Yelp Rating Prediction for Different Cities Using Word Vector Similarity between Friends

Janette Cheng, Jintian Liang, Rafael Musa

December 11, 2016

1 Introduction

Social networks can provide powerful information to services that generate recommendations for their users. Previous research has already shown that incorporating information and opinions from users' friends can improve upon standard collaborative filtering systems. Indeed, using information about a user's social graph can be especially useful when individual users have not left many reviews, especially considering that friends are more likely to have similar tastes.

Our project looks at predicting star ratings for businesses using data from Yelp. We aim to expand upon the work done in "Predicting Yelp Ratings Using User Friendship Network Information," a paper from last year's CS224W class. Yang, Yuan, and Zhang used a modified latent factor model which takes into consideration each user's friends when predicting reviews. Our model also uses information from the Yelp friendship network; however, the utility of incorporating friends' opinions may vary depending on the population being considered. For example, friends' opinions may be more valuable for predicting reviews in more homogeneous cities. Thus, we evaluate our model on three subgraphs corresponding to different cities—Waterloo, Ontario; Edinburgh, Scotland; and Charlotte, North Carolina—which vary in size, culture, and diversity.

One could also see how friends might have different preferences. For example, one friend could care a great deal about service at a restaurant while the other only cares about quality of food. Thus, we also incorporate semantic information from review text (specifically in the form of word vectors produced by the GloVe algorithm developed by Pennington et al.) by ignoring edges to dissimilar friends. We did not come across a model in our research that similarly uses a social network in conjunction with users' written reviews in order to predict ratings. The incorporation of word vectors into our model allows us to use some of the richest information available in the Yelp dataset.

2 Related Work

In "A Social Network-Based Recommender System (SNRS)," He and Chu propose an alternative to collaborative filtering called the "social network-based recommender system" (SNRS). In order to predict a user U's rating for an item I, the system incorporates U's preference for items similar to I, general acceptance of the target item I, and the ratings of U's friends. The authors also put forward an "iterative classification" method that incorporates the ratings of friends of friends to increase cover and ameliorate the cold-start issue, which is a known problem with using collaborative filtering to predict the reviews of users who have not left many (or any) reviews. SNRS was evaluated on restaurant reviews from Yelp. In general, 18.6% of the restaurants reviewed by a user were also reviewed by at least one of his/her friends (more than chance). They also found that if two reviewers are immediate friends, their reviews differ by 0.88 on average versus 1.05 for non-friends. The authors evaluated their recommendations against collaborative filtering and Naive Bayes models, using the average rating of immediate friends, and using the ratings of friends weighted by cosine similarity to the friend. SNRS was found to have the lowest mean average error (MAE) and lower MAE when the system does not take into consideration friends of friends (0.716 without versus 0.682 with).

He and Chu suggest several future directions, the most promising of which is semantic filtering (e.g. filtering to look at reviews from similar users that are focused on food to predict a rating specifically for food, looking at semantics in friend relationships, etc.). This was found to be helpful when evaluated on a dataset from 22 students who were asked to rate articles based on 3 factors and to specify which of the other students they agreed/disagreed with on average for each of these 3 factors, suggesting that extracting semantic information from review text and focusing on semantically similar reviews could improve predictions.

Koren et al. focus on latent factor models for collaborative filtering. Latent factor models characterize items and users based on around 20 to 100 factors. For items, each factor measures a quality such as comedy versus drama, depth of character development, etc. For users, each factor measures how much a user tends to like movies with high scores on the corresponding movie factor. The model presented is based on matrix factorization. The factors are stored in vectors q_i (for the item) and p_u for the user. The most basic prediction of user u's rating of item i (r_{ui}) is the dot product of the item and user vectors ($q_i^T p_u$). In order to compute the mapping from item/user to vectors q_i/p_u, the system minimizes the regularized square error using either stochastic gradient descent or alternating least squares. Additional inputs may be added to improve the rating, the most basic of which are the item bias (b_i) and the user bias (b_u), which capture the general acceptability of the item and how critical the user generally is, respectively. Our model uses an additional bias to capture how much users are influenced by their friends (see Section 4). One can also augment the representation of the user or item by adding additional vectors to p_u or q_i. For example, the authors suggest that one could add vectors that correspond to different demographic factors to p_u.

