Data Science AC209a / Harvard University

Analyzing Phoenix Yelp reviews

Nick Hoernle, Sophie Hilgard, Andrew Ross, and Reiko Nishihara

Harvard School of Engineering and Applied Sciences
IACS Institute for Applied Computational Science

INTRODUCTION

For our take on Data Science with User Ratings and Reviews, we decided to try working with data from the “Yelp Dataset Challenge” [1]. Our goal was to explore interesting trends in the dataset, as well as to try out techniques for training recommender systems as outlined in [2] (BellKor’s Pragmatic Chaos solution to the Netflix Grand Prize).

DATA EXPLORATION

This was one of the key milestones of the project and we explored many aspects of the data:

Baseline analysis: The data contain 2.7M reviews from 687K users for 86K businesses in total. Review counts are very left skewed, with only 10.3% of users giving more than 50 reviews -- although the top 692 users have more than 1,000 reviews each.

User Analysis: Users typically are one dimensional in their reviews yet they may disagree dramatically on businesses. **Figure 1** to the right shows that a large majority of users give identical ratings, but businesses tend to receive fairly varied ratings.

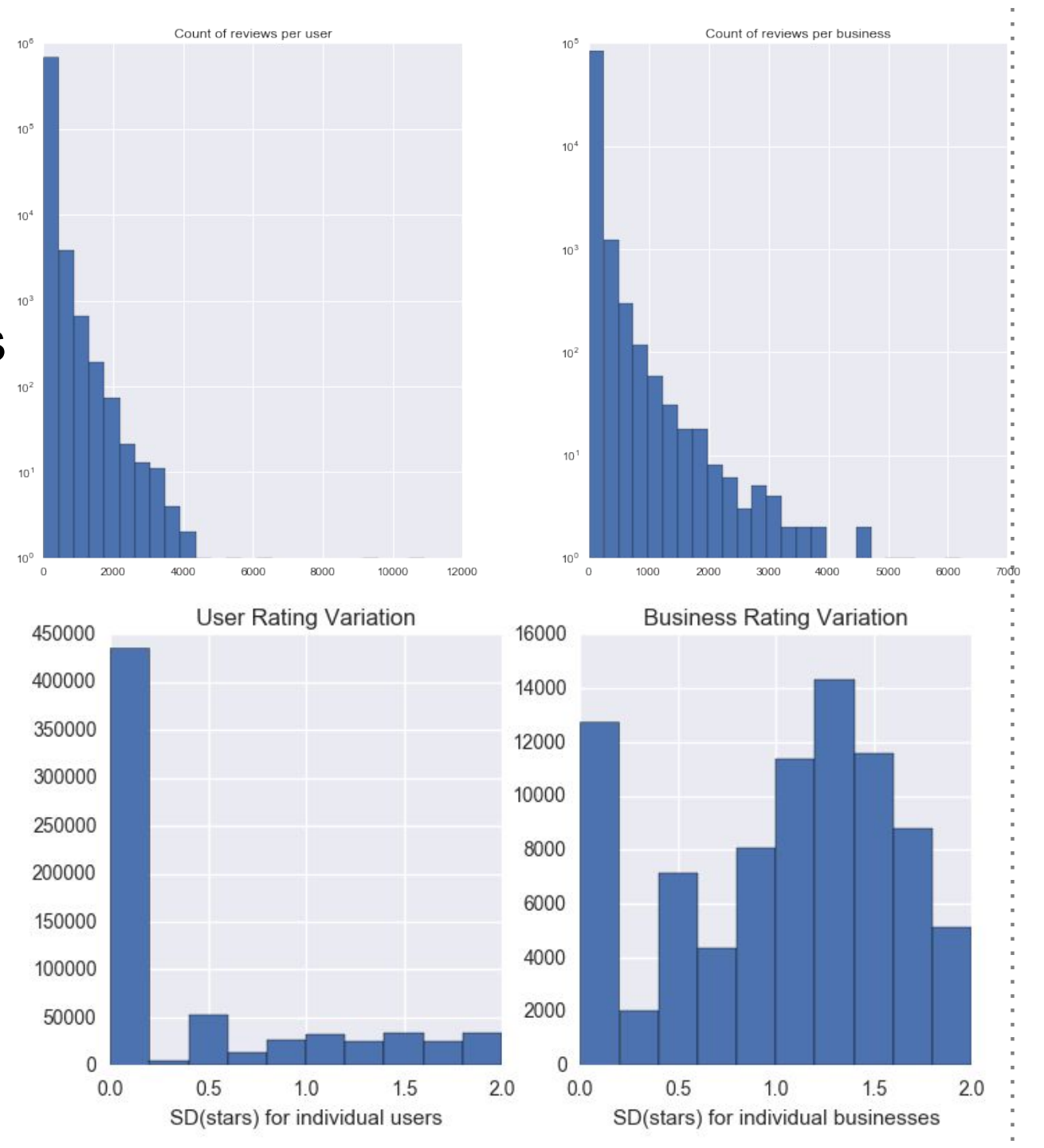


Figure 1: Data is heavily skewed: Many users only give a single review and many businesses only receive several. Despite the similarity in skewness, however, users tend to display much less variation in their reviews than businesses.

Location based analysis: We clustered businesses by latitude and longitude and plotted the average star ratings for the resulting clusters of still-open businesses. We found that businesses in the vicinity of fewer competitors are able to stay open despite lower ratings.

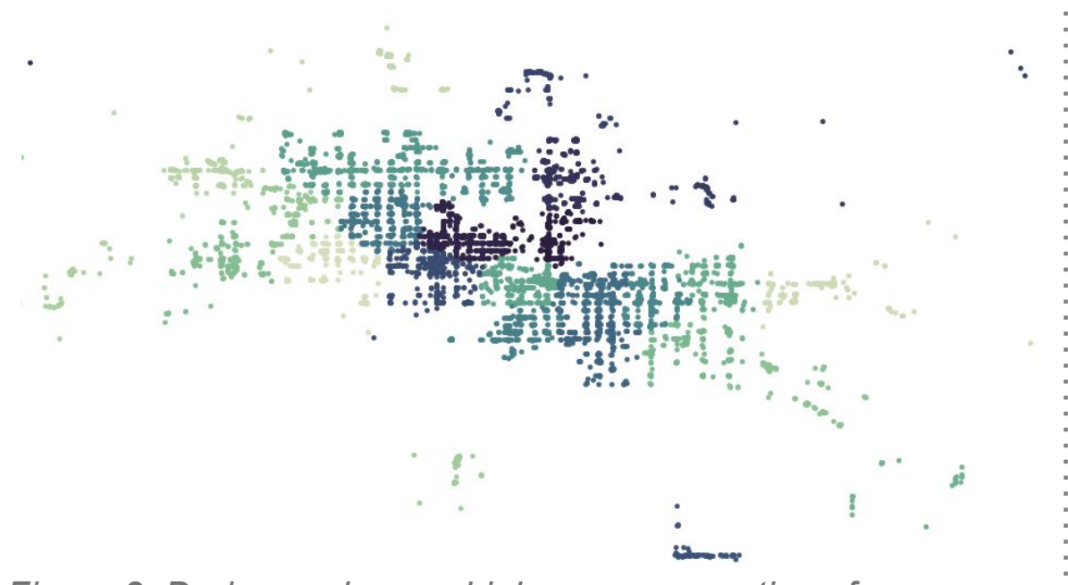
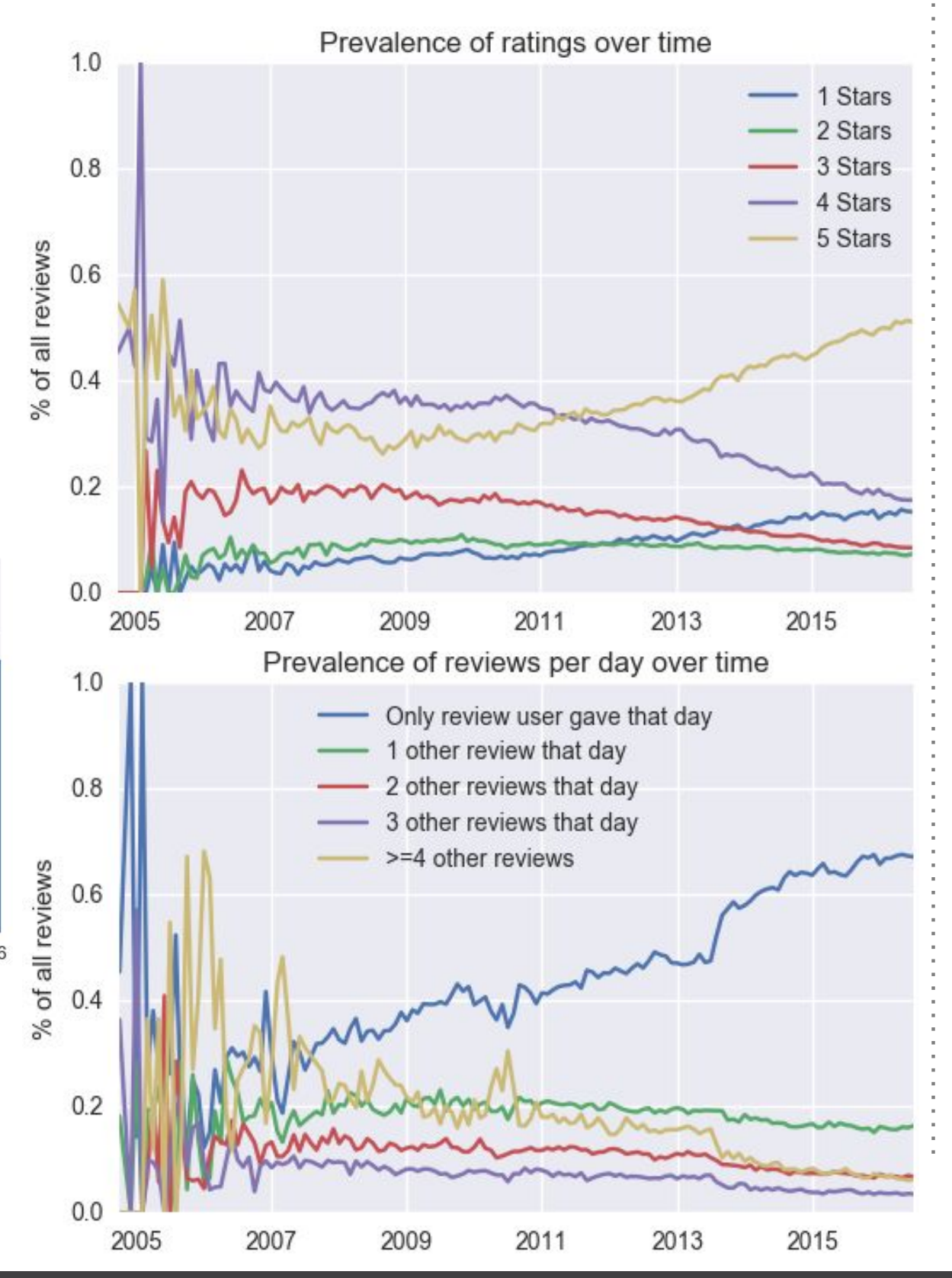


