-of-words with bigrams and trigrams added. This was done to add context to words, on the intuition that phrases such as not bad should be distinguished from the components words, since the meaning is quite different. Both trigrams and bigrams ended up resulting in a performance reduction, even after applying other methods such as dropping infrequent words. Thus, we ultimately rejected both.

5. Doc2Vec - described here. We tried several vector sizes, from $p = 100$ to $p = 1000$. Ultimately, the results were no better than those on various bag-of-words models.

In all cases, we used a standard set of stop words, provided in Python's "sklearn" library. In the end, we used two preprocessing techniques for different review-prediction models.

1. For our baseline Gaussian Naive Bayes model, and for Random Forests and Logistic Regression, we used bag-of-words with 10000 words, and then computed the top 100 principal components. With this model, all reviews were converted to lowercase and stripped of punctuation before bag-of-words.

2. For SVMs, we used bag-of-words with 700 words, with several modifications. We duplicated nouns that were preceded by adjectives, and in cases where a sentence was ended by punctuation other than a period, we duplicated the last adjective in the sentence. We found that this strategy increased the importance of more "intense" adjectives, such as "love" or "worse", in comparison to adjectives such as "great" or "bad". We refer to this method as the "Revised BoW" method.

4.2 Feature Selection

In all cases, we add the features "Useful", "Funny", and "Cool". In Yelp, other users may label a review using these indicators - we simply take the number of times a review received each label as a new feature. All three features are quite predictive, as seen below in Fig.1, our analysis of the top ten features for both the bag-of-words/SVM model, and the revised bag-of-words model.

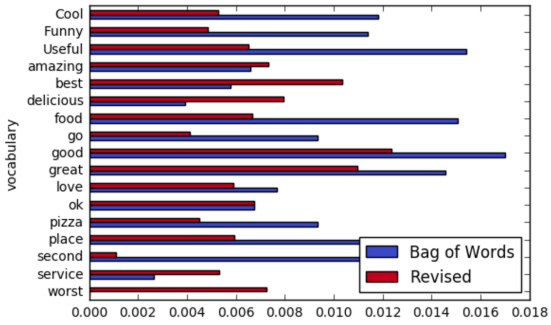

Fig.1 Feature importances

We can see that the most predictive features include both positive words like "good", "best", and negative feedback such as "worst" and "bad", as well as the Cool, Funny, and Useful ratings received by individual review.

From Figure 2 we can see the average feedback other users give to the reviews by of different rating. One way to interpret is that customers with 5 stars or 1 in rating may be considered biased because the rating is too extrem or customers are not enough sensitive and is less likely to give objective judgement than 3-4 rating comments, which takes the majority of the reviews.

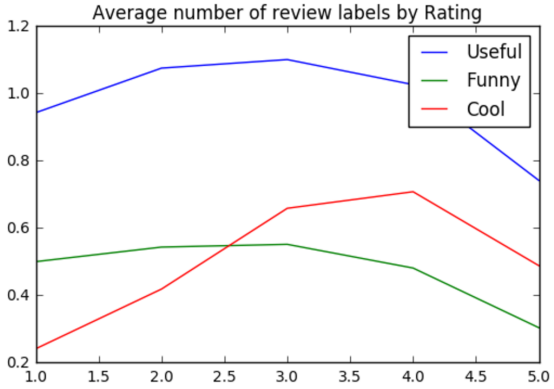

Fig.2 Review labels

Our selection of 100 principal components came from analyzing the explained variance ratios of each of the principal components of the training data. A scree plot presenting this analysis is visualized below in Figure 3:

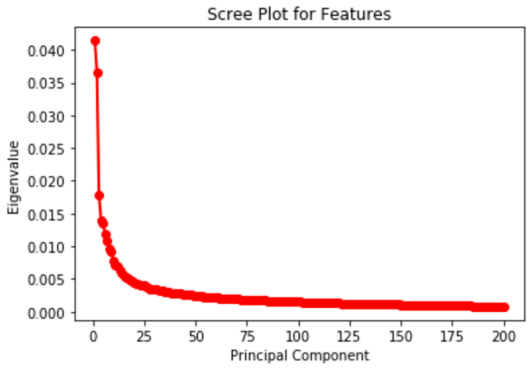

Fig.3 Scree plot of featurized data

Based upon this scree plot, we tried values from 25 to 200, and eventually found that we achieved best results with 100 principal components.