Yang et al. focus on the phenomenon of homophily. That is, users tend to like similar items that their friends do and also befriend people who like similar items. As such, Yang et al. build on previous work on latent factor models for collaborative filtering by also including a social network aspect. For each user i and item j, the authors incorporate latent features ϕ_i and ϕ_j in addition to intrinsic features x_i and x_j to perform item recommendation. However, they perform a similar process for friendship prediction, given some user i and another user i'. The latent factors are then learned using stochastic gradient descent. Friendship prediction is then incorporated into item recommendation. The authors evaluated their Friendship-Interest-Propagation (FIP) model on the Yahoo! Pulse data and found that their model performed better than existing interest targeting models.

McAuley and Leskovec set out to combine user numerical ratings and text reviews to improve predictive power. Instead of using the latent factor model, which only uses the ratings, or latent dirichlet allocation, which focuses on the review text, they devise a minimization problem that factors in the ratings as well as the likelihood of the review text. In "GloVe: Global Vectors for Word Representation", Pennington et al. propose a vector space representation of words that could also be used to incorporate review text into rating prediction. In our model, we hope to take inspiration from McAuley and Leskovec's integration of rating and review information, and try to incorporate the word vectors into the standard latent factor model.

3 Network Characterization

The Yelp dataset does not give information about users such as age, gender, ethnicity, or location. Thus, in order to reason about the results obtained in Section 5, we give information below about each of the three networks (Waterloo, Edinburgh, and Charlotte) in terms of graph characteristics.

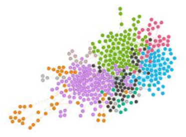
Figure 1: Waterloo

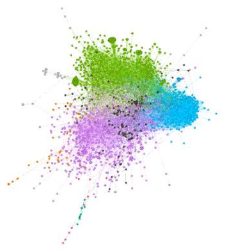
Figure 2: Edinburgh

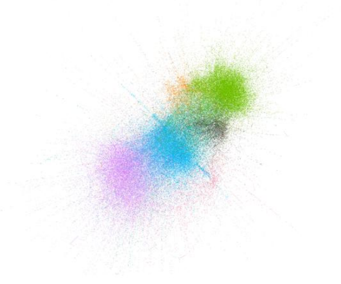
Figure 3: Charlotte

Figures 1, 2, and 3 are visualizations of the networks from Gephi laid out using the ForceAtlas 2 algorithm. Nodes have been colored by modularity class (unique to each network), calculated using the Louvain Method for community detection from Blondel et al. implemented in Gephi. We do not have demographic data for users, so it is difficult to determine what members of different communities have in common. However, there are some more general trends. For example, both Waterloo and Charlotte are more oblong and have communities that are farther away from each other in the network, which is consistent with their relatively higher modularity scores (see Figure 4). Edinburgh, on the other hand, has three large communities that are all adjacent in the network.

City	Nodes	Edges	Clustering Coeff.	Diameter	Modularity	Num Comm.	Largest Comm.
Waterloo	458	946	0.053	10	0.60	34	25.11%
Edinburgh	1767	9058	0.12	9	0.49	81	35.77%
Charlotte	14179	49072	0.043	14	0.58	685	22.74%

Figure 4: Summary network statistics

The table above gives summary statistics for the three networks. "Largest Comm." is the percent of nodes that belong to the largest modularity class, and "Num Comm." is the number of modularity classes (both calculated using the Louvain Method). Note that while Waterloo is considerably smaller than Edinburgh in terms of number of nodes, the diameter of Waterloo is actually larger. Overall, it seems Edinburgh is the most connected, cohesive network as it has the highest clustering coefficient, smallest diameter, lowest modularity, and highest percentage of nodes in the largest modularity class. Waterloo and Charlotte, on the other hand, are relatively similar when it comes to modularity and clustering coefficient, which indicates that their networks may be more segmented or diverse.

Degree Distributions

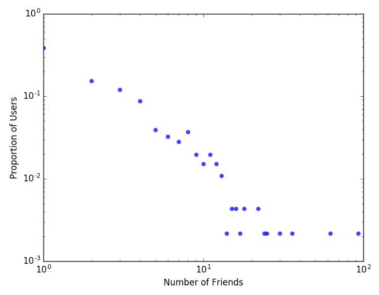

Figure 5: Waterloo

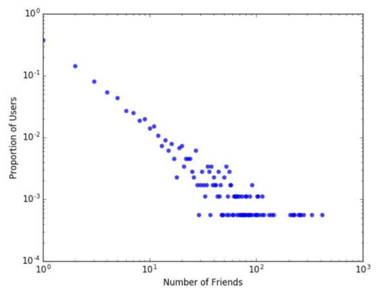

Figure 6: Edinburgh

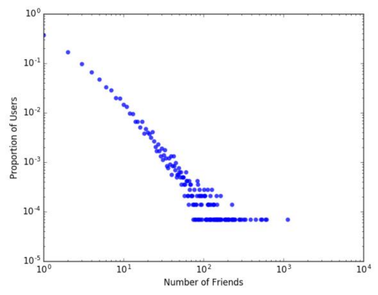

Figure 7: Charlotte

Figures 5,6, and 7 show the degree distributions for each city. While there are generally higher degree nodes in the larger networks—with nodes in Charlotte having higher degree than those in Edinburgh, that in turn have higher degree than those in Waterloo—the shapes of all three distributions are similar. Generally speaking, cold-start is an issue when predicting reviews on Yelp using collaborative filtering. Most users have very few or no friends, so all three networks are quite sparse. The distribution of the number of reviews per business is distributed similarly, with most businesses having very few reviews. Additionally, most users have left very few or no reviews.

Figures 8,9, and 10 show the distribution of star reviews for each city. More than half of all reviews are above 3 stars across all three cities, indicating that users are generally quite generous in their evaluations of businesses. The most common star review for both Waterloo and Edinburgh is 4, whereas the most common star review for Charlotte is 5. Thus, it can be helpful to run the latent factor model for individual cities because α from $(*)$ in Section 4 varies between locations.

Distribution of Star Reviews

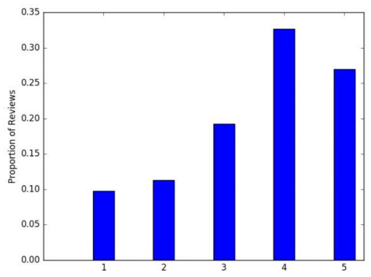

Figure 8: Waterloo

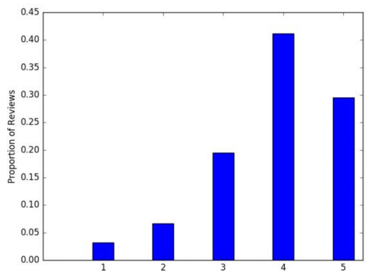

Figure 9: Edinburgh

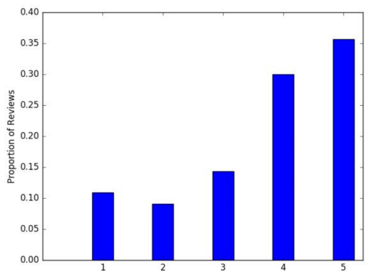

Figure 10: Charlotte

To incorporate semantic information from review text, we propose creating a vector for each user by averaging the word vectors of words used in all reviews written by that user. We used the 50-dimensional pre-computed word vectors from the GloVe project and pre-processed the review text by stemming and removing stop words. For the purposes of this project, we found that training the word vectors on the corpus of Yelp review text was not particularly beneficial (the distributions discussed below did not change in any meaningful way). In order to determine the similarity between two users' vectors, we tried two different measures of similarity—cosine similairty and Euclidean distance—and compared their performance in the model.

Figures 11 and 12 show the distribution of cosine similarities between the word vectors of friends and random pairs. The random pairs were chosen by picking d random nodes for each node n with degree d. Overall the word vectors are quite similar (most cosine similarities are close to 1); however, it does seem like friends tend to have more similar word vectors than random pairs, implying that friends do indeed tend to be more semantically similar in their reviews. Ignoring friends that have relatively low cosine similarity is then potentially a way to only consider the reviews of those that are closest to the user.

Distribution of Cosine Similarities

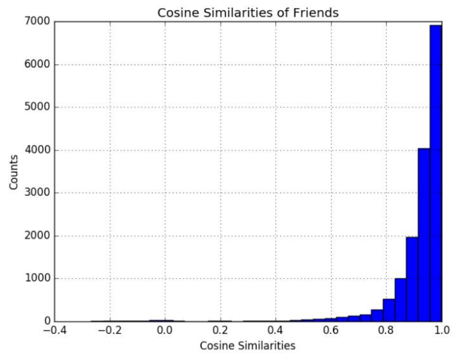

Figure 11: Friends

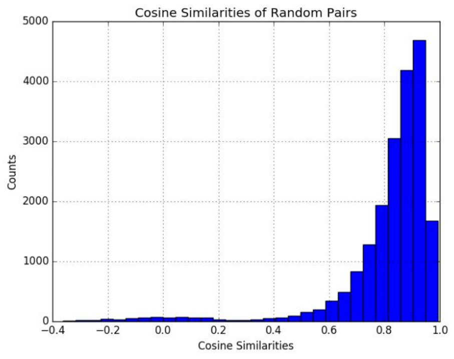

Figure 12: Random pairs

From Figures 13 and 14, we can see that the distribution of Euclidean distances is basically the reverse of what we saw for cosine similarity. Almost all distances are below 3, and the distances between friends are generally smaller than those between random pairs. This again shows that the word vectors for users that are friends are more similar. The results of using cosine similarity versus Euclidean distance to ignore dissimilar friends is discussed in Section 5 (see Figure 15).

Distribution of Euclidean Distances

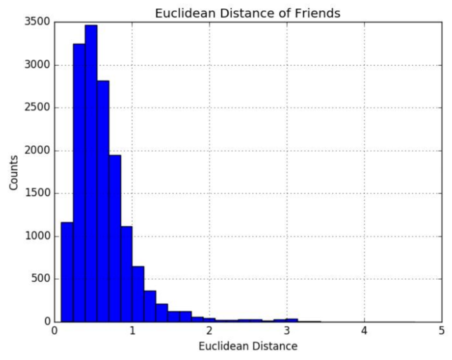

Figure 13: Friends

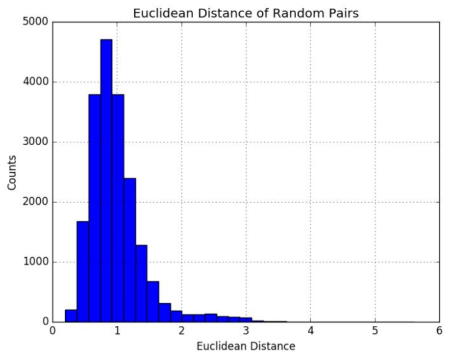

Figure 14: Random pairs

4 Models

First, we implemented a very simple baseline model as a sanity check. This model always predicts α, the average star rating across all reviews in the training set for a given network. For example, if we are looking at Edinburgh, the baseline model would predict the average of all star ratings for businesses located in Edinburgh regardless of which (user, business) pair we are considering.

We based our recommender system on the latent factor model used by Koren et al., Yang et al., and McAuley & Leskovec. This model includes the baseline α, one bias term β_u for users, one bias term β_i for businesses, and two k-dimensional vectors: p_u for user and q_i for business factors. These factors encapsulate the hidden relationships that give name to this model. Intuitively, each entry in the business factor vector represents some attribute of the business being reviewed and the corresponding entry in the user factor vector represents how much the user enjoys that attribute. β_u captures how critical the user is and β_i captures the general acceptance of the business (i.e. how much users generally like the business). $\hat{r}_{u,i}$, the predicted rating of user u for business i, is defined as follows:

$$\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + p_u \cdot q_i$$

Note that α is just the average rating across all reviews, which is not learned. All other parameters are learned.