Figure 2. Darker nodes are higher average ratings for open businesses, whereas lighter nodes are lower average ratings.

Time based analysis: We found differences in the distribution of ratings over time, which turned out to be partially explained by differences in user behavior when giving one vs. many reviews per day -- a phenomenon also noted in [2].



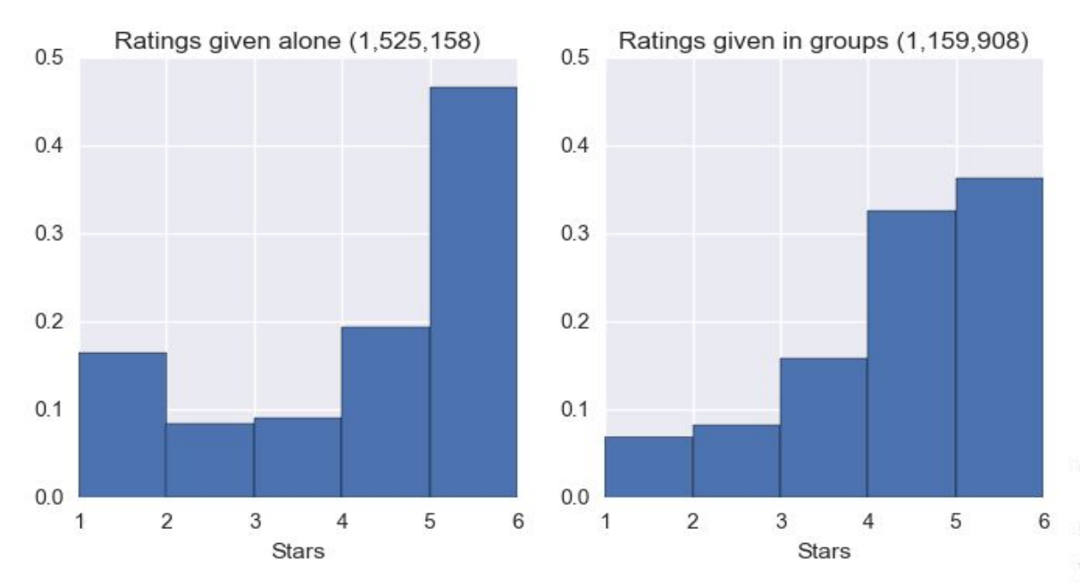


Figure 3. Above: Users who only give one review per day tend to rate businesses more extremely (more 1s and 5s). Right: We can see that 1s and 5s have become more prevalent over the past few years, but below that, so have users who only give a single review per day. Even when you control for year, the increased-1s-and-5s effect remains.

BASELINE MODELS AND PREDICTORS

To evaluate models, we first limited ourselves to considering just restaurants and bars in Phoenix, which was the city with the most reviews (647,792). Within Phoenix, we split our reviews 70%-30% into a training and test set we shared across analyses.

To train models, an important prerequisite is generating baseline predictors for users and businesses (aka items, subscripted i). These can also be used as standalone models for predicting ratings $\hat{r}_{ui}$ using:

$$\hat{r}_{ui} = \mu + b_u + b_i$$

Baselines b_u and b_i account for “intrinsic” item likeability / user effusiveness, and explain most of the variation in ratings. By computing very accurate baselines, BellKor [2] was able to outperform Cinematch without any user preference modeling. We tried the following methods:

Always predicting μ: (RMSE: 1.354) $b_u = b_i = 0$

Simple averaging: (RMSE 1.347) $b_* = \frac{\sum_*(r_* - \mu)}{|R(*)|}$

Beta prior inspired method: (RMSE 1.229 for α=8, a+b=α)

$$b_* \sim 4 \cdot \mathrm{Beta}\left(a + \alpha \sum_* \frac{r_* - 1}{4},\ b + \alpha \sum_* \frac{5 - r_*}{4}\right) - \mu + 1$$

Decoupled regularized avging: (RMSE 1.2247 for λ_1=5.25, λ_2=2.75)

$$b_i = \frac{\sum_u (r_{ui} - \mu)}{\lambda_1 + |R(i)|}, b_u = \frac{\sum_i (r_{ui} - b_i - \mu)}{\lambda_2 + |R(u)|}$$

Least-squares + L2 method: (RMSE 1.2251 for λ_3=4.5)

$$\min_{b*} \left[\sum_{u,i} (r_{ui} - \mu - b_u - b_i) + \lambda_3 \left(\sum_u b_u^2 + \sum_i b_i^2 \right) \right]$$

We wrote a memory-conscious gradient descent algorithm to solve this without crashing Python!