5 Time Series Analysis

We also considered using the date on which a review was posted as a feature for this stage of our pipeline. To check if this was informative, we calculated the mean stars received per day for all restaurants to see whether customers have specific time-based trend for ratings. Ruling out the days that receive less than 20 reviews to reduce variance, we get a result without any obvious trend or seasonality to show that there is particular pattern people rate the restaurant in the specific time period, and centered between 3 and 4, which indicates the average rating of the restaurants.

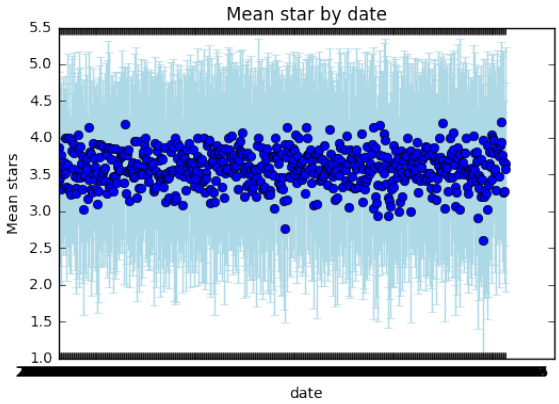

Fig.4 Mean stars over time

6 Classification Models

After obtaining our data, we utilized a variety of classification methods to predict review star ratings. We initially analyzed each model by comparing the average of its predictions for each restaurant to the true star rating of that restaurant, without the second stage of our pipeline (that is, ignoring restaurant-specific features). Our baseline model, a Naive Bayes classifier, received an RMSE of 0.8617.

We were able to yield significantly better results with other methods and some hyperparameter tuning; specifically, we tried SVMs, random forests, and logistic regression. For SVMs, we experimented with both one-vs-one and one-vs-rest classifiers along with tuning regularization parameters and using both linear and radial basis function kernels. The resulting RMSE from

these steps yielded finals RMSE that hovered in the range of 0.6193 to 0.7027. A definite improvement, but we felt we could do better still.

After training a variety of random forests, the one that yielded the lowest RMSE was one fit with 20 weak learners (decision trees), with a a maximum tree depth of 20 to prevent over-fitting. Our choice of 20 weak learners arose primarily from the increasingly high overhead in terms of training time, with negligible additional performance gains. The final RMSE for this method turned out to be 0.4907, slightly outperforming all of the SVMs we trained.

Our best and final model was trained with logistic regression. After trying various types of penalties and regularization parameters, we got the best performance with l_1 regularization and $C = 0.2$, an RMSE of 0.4080.

The results of all these methods (hyperparameters, RMSE) are summarized compactly in the table below.

7 Geographical Analysis on Location Data

As part of the second stage of our pipeline, we wished to take note of each business's geographic location, on the grounds that different neighborhoods might have different average ratings. Since the various urban areas under consideration are located quite far apart from one another, we performed analysis on each one separately. For each state (not each city, since some cities were adjacent to one another), we took the latitude and longitude for each business and applied k-Means clustering with a Euclidian distance metric. In order to obtain the ideal k value for each state, we used a rule of thumb, $k \sim \sqrt{n/2}$, to determine an initial value of k. We use this instead of the Elbow Method since it did not work well in our particular case - our k values from that method were unreasonably small. From there, we checked with values close to the initial k value to determine the optimal number of clusters via visual inspection. To see the full clustering of each city we analyzed on a map of the United States, you can visit here and see a visualization we generated using the Google Maps API. An example of these clusters for a single city is below in Figure 5.

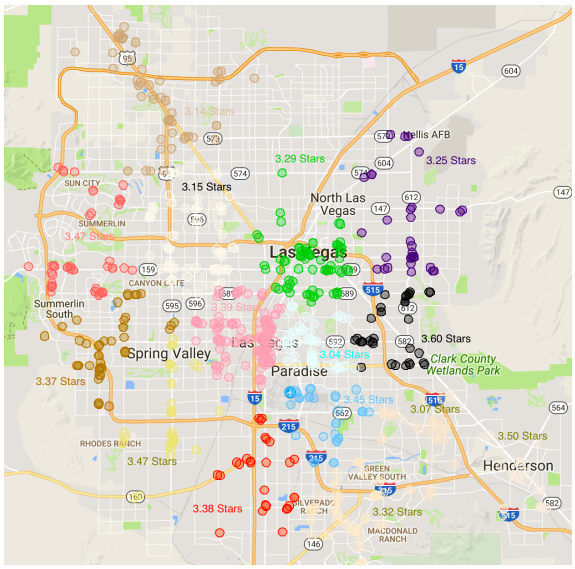

Fig. 5 K-means clustered of restaurants in Las Vegas, NV (with mean star rating)

We can see from the picture that restaurant from the main boulevard of Las Vegas (e.g. South Las Vegas Blvd), is expected to receive a higher average star rating than those away from downtown. Noting that famous hotels like Bellagio are located there and the service of the nearby restaurants necessitates a very high quality in order to survive and earn recognition from its customers.

8 Extra Specific Business Features

In the second stage of our pipeline, we used several features from the business CSVs. In addition to a one-hot encoding based upon the clusters generated with k-Means, we considered restaurant price range (1-4), and a variety of binary features - namely, whether the restaurant accepted credit cards, accepted reservations, and whether it offered delivery or takeout. We also performed one-hot encoding on the four restaurant categories present in the data - Chinese, Mexican, Pizza, and Coffee & Tea, to generate features corresponding to those categories. Finally, we included the average predicted star ratings computed by stage one of our pipeline.

We considered a variety of models to fit this data, ultimately settled on a simple penalized

Method	RMSE: Reviews Only	RMSE: Full Pipeline	Hyperparameters
Gaussian NB	0.8617	0.7308	None
SVC: One-vs-One	0.6989	0.5413	$c = 1$, kernel=linear
SVC: One-vs-Rest	0.6193	0.5216	$c = 1$, kernel=linear
SVC: One-vs-One	0.7027	0.5341	$c = 10$, kernel=RBF
Random Forest	0.4907	0.4781	N_Trees=20, max_depth=20
Logistic Regression	0.4080	0.3858	penalty=l_1, $C = 0.2$

regression, primarily because the dimensionality and number of data points were both low. With lasso regression, cross-validated for optimal parameter λ, the only feature with coefficient set to 0, besides several of the centroid features, was the binary feature representing whether a restaurant offered takeout (which 88% of the restaurants offered). The question here is, how much did this change our initial star predictions, calculated based upon ratings? In our final stage-2 model, the stage-1 predicted star rating had a coefficient of 0.88, with several other features having coefficients in the range of 0.02 to 0.1. Thus, the model performed an appreciable amount of tuning on the stage-1 data, adjusting it based on restaurant-specific features not present in the reviews.

9 Conclusion

After analyzing the data provided by the Yelp dataset, we can see that customers are more likely to give positive feedback when they respond with a comment about the restaurants, the average star is between 3 and 4, and the most important words used in the reviews are typically positive. However, negative feedback is equally important for prediction, especially under positive-dominant circumstances, where we use a revised bag-of-words method to mine them out. The model points out that adjective like "good", "amazing", of the reviews and useful, cool, funny from other users is really predictive of the restaurant stars. The time series analysis shows that the rating routine of the customers doesn't change much by time, which makes their judgement quite reliable. And the K-means clustering on the location of the restaurant indicates that geographic information has influence on the customer feedback about the restaurant according to commercial or social reasons.

Our final model consisted of a two-stage pipeline. First, a logistic regression was run on features from individual reviews, to predict the star rating given in each review. We then aggregated these predicted star ratings by restaurant and averaged them. In the second stage, we fit a linear regression with l1-penalty on these aggregated ratings, in addition to a number of features specific to each restaurant, in order to get a final prediction for the star rating of each restaurant.

10 Acknowledgements

Yelp, in generosity, provides all the data we used in the report. Google provided their Google Maps API which we heavily utilized for visualization generation in our project.

11 Github

To reference our source code, please visit our Github repository, which can be accessed by clicking here.

Our Kaggle RMSE for our final model (Logistic Regression as stage 1 of the pipeline) was 0.36342, in line with the RMSE calculated on validation data. Our team name was "Four in MLK", and our respective Kaggle usernames are: jsimonian (Joseph), yiyuanli (Yiyuan), atullanka (Atul), bobflanagan (Nikhil).

12 References

- Yelp Dataset Challenge
- Prediction of Yelp Review Star Rating.
- Distributed Representations of Sentences and Documents