Predicting Yelp Ratings From Business and User Characteristics

Jeff Han
Justin Kuang
Derek Lim
Stanford University
jeffhan@stanford.edu kuangj@stanford.edu limderek@stanford.edu

I. Abstract

With online evaluation systems, people have a new way of making an informed decision. Ebay, Amazon, Stack Overflow, and Yelp are all examples of online systems where users submit their evaluation of a particular item whether it be another user, a product, etc [1]. These networks allow a user to submit their opinion to be read and evaluated by other users in the network. These crowd-sourced reviews act as a method for users to infer evaluations like whether a restaurant is worth going to, if a product is good quality or whether to trust an online seller. These systems form the backbone behind the trust in online transactions [2].

We will look at the Yelp network specifically. Past work has included ways to predict the star rating by doing sentiment analysis on review text [3]. We will approach this task from a new angle by predicting solely based on features of the user and business, without information from the review text. In this scenario, then simply given a user and a business, we can predict how the user will rate the business. We can extend this as a way to recommended businesses to a given user based on the prediction of how much they will like the business. The work in this paper can be extended to several applications beyond Yelp. In general, it shows how a user's evaluation depends on the surrounding factors and context it is within.

II. Introduction

Online evaluation systems have given people a new factor to consider when making everyday decisions. For example, before buying a new TV or furniture set, people can now first check its reviews on Amazon. When deciding where to eat pizza tonight, hungry customers can look up restaurant reviews on Yelp.

We will explore the underlying user and business properties of the Yelp network and how that can be analyzed to draw conclusions about a given review. In particular we will explore what information can be extracted from user and business features and how to predict star ratings based on those features.

III. Dataset Description

In this paper, the data we use comes from the Yelp Dataset Challenge sponsored by Yelp. It consists of 42,153 businesses, 320,002 business attributes, 31,617 check-in sets, 252,898 users, 403,210 tips, and 1,125,458 reviews. As shown in the graph, the dataset is comprised of the following: 110772 1 star reviews, 102737 2 star reviews, 163761 3 star reviews, 342143 4 star reviews, 406043 5 star reviews. From this we can see that the dataset is skewed towards positive higher star reviews. Users rate businesses using integer star ratings (1, 2, 3, 4, 5), average star ratings for businesses are defined at a half star granularity (1.0, 1.5, 2.0, 2.5....5), and average star ratings of users are defined up to two decimal points (3.41). In order to reduce the reviews to a manageable size we only take into account users who have made 50 to 100 reviews, which amounted to a sample size of over 8,000 users. The review, business, and user data were centralized in a Mongo database which we used as the main datastore for this implementation.

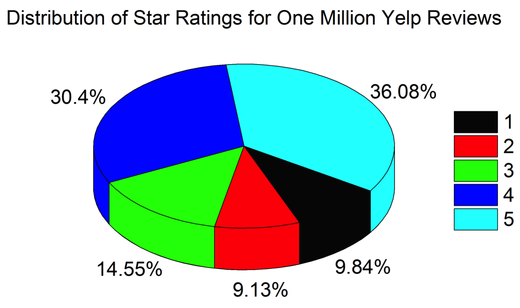

Figure 1: Star Rating Distribution

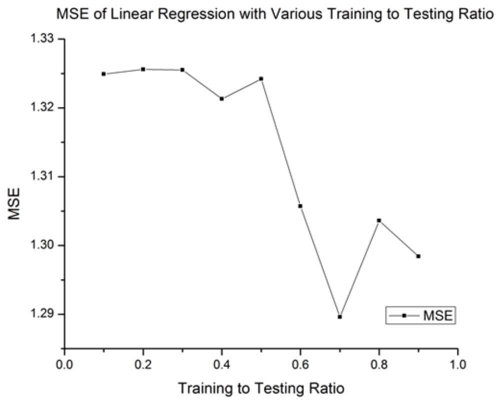

Figure 2: Results of Linear Regression for Different Training/Testing Splits

IV. Linear Regression

Baselines. We begin by defining two baselines to be used as a metric to compare against. Baseline 1, **B1**, will predict the star rating based solely on the average star rating given by user. That is, a prediction for each user is made based on the average of the user's previous rating. Baseline 2, **B2**, will predict the star rating based solely on the average star rating of all reviews. That is, it takes the average of all tested reviews and uses that average as the prediction for the remaining reviews. Running **B1** yields prediction accuracy of 42 percent. **B2** yields prediction accuracy of 39 percent.

Linear Regression on Reviews. This method combines features from both the users and from the businesses. For each review we pull features both from the user who made the review and from the business the review is for. The data set is split into training and test portions in increments of 0.1, that is (training, testing) set to (0.1, 0.9), (0.2, 0.8) and so on, for a total of nine splits. We also go through every possible permutation of the features, such that we either include or omit the feature for a total 2^n possible permutations, where n is the total number of features. Thus for for this linear regression model, a total of $(2^n) * (9)$ runs are done. When testing each review, we produce a predicted rating and the MSE when compared to the actual rating. The average of the MSEs for each training/testing split are shown below in Figure 2.

