Recommender Systems Designed for Yelp.com

Naomi Carrillo, Idan Elmaleh, Rheanna Gallego,
Zack Kloock, Irene Ng, Jocelyne Perez,
Michael Schwinger, Ryan Shiroma

Advisors: Dr. Alexander Ihler and Sholeh Forouzan

University of California, Irvine

August 17, 2013

Abstract

Although people's preferences are difficult to predict perfectly, providing reasonable predictions is highly useful, both economically and socially, in a broad spectrum of fields from marketing to demographics. Given such demand, developing and improving automated computational recommendation systems is a constant priority. The RecSys2013: Yelp Business Rating Prediction contest exemplifies one idea of such a recommender system – it requires an algorithm that will make high-quality predictions about a collection of specific user and business pairs. We use common existing recommender systems, such as nearest neighbor predictions, weighted averages, matrix factorization, and clustering, further customizing these traditional methodologies to include the distinct features unique to the Yelp data set, such as number of check-ins, user gender, and review counts. Our methods achieved an accuracy of 1.24039 RMSE, placing us at 51st of 401 at the time of submission on the Kaggle leaderboard.

1 Introduction

Collaborative filtering, a method used to make predictions based on a large data set, is a growing technique used for a number of different fields, one of the most common being electronic commerce. With the growing popularity of businesses like Amazon, Netflix, and Yelp, these companies gather vast amounts of clientele information. These continuously growing data sets create the need for methods to efficiently and accurately predict user preference as a marketing tool for the purpose of boosting sales by showing the customer similar things they would enjoy. Using collected knowledge about users, such as their ratings as an indicator of user preference, and known basic collaborative filtering methods, such as nearest neighbor or matrix decomposition, there are almost countless options to add different known factors to create more specific models and therefore more accurate predictions. These predictions become suggestions that, if accurate, will peak user interest and can improve sales or website traffic, or even both.

2 RecSys Challenge 2013: Yelp Business Rating Prediction

The bulk of our recommender systems centers around a competition hosted by the Association for Computing Machinery on Kaggle and organized by the Yelp. Kaggle, "the world's largest community of data scientists" [6], is a website that holds competitions of predictive modeling and analytics. The theme of the contest is to obtain personalized business ratings for each Yelp user. We are asked to predict the future ratings of each user on a one to five scale, where 5 stars is the highest rating. The model is graded based on its accuracy using the root mean squared error metric through the Kaggle website. Participating in this challenge became the basis for our summer research project.

2.1 Data

Our data came to us in two different files: a training set and a test set. In the training set, we have most of the information about both the users and business, such as average stars and ratings given by a user to a business. In the training set, we are given the following information:

- $11,537$ businesses
- $8,282$ check-in sets
- $43,873$ users
- $229,907$ reviews

In the test set,we are given the following information:

- $1,205$ businesses
- 734 check-in sets
- $5,105$ users
- $22,956$ reviews

More specific meta-data about the businesses in our data given include:

- business ID
- business name
- neighborhood
- full address
- city
- state
- latitude
- longitude
- star
- review count
- categories
- open status

For the users we are also given extra meta-data, such as:

- user ID
- first name
- review count
- average stars
- votes

3 Data Pre-processing

After analyzing our data, we found four distinct groups of user-business pairs. For the first set, we have ratings available for both the user and the business – this gives us the greatest opportunity to improve on our score and allows us to use standard collaborative filtering models. The next two sets have either business or user data available, but not both. The final set has no user information and very limited business information – we call this the cold start problem.

4 Predictive Methods

4.1 Mean Predictors

The simplest type of predictor is a mean predictor. The way mean predictor work is giving a prediction that consists of an average of all of a user's ratings, or an average of all of a business's ratings or a mixture of both averages. Mean predictors tend to be fairly fast compared to other prediction models; however, they are effective only when there is a sufficient amount of data provided to us.