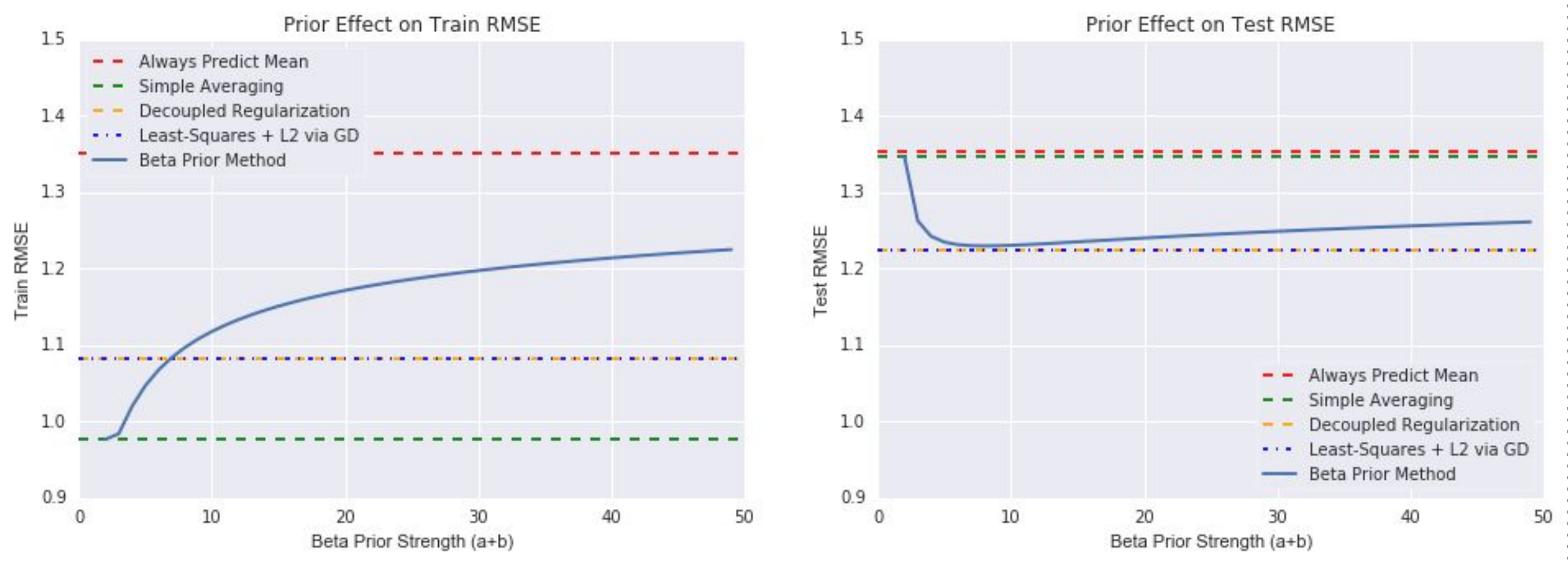


Figure 4: comparison of several methods of computing baselines, showing both train and test RMSE, as a function of the prior strength for the Beta method. Note first how much simple averaging overfits -- it has the lowest train RMSE, but is barely better than predicting a constant value on the test set. Note also how the decoupled regularization and least squares L2 methods arrive at the same RMSEs (and, it turns out, end up with the most similar values). The Beta method initially is identical to simple averaging, but improves to match the regularized methods for the best prior strength value.

Time-dependent baselines: We also tried to replicate [2]’s use of time-dependent baselines, which is how they beat Cinematch with baselines alone. The first strategy they suggest is to bin time and assign each item bin-specific terms:

$$b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$$

These terms can then be determined via gradient descent. We tried bins of width 1 year and 6 months, but got RMSEs of 1.226 and 1.228, respectively (despite trying many λ and learning rate params).

We also tried including terms related to the day of the week and number of reviews given in the same day, and got 1.224 -- good, but not better.

METHODS

Data Selection:
The Yelp dataset includes data for 10 cities around the world. However, for the purposes of generating a strong recommendation system, it seems advisable to focus on a subset where there will be at least some overlap between businesses rated by users. Therefore we choose to subset to Phoenix, AZ, which had one of the largest datasets. We additionally choose to subset to only Restaurants and Bars, as users are likely to share preferences over a single category but may have different preferences when there are many categories being compared.

Network Based Analysis: Businesses can be viewed as a correlation network based on the level of user likeness. Two businesses are positively correlated when users gave both businesses more stars; and are negatively correlated when users rate a business as good but the other as bad.
Spearman's ranked correlation coefficients were calculated for the top-100 frequently reviewed businesses in Phoenix (**Figure 5**). We applied PageRank algorithm to infer the importance of a business in the network, and predict stars (RMSE=1.732).

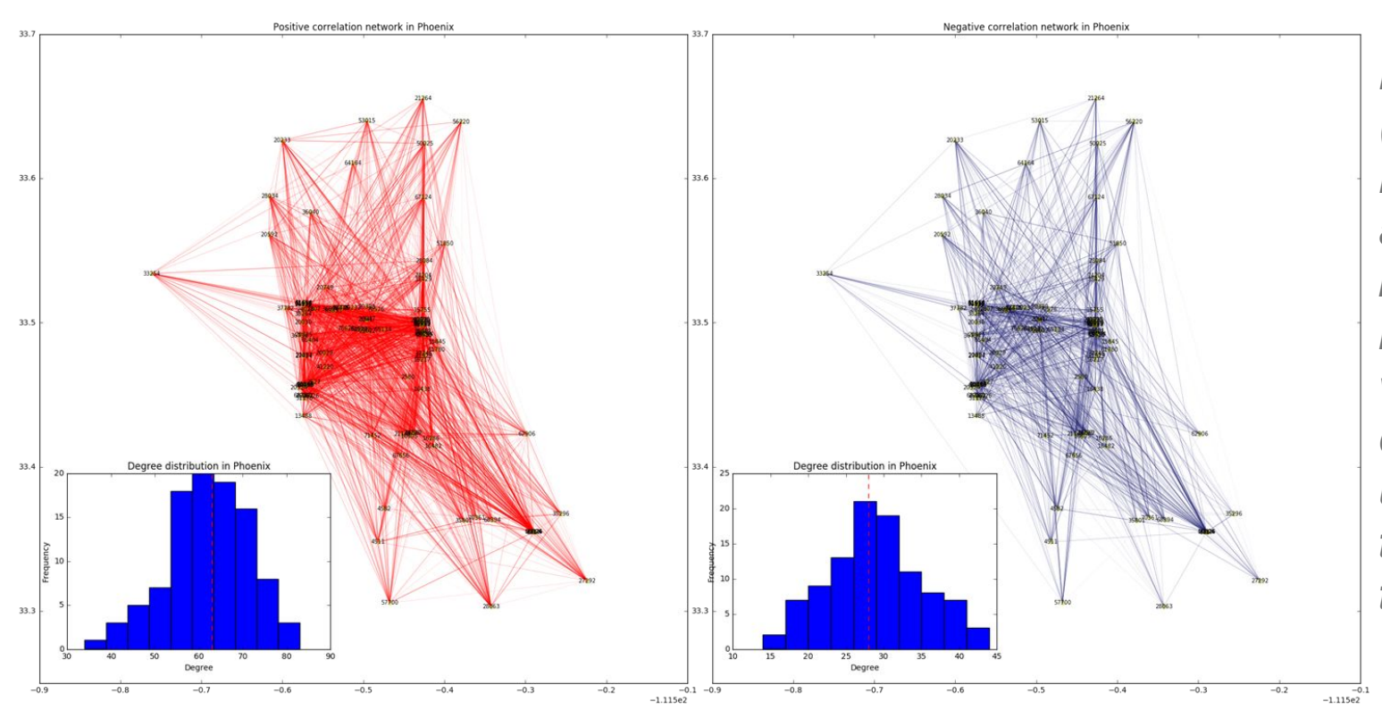


Figure 5: Positive (red) and negative (blue) business correlation networks in Phoenix. Nodes represent business IDs, and X and Y axes are their longitude and latitude. Degree indicates the total number of edges a node has. The median value was higher in the positive correlation network, suggesting when users like both two businesses, they tend to review more frequently than a business that is not liked.

Linear Regression Methods for Content Based Filtering:
We created matrices with columns for all of the attributes of users and restaurants and trained a lasso linear regression model on the attributes with stars as the target variable. In this way, we were able to determine the attributes that are most important in determining the number of stars that a user gives a restaurant. When running this model on new data, we achieved an RMSE of 1.3058 -- lower than the baselines, but interpretable.

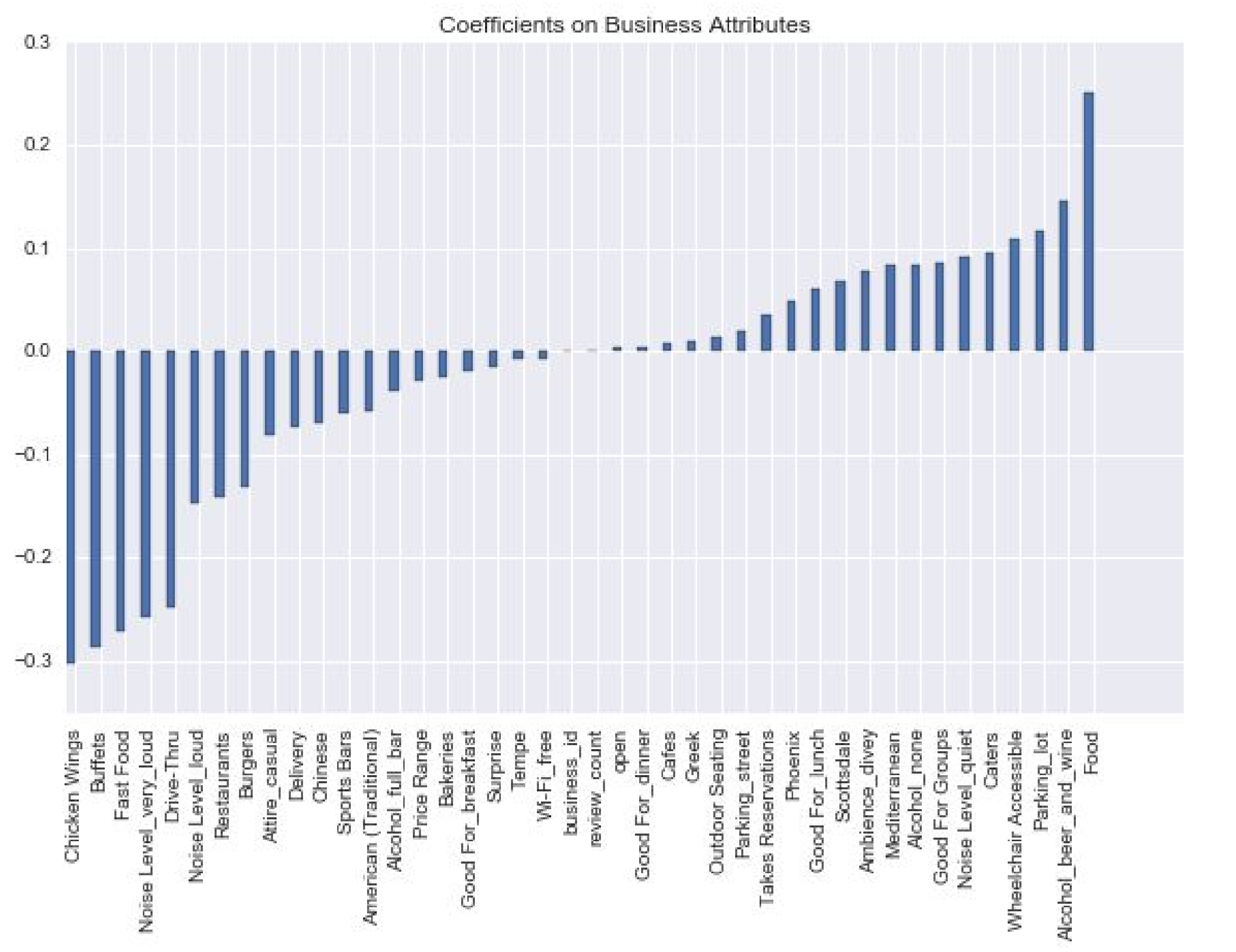


Figure 6. Coefficients of predictors in the linear regression model.

MATRIX FACTORIZATION

Latent Factor Models:
Many of the most successful models in the Netflix Challenge are latent factor models. In these models, vectors of factors describing users and items are inferred from the sparse matrix of users and item ratings using a singular value decomposition. We trained and tested these models over a variety of latent variable counts and regularization factors.

It turned out that our baseline predictors were actually better than three independent implementations of matrix factorization we tried, although when combining baselines with matrix factorization (i.e. using matrix factorization to predict residuals), we were able to do slightly better. When we limited our data to users with multiple reviews, all methods performed better and matrix factorization finally won out (by a hair).

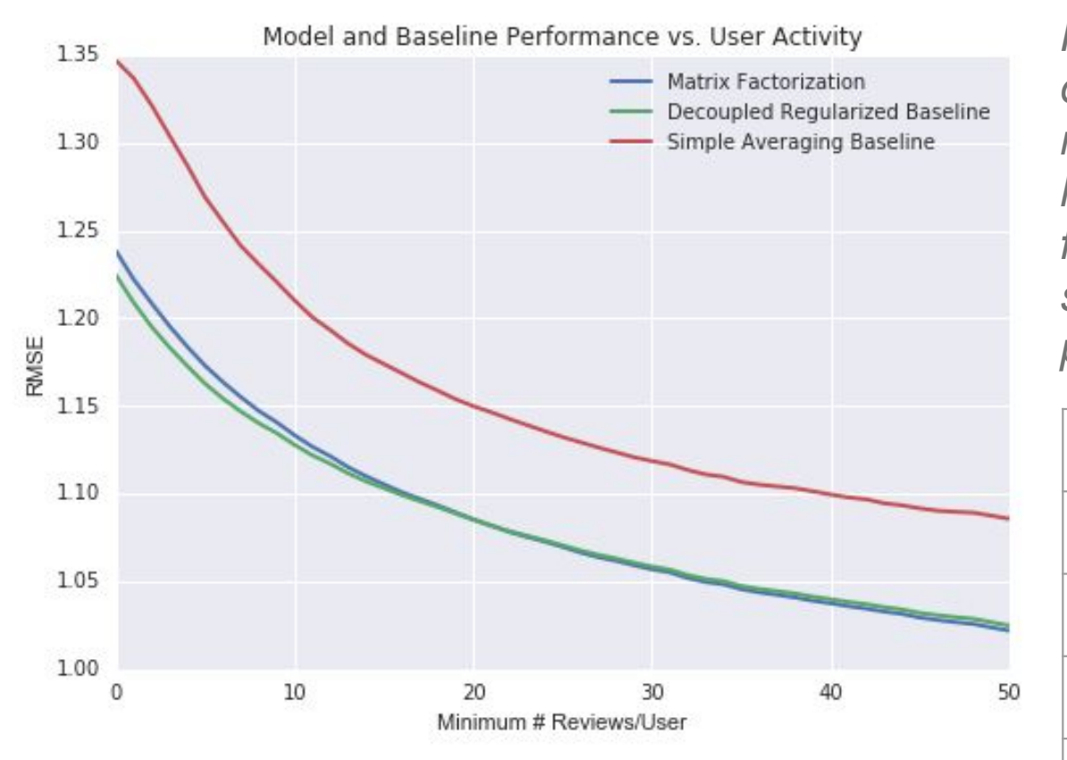


Figure 7: On the left, RMSE of two baseline methods and one MF method as we increased the minimum number of reviews for inclusion in the dataset. Because most users have only a single review, RMSE decreases significantly for all models when we add this limit. Below, RMSEs for several MF implementations on the full dataset (with some parameter tweaking). Including baselines helped a little bit.

MF Model Implementation	RMSE
Vowpal Wabbit [4] MF	1.2362
Surprise [5] SVD++	1.2344
GraphLab Create [3] MF	1.2522
GraphLab on Residuals	1.2243 (best)

ENSEMBLE METHODS

Random Forests:
The Netflix methods also heavily rely on training hundreds of smaller models on subsets of parameters and then using ensemble classifiers to predict the final number of stars. Using the content-based filtering matrix described earlier, we also trained a large Random Forest Classifier (n_estimators = 200) on subsets of the predictors after using cross validation to determine the optimal parameters for such a model. The resulting RMSE for this model operating on top of the Decoupled Regularized Average baseline was 1.247.

CONCLUSIONS

User Recommendation Models:
One of the major conclusions of this project is that the sparsity of the user-ratings matrix makes it very difficult to do better than simple popularity recommenders for the vast majority of users. In particular, it is nearly impossible to infer much about the users who give only a single rating, and as we see in the time analysis, this single rating may not even be indicative of typical rating behavior - in particular it is likely to be more extreme than usual, likely because a user who gives a single rating is only using Yelp because he or she has had a particularly good or bad experience.

Data Trends:
One of the more enjoyable aspects of this project was the data exploration. In particular, we were able to identify a number of interesting trends in the data sets that either confirmed or made us reconsider previously held notions about consumer opinion of restaurant quality. In the future, we would likely pursue these other aspects of the Yelp Data Challenge, like trying to predict the rise and fall of restaurant trends or whether a good or bad location is enough to cause a business to succeed or fail.

Managing Large Data Sets:
Throughout this project we encountered memory errors and long computation times. Although we found and implemented some workarounds, if we were to continue this work in the future, we would likely switch away from storing our data as in-memory numpy and pandas objects and investigate using databases or remote clusters instead.

[1] The Yelp Dataset Challenge, https://www.yelp.com/dataset_challenge
[2] The BellKor Solution to the Netflix Grand Prize, Koren 2009, http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
[3] GraphLab Create, https://turi.com/products/create
[4] Vowpal Wabbit, https://github.com/JohnLangford/vowpal_wabbit
[5] Surprise, http://surpriselib.com