The total possible features used for this run are [user review count, user average stars, user friends, user fans, business review count, business stars]. Running through every permutation of features shows that the features with the highest weights are the user average stars, business average stars, and business review count. Thus these features have the highest impact on the overall score, which agrees with intuition. All of the other features have relatively low weight and thus very little impact on the prediction. Compared to the baseline, this method achieves much greater performance.

V. Collaborative Filtering

Basic Procedure. Given a user and a restaurant, first we find a group of similar users to the given user who have also been to the restaurant. We will define this group as the similarity group. Similarity between users is calculated from the users' average ratings given to different categories of restaurants, and the user' overall average star ratings. From this similarity group, we use a combination of the ratings similar users have assigned to the restaurant, and make a prediction of what the given user will rate the given restaurant. We find the similarity group by looking at all the users who have gone to that given restaurant, and rank them based on similarity as defined later. From this similarity ranking we base our prediction on the top N most similar users as defined by our model. An illustration of this procedure is shown below.

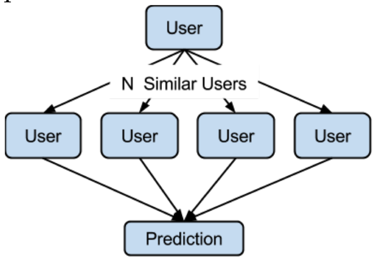

Figure 3: Making a Prediction from N similar Users

Utility Matrix. We define user features as the average rating given to different categories of restaurants. Example: American: 2, Chinese: 4, Italian: 4, Thai: 1. We define a utility matrix to capture these user features. We have two groups: users and user features. The rows of the matrix correspond to the users and the columns correspond to the user features. The value of a particular cell in row i and column j represents the rating which user i gave category j. A sample of this utility matrix is shown in Figure 4 below.

	Thai	American	Chinese	Italian
User 1	4		5	
User 2	2		2	4
User 3		4	4	

Figure 4: Utility Matrix

Similarity Measure.We explore three different methods of measuring similarity: Jaccard Similarity, Cosine Similarity, Pearson Correlation Coefficient. The flaw with Jaccard Similarity is that it ignores the value of the rating and only looks at the set of features with ratings. Pearson Correlation Coefficient is similar to Cosine Similarity but has a downside in that it slightly more computationally expensive. For these reasons, we settle on using Cosine Similarity, shown below in Figure 5.

$$\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \cos(\boldsymbol{r_x}, \boldsymbol{r_y}) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$$

Figure 5: Cosine Similarity

Using the Cosine Similarity measure by itself for our purposes presents an issue: all features are counted as positively correlated. That is, by simply having a feature present, it will have a positive impact on the prediction, regardless of how positive or negative the rating is for that feature. To explain this issue with an example, if user 1 rates Thai restaurants 4, meaning he likes Thai food, and user 2 rates Thai restaurants 1, meaning he doesn't like Thai food, this will result in a postive similarity simply because they have both been to Thai restaurants. This is erroneous for our application of an accurate prediction. To address this, we normalize the utility matrix by subtracting the row mean from each value. This makes it so that a user's lower rated features will use a negative value to be used in computing similarity.

Rating Prediction Methods. Now to arrive at a prediction we analyze two different methods, namely **unweighted average** and **weighted average**. For an unweighted average, we take the N most similar users, and simply use the average of those user's ratings of the given restaurant.
For a weighted average, we weight each of the top N most similar users by their similarity value. We compute the weighted average as defined in Figure 6. In the weighted average figure, let N represent the set of N most similar users, s_{xy} represent the similarity between user x and y, and r_{xi} and r_{yi} represent user x's and user y's rating of category i.

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

Figure 6: Weighted Similarity Average

VI. Results

We ran the algorithm defined by our model through every review in the entire reduced dataset. For each run, we produced both weighted and unweighted predicted ratings and calculated the MSEs from the actual rating.

All the MSEs for the run are then averaged as a measure of performance for the run. We repeated this using a different similarity group size N for all group sizes up to N=20. We also ran it for a group size N=all that matches the size of all users who have gone to the restaurant in question. The results are shown in Figure 7 below.

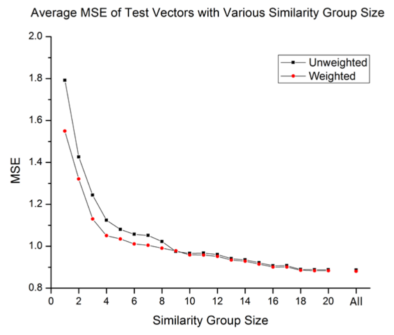

Figure 7: Average MSE with Varying N
The run that used a group size N=all turned out to produce the lowest average MSEs for both weighted and unweighted methods. A comparison of the predicted ratings to actual ratings, along with a trendline, are shown in Figure 8 below. For readability and to avoid cluttering of points, a subset of a hundred points are used in the Figure 8 .