Our goal was to incorporate information about the social network into the latent factor model. In order to do so, we added a new term to our predicted review expression. In this model, each user has a friends bias f_u which captures how much a user is affected by ratings from his or her friends. We let $F(u)$ be the set of all friends of user u.

$$\hat{r}_{u,i} = \begin{cases} \alpha + \beta_u + \beta_i + p_u \cdot q_i & \text{if } |F(u)| = 0, \\ \alpha + \beta_u + \beta_i + p_u \cdot q_i + f_u \left(\frac{1}{|F(u)|} \sum_{u' \in F(u)} r_{u',i} - \alpha \right) & \text{otherwise.} \end{cases} \tag{*}$$

We implemented this model using stochastic gradient descent with L_2 regularization, minimizing the mean squared error of the prediction versus the real review value. We tuned hyperparameters such as step size η for gradient descent, the coefficient λ for regularization, and k by seeing which combination worked better from $\lambda \in \{0.2, 0.4, 0.6\}, k \in \{10, 20, 30, 40, 50\}, \eta \in \{0.01, 0.05, 0.1\}$. For our experiments, the optimal values were $\lambda = 0.4, k = 20, \eta = 0.01$. We cached the average friend rating for each (user, item) pair to avoid recomputing the quantity.

Finally, we incorporated similarity measures between the word vectors to improve the model. We considered cosine similarity and Euclidean distance between the word vectors. Cosine similarity is normalized between 0 and 1, so it measures only the semantic similarity between the reviews from different users. The Euclidean distance between word vectors is not normalized, so it also takes into account the amount written by each user (which affects the magnitude of the vector) when computing a measure of similarity between two friends.

We define the sets $F_{cos}(u) = \{x | x \text{ is a friend of } u \text{ and } \texttt{CosSim}(\texttt{WordVec}(x), \texttt{WordVec}(u)) > 0.9\}$ and $F_{eucl}(u) = \{x | x \text{ is a friend of } u \text{ and } \texttt{EuclDist}(\texttt{WordVec}(x), \texttt{WordVec}(u)) < 0.8\}$. The thresholds of 0.8 and 0.9 were determined by examining Figures 11 and 13 as well as testing various different thresholds on a subset of the network. $F_{cos}(u)$ and $F_{eucl}(u)$ are both subsets of $F(u)$ that represent the friends that are closest to u in terms of word vector similarity (we "prune" friends that are relatively dissimilar). We use the equation $(*)$ to predict ratings using each of these two sets as the friends set.

5 Results and Evaluation

In order to evaluate our results, we used a baseline model and a null model. As discussed previously, the baseline model always predicts the average review value in the training set. The model is very basic, and is used as a sanity check to make sure our methods lead to reasonable improvements in prediction.

For the null model, we followed the configuration model to reconstruct a friend network with the same degree distribution but randomized edges, and trained the prediction model using that network as the friend network. We can compare our models to the results obtained by this null model to understand whether and to what extent including friends and their word vectors improves our prediction accuracy.

The table below summarizes our results for the simple baseline, the null model, using all friends, and pruning friends based on cosine similarity and Euclidean distance between their word vectors. We ran cross validation with ten folds, and we report the mean (Fig. 15) and standard deviation (Fig. 16) of ten estimates of MSE for each pair of model and city:

City	Baseline	Null Model	All Friends	Cosine Pruning	Euclidean Pruning
Charlotte	1.755	1.493	1.488	1.486	1.488
Edinburgh	1.025	0.958	0.928	0.930	0.929
Waterloo	1.597	1.469	1.462	1.447	1.450

Figure 15: Average Mean Squared Error

City	Baseline	Null Model	All Friends	Cosine Pruning	Euclidean Pruning
Charlotte	0.0012	0.011	0.017	0.0094	0.019
Edinburgh	0.003	0.032	0.020	0.028	0.022
Waterloo	0.010	0.096	0.078	0.083	0.086

Figure 16: Standard Deviations

For all models, we note that the baseline performed worst, followed by the null model, followed by the models including friends. This suggests that including the user and business biases and the latent factor vectors for users and businesses improved prediction, and that adding information about the reviews from friends led to further improvement.

To quantify whether the improvements are statistically significant, we run the two-sample Welch's t-test, which allows for different variances per sample. We note that the baseline estimates all have the smallest standard deviations, which is expected since no learning occurs and averaging the score of all predictions is robust to changes in training and test sets. We obtained the p-values below when testing results obtained from different methods.

City	Null vs. Baseline	Null vs. All Friends	All Friends vs. Euclidean	All Friends vs. Cosine
Charlotte	6.75e-15	0.45	1.0	0.75
Edinburgh	9.25e-5	0.02	0.91	0.86
Waterloo	0.0022	0.86	0.74	0.68

Figure 17: p-values

We note that the improvement of the null model over the baseline is statistically significant in all cases. The model using friend data performs significantly better than the null only in Edinburgh. We speculate that because Edinburgh is more ethnically and culturally homogeneous than the other cities, people are more likely to have similar tastes to their friends on Yelp. This homogeneity is reflected in the Edinburgh friendship network, which has the lowest modularity, smallest diameter, and highest clustering coefficient (discussed in Section 3). The better performance of the all-friends model in Edinburgh is also notable considering that the friendship networks for all three cities were rather sparse. The size of the city/network did not seem to make a clear difference, as predictions for Waterloo (the smallest city in terms of number of nodes) did not do much better or worse than those for Charlotte (the biggest city).

We also see that ignoring dissimilar friends based on Euclidean distance or cosine similarity of their word vectors does not have a statistically significant effect. It seems like using cosine similarity is marginally better than using Euclidean distance, but we cannot make any conclusive claims with p-values this high.

Overall, our results reveal that using the latent factor model is an effective way to predict Yelp reviews, but that adding information about the friendship network does not necessarily always improve on that model. In fact, it seems whether or not using the friendship network improves predictions is related to characteristics of the network and city being considered

6 Further Work

The cities (Charlotte, Edinburgh, and Waterloo) that we used for this project were chosen based on their traits and the tractability of implementation. There exist other cities, such as Las Vegas and Phoenix, that would be interesting to analyze due to their diversity; however, we could not do so because of the sheer number of reviews we would have to process. It would also be insightful to build a classifier to categorize the cities into groups and see how our model performs for cities belonging to the various groups. This is a natural extension of our current project, where our model performs much better for Edinburgh than it does for Charlotte and Waterloo.

For the natural language processing aspect of our project, we only analyzed various similarity measures using word vectors. There are other areas of NLP yet to be explored. For example, it would be interesting to see if performing sentiment analysis on the review text could improve the model even further.

In addition, the Yelp dataset includes some user information that we have excluded. This includes the years in which the user attained Elite status (a Yelp feature), the compliments they received for their post (”cool”, ”useful”, etc), and the number of fans they have. These features could have been utilized in the machine learning process to improve user representation and therefore generate a potentially better model. The Yelp dataset does not provide demographic information about users, but natural language processing techniques could be used to infer different factors. For example, a classifier could be run on users' names to guess their gender, which could then be used to determine if including information about gender can produce better predictions.

Contributions

Janette: Gephi visualizations, finding papers for related work, cross validation, word vector processing, and network characterization writing, data, and plots

Jintian: Pre-processing and formatting data, network analysis to decide on cities, null model generation, writing further work section

Rafael: Coming up with the algorithm (null model, and building on gradient descent algorithm from previous paper), statistical analysis, writing up model and evaluation sections

References

[1] He, Jianming and Chue, Wesley W. “A Social Network-Based Recommender System (SNRS).” *Data Mining for Social Network Data*, pages 47-74. 2010.

[2] Koren, Yehuda, Bell, Robert, and Volinsky, Chris. “Matrix Factorization Techniques for Recommender Systems.” *IEEE*, pages 42-49. 2009.

[3] Yang, Shuang Hong et al. “Like like alike - Joint Friendship and Interest Propagation in Social Networks.” In *Proceedings of the 20th international conference on World wide web*, pages 537-546.

[4] McAuley, Julian, and Leskovec, Jure. “Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text.” In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165-172.

[5] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. "GloVe: Global Vectors for Word Representation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[6] Blondel, Vincent D, Guillaume, Jean-Loup, Lambiotte, Renaud, and Lefebvre Etienne . "Fast unfolding of communities in large networks." In *Journal of Statistical Mechanics: Theory and Experiment*, page 1000. 2008.