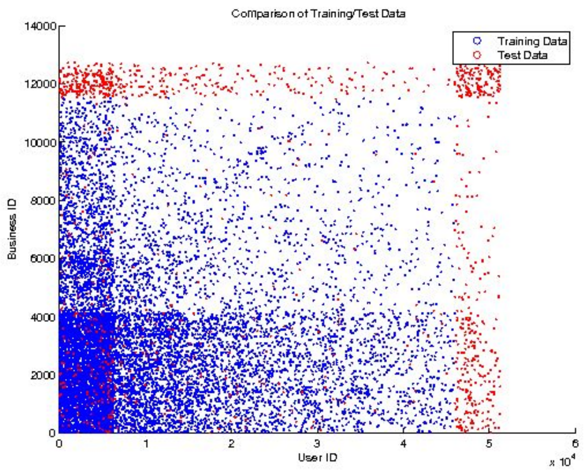

Figure 1: We see a large number of test ratings (red) from users and businesses from which we have no training data (blue).

Our baseline model takes into account the mean of the users and the mean of the items to predict a rating. We notate these as follows: μ is the average rating for all ratings included in the training set (for the Yelp set, $\mu = 3.7667$). u is the vector containing average ratings for each user and subtracting μ. b is the vector containing average ratings for each business and subtracting μ. $\hat{R}$ is the predicted ratings matrix.

We then uses the following equation [2] to predict user i's rating for business j:

$$\hat{R}_{ij} = \mu + u_i + b_j.$$

Using this predictor we scored a Kaggle RMSE of 1.25112, earning us 130th place.

When using mean predictors, one issue is the occurrence of "sandbag" ratings, which are low ratings, in the $1-2$ stars range, for a business that has a majority of ratings in the $4-5$ range. These lower ratings are said to "sandbag", which accounts for a good portion of error in our predictor. In order to reduce this error, we have to first find a way to determine whether or not a rating is a "sandbag" rating. A method to determine which ratings are "sandbag" ratings is to look at how many standard deviations from the mean. The standard deviation serves as a measure of the variance between each value to the mean. The values less than at minimum of three standard deviations are "sandbag" ratings. We then omit these "sandbag" ratings when computing the mean of our predictor. A business mean predictor without "sandbag" ratings scored a RMSE of 1.29371 on Kaggle, earning us 270th place. While better than the global mean benchmark, our method scored lowered than business mean benchmark; we hypothesize that the omission of "sandbag" ratings causes too large of a sparsity to accurately provide a good prediction.

Another issue is the instability of the mean predictors. For certain user-business pairs, a user mean predictor would perform better; whereas, for other user-business pairs, a business mean predictor would perform better. We attribute this instability to the difference in the amount of user information and the amount of business information. For user-business pairs with more business information, a business mean predictor would work better, while for user-business pairs with more user information, a user mean predictor would work better. To account for this instability, we proceed to average both the user and business means and weigh them proportionally to amount of business and user information available. This predictor received a RMSE of 1.25217 on Kaggle and a position of 135th place, which was significantly better than both the user mean benchmark and business mean benchmark on Kaggle, but scored worse than an unweighted combined mean predictor where the business mean and the user mean are equally averaged. This leads us to the conclusion that the number of ratings available may not have been an as good indicator of which mean predictor to use as we proposed. Another issue with this predictor is that for user-business pairs that we have no information on, we are forced to predict the global mean by default, which is not a very effective method.

4.2 Singular Value Decomposition

Singular Value Decomposition (SVD) is a latent factor method popularized by its massive success in the Netflix Prize competition. A matrix is approximated with SVD by multiplying two generated feature matrices P and Q with rank k.

$$\hat{R}_{ij} = P_i Q_j^T.$$

For our research, R is a ratings matrix with rows consisting of users and columns consisting of businesses. For example R_{ij} corresponds to the rating for user i and business j. We then minimize the squared differences between the ratings matrix R and the SVD approximated matrix, $\hat{R}$. We also subtract the global average rating, μ, to make the SVD prediction more accurate.

$$\hat{R}_{ij} = P_i Q_j^T + \mu.$$

$$\underset{P,Q}{\operatorname{argmin}} \sum_{(i,j)\in R} (R_{ij} - \mu - P_i Q_j^T)^2.$$

We also include a weighted lambda regularization (Tikhonov regularization) to avoid overfitting to our training set. [3]

$$\underset{P,Q}{\operatorname{argmin}} \sum_{(i,j)\in R} (R_{ij} - \mu - P_i Q_j^T)^2 + \lambda(||P_i||^2 + ||Q_j||^2).$$

In order to find optimum matrices for P and Q we implement the alternating least squares approach as described by Cichocki and Zhou. [3] [9] With this technique, we alternately solve for P and Q. An example of solving for P:

$$P_i = [Q_{I_i}^T Q_{I_i} + (\lambda I_k |R(i)|)]^{-1} (Q_{I_i} R(i))^T.$$

where,

- $Q_{I_i}$is the feature vectors of Q from businesses rated by user i.
- $R(i)$ is the ratings made by user i.
- I_k is the identity matrix of size k.

After obtaining our predicted ratings via SVD, we add the global mean back to each of our predicted ratings. Our best results for this method were found to have a rank (k) of 8 and a regularization constant (λ) of 0.55. Our SVD method scored an RMSE of 1.25622 and 153rd place on Kaggle.

4.3 Nearest Neighbors

When trying to find the best rating we can infer that similar people have similar taste. From that logic we can find ratings for users based on other user's ratings. This is called the nearest neighbor method. If user i has no rating for business j, we will go through our data set and filter out n users that have rated business j. We then determine the similarity between user i and each of the n users who rated business j via the Euclidean distance formula.

$$d_{ij} = \sum_{k}^{n} (user_i - user_k)^2$$

Using the ratings of user i for other businesses aside from business j, we compare these ratings to the ratings of n users who have rated business j. The user k with the smallest difference is the most similar to user i. We then take the rating given by user k and give the same rating to user i.

We can further complicate this method through specifying a weight to each of the n users who have rated business j based on the distances between two users. We the predict our ratings using the following equation: [7]

$$\hat{R}_{ij} = \mu + u_i + b_j + \sum_{k}^{n} (R_{kj} - \mu - u_k - b_j) \times d_{kj}$$

Users who have a smaller distance value with user i will contribute a higher portion towards the predicted value; whereas, users who have larger distance value with user i will contribute a smaller portion towards the predicted value. This simple method earned us a RMSE score of about 1.27229 and 212nd place on Kaggle.

4.3.1 Weighted Similarity – Jaccard Index

Using a weighted similarity predictor with Euclidean distance alone can present a problem. Consider two distinct Yelp users, user i and k. User i is someone who enjoys primarily Mexican restaurants and user k is an avid Mexican music enthusiast. Now consider a Mexican restaurant that has live Mexican music; it is possible that both user i and user k would rate the restaurant highly. Using a similarity predictor with Euclidean distance will declare these two users as users who are very similar, despite the fact the users rate completely different businesses. By considering user i and user k very similar, it can skew possible prediction ratings.

Another type of distance that can be used in the weighted similarity predictor is the Jaccard distance, derived from the Jaccard index. The Jaccard index is defined as:

$$J(i,k) = \frac{|I \bigcap K|}{|I \bigcup K|}$$

where,

- $J(i,k)$is the Jaccard distance between user i and user k
- I is the set of all ratings made by user i.
- K is the set of all ratings made by user k.

However, using this index as a distance for the weighted similarity predictor alone is highly inaccurate. Again, consider two distinct users, user i and user k. Now, consider some business j that both user i and user k have rated. Let user i rate business j only 1 star, and let user k rate business j 5 stars. Despite the fact that the users opinion on the business is very different, the users are considered similar through the Jaccard index.

An idea proposed by Laurent Candillier, Frank Meyer, and Francoise Fessant is to combine the Jaccard index with Euclidean distance [1]. By simply taking the product of the Jaccard index and Euclidean distance, it adds a weight to the Euclidean distance based on the similarity between the sets. Candiller, Meyer, and Fessant used this idea and applied it to data sets from MovieLens and Netflix, obtaining an accurate prediction set. [1]

Unfortunately, testing this method on the Yelp data set did not do so well, obtaining a RMSE of 1.32948 and 366th place on Kaggle, which was worse than both the user benchmark and the business benchmark.

4.3.2 Dealing with Sparsity and Cold-Start

Given the mechanism of the nearest neighbor method, we need to have enough users who have rated the same businesses to accurately select and compare neighbors; however, with the 99.9% sparsity of the Yelp data set, there exists no such neighbors for a large amount of users in our data set. Without enough neighbors, our primitive nearest neighbor method can only achieve a score of about 1.27229 RMSE and 212nd place on Kaggle. The weakness of our nearest neighbor method is caused by the inability to select neighbors. If user i and user k both have not rated the business j, then according to our model, they are automatically not considered as neighbors; thus, we may run the risk of prematurely disqualifying certain users as possible neighbors, who may provide meaningful contributions towards the prediction model. In order to counteract this sparsity problem and improve on our nearest neighbor method, Xiaoyuan Su, Taghi M. Khoshgoftaar, and Russell Greiner suggests boosting our nearest neighbor method with an imputation technique. This technique involves creating pseudo-ratings through substituting in the user's personal average ratings – in short, we assume that under normal circumstances, user i will most likely rate business j the same way user i rates any other business. Through this imputation technique, we are able to temporarily create more data to utilize in the neighbor selection process. [8]

Although this imputation technique works well, we quickly discover another problem within our Yelp data set: the cold-start problem. If we do not know user i's average rating, we cannot implement the imputation technique. A brief and trivial way to solve this problem is to use business j's average rating instead. This logic assumes that the ratings business j receives would most likely be around the business j's overall average rating for any user. With the current assumptions, we have solved most of the problems that we have encountered except the case where we do not have both business j's average rating and user i's average rating. Through analyzing the data set in terms of time-span, we can see that on average, a user's first rating tends to be around 0.0052 higher than the global mean; whereas, a business's first rating tends to be around 0.0547 higher than the global mean. With this idea, we can then assume that on average, any business's first rating received will be around 0.0052 higher than the global mean; this also applies to users.

Through temporarily filling in enough pseudo-ratings into our data set, we can accomplish a more thorough and accurate nearest neighbor model, which granted us a RMSE score of about 1.25155 and 132nd place on Kaggle.

4.4 Clustering

A clustering algorithm is effective when the data can be reduced to groups, where members of the group are similar to each other. Since members of a group are all similar, clustering assumes that such members can all be treated as the center of that group.

Before we begin clustering, we divide all the given data, including the business-user pairs to be predicted, into groups. We select restaurants, shopping, bars/nightlife, auto, and health because these are the categories that occur most frequently amongst businesses. One complication, however, is that some groups are combinations of other popular groups; for example the "health" group contains every business which are categorized as spas and health. On the other hand, for crossovers such as a food court residing within a shopping mall, the average of all ratings of businesses within a category is calculated, and the predicted rating is the average of all of a business's categories. We cluster over users after having segregated the businesses – it seemed more likely that there would be some similar groups of users for each of these smaller groups.

We must first choose centers for each group. While there are some ways to intelligently choose both the amount and the accurate position of centers, such as affinity propagation [4] [5], we instead use random placing. Through simple trial and error we then choose the number of centers for each subgroup. The distance between each data point and each cluster center is calculated, and each data point is assigned to the center that it is closest to. After all data points have proceeded through this process, the cluster centers are moved to the center, or mean, of all points assigned to that one cluster. The distances are then re-calculated. We repeat this process until the cluster centers cease to move (for time's sake, we stopped moving the centers once the distance between the old clusters and the new clusters is very small.) If any cluster center is assigned zero data points, it is reassigned to a random location.

Our clustering method is shown to be a relatively ineffective way of analyzing our data set, achieving a RMSE of about 1.30861 on Kaggle, placing us at 288th place. We hypothesize that the sparsity of our data set contributed to the inaccuracy of our clustering method.

5 Incorporation of Yelp-specific Feedback

5.1 Gender

We can improve our methods through the extra user information provided by Yelp – we will focus on gender in this section. Though Yelp does not directly provide us with a user's gender, we can simply assign a gender to each user based on their first name. After compiling a list of n users who have rated business j, we can increase the weight of user k if user k shares the same gender as user i. This logic relies on the assumption that users of the same gender will rate similarly. Implementing this logic into our nearest neighborhood method, we achieved a RMSE score of about 1.24977 and 122nd place on Kaggle, which confirms our assumption that users of the same gender tend to rate similarly.

5.2 Yelp Review Votes

A feature unique to the Yelp website is the ability for any Yelp user to read and rate another user's review for a business. A user has the option of voting a review to be "funny", "useful", "cool", or any combination of the three. Since the terms "funny", "useful", and "cool" portray a positive connotation, we assume that a review that has acquired an abundance of such votes is more likely to substantially affect another user's opinion of a business. From this ideology, we place a heavier weight on user's reviews that have been voted "funny", "useful", or "cool".

Deciding the weights for the "funny", the "useful", and the "cool" votes raised further questions. The "useful" rating seems to be the most accurate; however, we cannot disregard the social influence of a "cool" vote or a "funny" vote. We test our hypothesis by equally weighing each vote option. Users with the most votes on their reviews are considered to contribute more weight towards making our prediction. This simple assumption did relatively well, scoring a RMSE of 1.28893 and 255th place on Kaggle, which was bested the user mean benchmark, business benchmark, and global mean benchmark.

6 Data Post-processing

A consequence of the abundance of cold start test data is that many of our predictive methods had a hard time giving quality predictions for a large percentage of the test data. The most informative group, the known user/known business group, contributes to only 28% of the test data. The known user/unknown business group and the unknown user/known business group consists a total of 27% and 33% respectively. The remaining group of which both user and business ratings are unknown consists a total of 12%. Table 1 shows which methods were most suitable for each group based on the results we received in validation.

Table 1: Data Splitting

	Known Business	Unknown Business
Known User	Neighborhood Clustering SVD	User means
Unknown User	Business means Business clustering Category means	Category means Global mean

A concluding technique to all of our models is the correction for global effects. [7] With a large sample size we assume the mean of the test set would be approximately the same as the mean of the training set. With this inference, we shift the predicted ratings by a constant value (τ) such that the mean of the predicted ratings equals the training mean. [9]

$$\tau = mean(R) - mean(\hat{R}).$$

7 Blending and Results

Once we optimize all of our models, we can perform a linear blending algorithm. We weigh each model based on how well their respective RMSE scored on the Kaggle leaderboard. After blending our methods, we achieved a score much lower than our best individual predictor. Our blended predictions received an RMSE of 1.24039, which placed us 51st out of 401 at the time of submission. Refer to Table 2 for the results of each method and our blended predictor.

8 Related Work

Since the Netflix prize competition, there has been a lot of research in the area of online rating prediction. Research on the Netflix data set was very helpful in our attempts. However, in the case of the Yelp data set, much less data is provided. This can be a result

Table 2: Results

Method	RMSE	Kaggle Ranking
Mean Predictor	1.25112	130th
Weighted Mean Predictor	1.25217	135th
SVD	1.25622	153rd
Boosted Nearest Neighbor	1.24977	122nd
Clustering	1.30861	288th
Vote Weighting	1.28893	255th
Jaccard Index Similarity	1.32948	366th
Blended Model	1.24039	51st

of Yelp data being localized to the users' location and users being less likely to rate a restaurant of average quality than any movie on Netflix. With a relatively large amount of sparsity, basic means and averages models surprisingly perform at or better than standard collaborative filtering techniques. [2] Our sparse modified nearest neighborhood method took advantage of this sparsity problem and performed the best out of our individual predictors. We also found research on incorporating implicit data, which further assisted our final results.

9 Future Work

For future work, the first thing we would also like to incorporate is temporal information in our methods. Generally, test data for these competitions is designed with future rating prediction in mind. Therefore if we can find trends in business and user ratings over time, we can better predict ratings with a moving average. We would also like to use check-in information to find whether differences in prediction accuracy occur with more popular businesses.

References

[1] Laurent Candillier, Frank Meyer, and Franoise Fessant. Designing specific weighted similarity measures to improve collaborative filtering systems. In Petra Perner, editor, *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, volume 5077 of *Lecture Notes in Computer Science*, pages 242–255. Springer Berlin Heidelberg, 2008.

[2] Phil Chen and Dan Posch. Collaborative filtering on very sparse graphs a recommendation system for yelp.com, November 2012.

[3] Andrzej Cichocki and Rafal Zdunek. Regularized Alternating Least Squares Algorithms for Non-negative Matrix/Tensor Factorization. pages 793–802. 2007.

[4] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, February 2007.

[5] Inmar E. Givoni and Brendan Frey. A binary variable model for affinity propagation. *Neural Computation*, 21:1589–1600, June 2009.

[6] Anthony Goldbloom. Kaggle, 2010.

[7] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4:1–24, 2010.

[8] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:19, 2009.

[9] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. 2008.