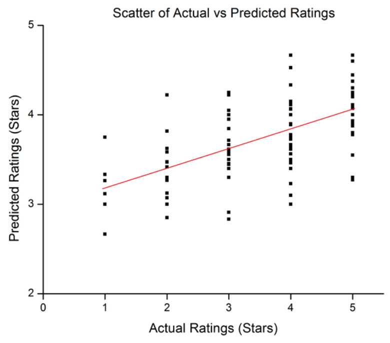

Figure 8: Results of the Lowest-error Run

VII. Discussion

From the results, we see that the weighted similarity prediction exhibited a lower MSE compared to the unweighted similarity prediction for a similarity group smaller than ten. this is not surprising as an unweighted prediction give each member of the similarity group equal influence on the overall prediction, which may be detrimental if the discrepancy between the most and least similar group members is large. However, after the similarity group exceeds ten members, the difference between the performance of the two measures converge. We believe this is due to dilution in the similarity group weighting system. As more members are allowed into the group, the total sum of similarity measurements (which form the denominator of the weights) grows larger and thereby reduces the influence of the most similar members from the overall prediction.

Another trend that exists for both weighted and unweighted similarity prediction methods is that the MSE monotonically decreases when increasing the similarity group size. This implies that the more reviews that we consider when forming the overall prediction, even from users that may be not similar to the test user, the better the prediction. In the "All" case, when all users who have rated a particular restaurant are in the similarity group, we see that the lower limit of the MSE is a bit below 0.9 for both prediction methods. This trend suggests a tradeoff between computation complexity and prediction accuracy since a larger similarity group leads to more computation. However, this is a reasonable tradeoff that is present in many recommendation systems.

Lastly, we recognize that the current system does not predict low star ratings, those of one or two stars, very well. The reason for this is twofold. First, the distribution of aggregate star ratings is non-uniform as three, four, and five star ratings account for over 80% of the total ratings in our dataset. Second, the star predictions are calculated by summing parts of the actual ratings from members in a test users similarity group and therefore, any high rating of the restaurant for a review in that group will skew the overall prediction in that direction. However, even though this presents a problem for reducing MSE, it may not effect the overall results of a recommendation engine that is created on top of this star prediction scheme. Even though predictions of one and two stars may be higher than their actual values, it is unlikely that they will be higher than the prediction of a three, four, or five star review. Thus, a recommendation engine that takes many restaurants as inputs will almost always serve an actual higher-rated restaurant over a misclassified low-rated restaurant. This way, it is highly unlikely that such a misclassification will ever reach the end user.

VIII. Future Work

Improving Similarity Measure. In the current implementation, the magnitude of the similarity measure between the test user and a member in the similarity group is used to assign a weight to the similarity group members rating for the restaurant in question. Thus, the weighting mechanism used is linear in the similarity measure. However, we suspect that this is non-optimal, since as the similarity group size increases, the weighted and unweighted similarity predictions converge due to a larger total weight denominator. Thus, we believe using a non-linear weight distribution model will better account for star rating predictions in large similarity groups.

Enhanced Low Star Predictions A few ideas that may improve the prediction of low star ratings is to subtract from a test users aggregate prediction the ratings of members in the similarity group with negative similarity. Furthermore, we believe it may be interesting to consider the businesses aggregate rating distribution and incorporate a random weighted variable from that distribution to the prediction.

Extension to Recommendation Engine Finally, we wish to use our rating prediction model as a part of an overall recommendation engine. The most simplistic way of implementing such a recommendation engine would be to rank all restaurants by predicted star rating for a given user and serve the recommendations in that order. More advanced algorithms can also incorporate location-aware services, sentiment analysis on what a user is currently craving, and social hot-spots. This recommendation engine would therefore attempt to curate a personalized dining experience for each user.

IX. Conclusion

In conclusion, we have shown that a user star rating of a restaurant can be predicted with high level of accuracy by using a similarity measure algorithm between Yelp users. By categorizing a users past reviews of restaurants into food categories, we formed a user feature vector by finding the average star rating of each type of food category the user has rated. The algorithm is also fast and computationally inexpensive since a recommendation is linear in the number of reviews of a specific restaurant, a users feature vector can be updated with new reviews in contant time, and an entire user can be onboarded in linear time with respect to number of reviews published. We believe the similarity measure algorithm with the feature vector we constructed can be used as the foundation of a restaurant recommendation engine. Such a recommendation engine can be useful in directed advertisements and help build a better user experience based on confident and trusted suggestions in Yelp's current and future services.

References

[1] J. Leskovec. Mining Massive Data Sets: Recommendation Systems Lecture Notes.

[2] B. Sarwar, G. Karypis, J. Konstan, J. Riedl. Item-based collaborative filtering recommendation algorithms. In Proceedings of 10th international conference on Wold Wide Web. ACM, New York, NY, USA, 285-295.

[3] JS Breese, D Heckerman, C Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In UAI'98 Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence.