CS 229 Final Project:
Divergent Recommendations for Yelp Users

Sigberto Alarcon Viesca, Christopher Heung, Shifan Mao

Abstract

The objective of this project is to make a recommendation for a Yelp user that is significantly different from what the user has tried in the past. We refer to such recommendations as “divergent recommendations”. For this goal, we leveraged machine learning methodologies and used various regression-based models and collaborative filtering to predict ratings and user-restaurant similarity. By combining these two predictions, we made "divergent recommendations" based on user and restaurant profiles. In addition, we implemented methods in solving constraint satisfaction problems (CSPs) to ensure the diversity of recommendations.

1 Introduction

Business review sites like Yelp have become commonplace in people’s daily decision-making. It is normal for potential customers to consult star ratings and other user reviews before deciding to eat at a restaurant. Yelp has helped users find new restaurants for foods that users already enjoy. However, there are few options for users that want to try seomthing unlike what they have tasted before.

In this work, we build a recommendation system can give restaurant suggestions that are highly different than the past places a user has tried. For example, given that someone like Domino’s Pizza and Pizza Hut, a simple recommendation system might recommend Patxi’s or Pizza my Heart. But one might be bored of pizza and might want to try something entirely different, like Ethiopian food. Our recommender determines the best divergent recommendation given a user’s history: one that is not only different, but that the user is also likely to enjoy.

2 Data

Our project utilized a small subset of the Yelp data set [1], which includes 1.6 million reviews from 366,000 users for 61,000 businesses. In order to create meaningful results, we only created recommendations for users who have at least 10 reviews. Additionally, we only looked at businesses that have at least 10 reviews. The publicly available data is formatted in JSON. An example of a review object from the data set is given below:

```
review
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating to half-star),
    'text': (review text),
    'date': (date, formatted like '2012-03-14')
}
```

3 Relevant past work

Current recommendation systems typically produce a list of recommended items based on two approaches: content-based filtering and collaborative-based filtering [3]. In particular, collaborative-based filtering is based on collective rating history of many users. It then predicts whether a user will like a particular item based on their similarity to other users [4,6-10]. Such recommendation system has been popular in industry, such as the item-to-item collaborative-based filtering used by Amazon and Last.fm [5]. However, one major assumption of such recommendation is that user’s future behavior will be consistent with his previous preferences. In the scope of our work, a "divergent recommendation" system will recommend businesses that are significantly different from users’ past visits. Therefore, when considering similarity between users, the difference between businesses also becomes an important factor for generating recommendations. By recommending a very different business that a similar user rated highly, the recommender allows the user to explore new experiences with a minimized risk of not enjoying them. This feature is the major distinction between this work and previous recommenders.

4 Problem Definition

4.1 Input

We are given a user u who we will make recommendations for, and we have all reviews that user u has written in the past. For a given restaurant i, the Yelp data set has metadata about i (including past reviews, location, categories, neighborhood, price range, opening hours) and we can find all reviews for i. i.

4.2 Problem Components

We want to recommend restaurants that are divergent, but that the user will still enjoy. We decided to break this problem into two parts so as to make evaluation more tractable by evaluating each model separately. We then combine the subproblems to provide an answer to the main one. The two components of the recommendation system include:

1. Rating Prediction
2. Similarity

A visual representation of the problem breakdown is given by the following flowchart:

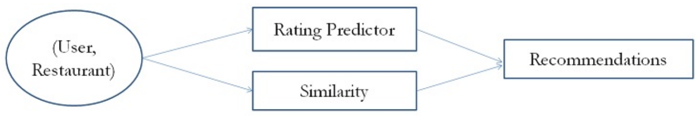

Figure 1: *Recommendation Flowchart*

4.3 Rating Prediction

In this subproblem, we want to ensure that the user will enjoy the divergent recommendation. Our goal here is to predict the number of stars that the user u will give to a restaurant i. We evaluate a prediction against the actual rating that a user gave a business.

For our baseline, we calculate the mean review score the user u has given in the past and predict that score for any restaurant. This is a lower bound approximation since it does not customize to the restaurant itself. For the oracle, we use the training error of our best model rather than the test error to approximate an upper bound.

4.4 Similarity Metric

In this part, we measure the similarity between the user u's past reviews and restaurant i. Intuitively, a person whose history is entirely composed of Italian and Mexican restaurants will be divergent and dissimilar from a Thai establishment.

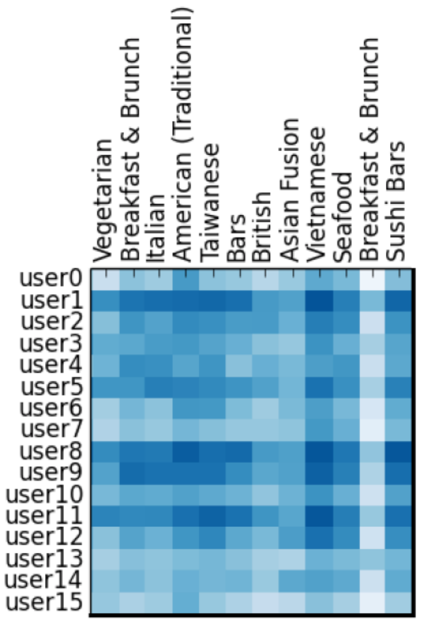

Figure 2: *User-restaurant similarity matrix. The darker squares represent higher similarity.*

One of the most significant challenges of our recommendation system is modeling a similarity metric because there is no single figure in the data set that represents it directly. Hence, we resort to heuristic methods to approximate similarity. As a label for supervised learning, we leveraged the fact that the Yelp data set contains categories for restaurants (e.g. "American (New)", "French", "Pubs") to define

$$\text{sim}(u,i) = \sum_{c\in(C_i\cap C_u)}^{k} w(c_k,u) \tag{1}$$

$$w(c_k,u) = \frac{1}{m}\sum_{l\in U_i}^{m} 1[(c_k = c_l)] \tag{2}$$

Intuitively, this can be thought of as a weighted Jaccard similarity. These equations define a metric $0 \le \text{sim}(u,i) \le 1$ that sums over the overlap in the categories of a business C_i and the categories of all businesses that a user has been to in the past C_u and weighs them by the percentage of that category in the user's reviewed businesses U_i. Even though this similarity is inherently limited due to the fact that it is untestable with our data set, it is a useful proxy because it is positively correlated with whether a user has been to a business with the same category and how many times a user has been to that category. Notice that $\text{sim}(u,i) = 1$ iff $C_i \cap C_u = C_i \cup C_u$ and $\text{sim}(u,i) = 0$ iff $C_i \cap C_u = \emptyset$.

However, this similarity metric is only available in a rich data set, like Yelp's. We want our model to be applicable to simpler data sets, so we used this similarity as the label for training various machine learning algorithms. To maximize applicability to predict the similarity metric in Eqn. (1), we included the mean and standard deviation of users' star ratings. To capture the overlap between users' and restaurants' profiles we also included the compiled review texts for the users and restaurants. We then use these features to predict the similarity metric. In Figure 2, we show some sample users and categories.

4.5 Output

We combine our rating predictor and similarity metric to obtain an overall evaluation of the divergent recommendations. We define a divergent constant α where $0 \le \alpha \le 1$ that represents the tradeoff between exploration and exploitation. Such parameter can be a user-defined parameter allowing the user's preference between exploration and exploitation of restaurant choices. Specifically, lower α value favors exploitation and a higher α favors exploration. We define the composite score as:

$$\text{score}_{\text{final}} = \frac{\alpha}{5}\cdot \text{rat}(u,i) + (1-\alpha)\text{sim}(u,i) \tag{3}$$

where rat and sim represent rating prediction and similarity, respectively. Note that we normalize our predicted rating score by its maximum value (rat=5) such that it will range between 0 and 1. With this formula, we can compare the user u against all restaurants i and output the top scoring restaurants as recommendations for a user's particular choice of α.

5 Early Model

5.1 Feature Selection

In order to run parametric models to learn a good recommendation, we first defined what features to extract and created a feature vector for each user-restaurant pair. From each user and restaurant, their past text consisted of all the reviews each has written or received, respectively. For each review, we used the bag-of-words approach and found the tf-idf score. We hypothesized that the words people used in the review for a particular restaurant would be highly reflective of that business.

We also looked at some metadata provided by Yelp. Each restaurant has a set of categories (e.g. Italian, American) and a set of attributes (e.g. Price Range, Good for Kids). We can additionally calculate the mean and standard deviation of ratings given for a restaurant and for a given user's history.

We created 3 feature sets to test on that consisted of permutations of the above features:

1. Only tf-idf
2. Only metadata
3. Both tf-idf and metadata

5.2 Rating Prediction

Using the above feature sets, we formulated rating prediction as a supervised regression problem. Specifically, given the reviews texts and restaurant metadata, we aimed to reproduce user's star rating of particular restaurant, i.e. rating prediction is continuous. We first applied 3 different parametric models using the SciKit-Learn Python library:

1. Linear Regression
2. Support Vector Machines (SVM)
3. Random Forest

In particular, Random Forest regressor is optimized by selecting appropriate number of trees ($n = 7$) based on the the choice that minimizes test error. In addition, regression using Support Vector Machines uses kernel of cubic order polynomial. We choose 5-fold cross validation to evaluate the performance of the above models on rating predictions based on the root-mean-squared errors (RMSE) of the test (validation) set as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (p_{u,i} - r_{u,i})^2}{n}} \tag{4}$$

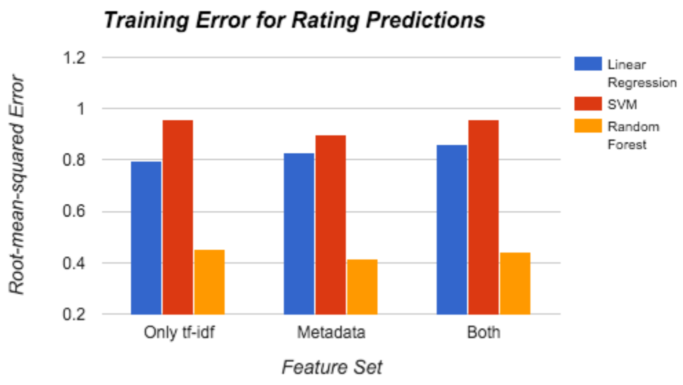

Figure 3: *Training Errors of Early Models*

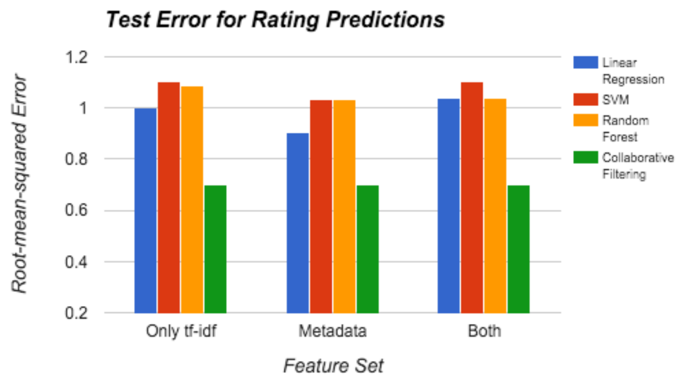

Figure 4: *Test Errors of Early Models*

In addition to these 3 parametric models, we also applied user-user collaborative filtering. The idea behind collaborative filtering is that if two users both like the same restaurants, then a user is likely to enjoy the next place that the other user likes. Formally, we estimate user u's rating of restaurant i as:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|} \tag{5}$$

where $p_{u,i}$ represents the predicted star rating, $\bar{r}_u$ represents the average review score for a user u and u' represents a similar user, $s(u,u')$ is the similarity between user u and u' (discussed below), and N is a set of "similar" neighbors. This formulation accounts for both user bias and similar users' biases of a particular restaurant i.

Figure 3 and Figure 4 shows a comparison of root-mean-squared errors of training and test data sets for different regression models. Note that collaborative filtering gives lowest test error among all models.

5.3 Similarity Metrics

We measure the distance between any restaurant and any user by taking the cosine similarity of the feature vector computed for each. If ϕ_u represents the tf-idf of user u and ϕ_i represents the tf-idf of user i,

$$\cos(u,i) = \frac{\phi_u \cdot \phi_i}{||\phi_u||\ ||\phi_i||} \tag{6}$$

Note cosine similarity varies from 0 and 1 since tf-idf feature vectors have non-negative entries. Using the combined feature set including review texts and restaurant metadata, we computed the cosine similarity and compared to our category similarity heuristic from Eqn. (1). The RMSE for this calculation was 0.210.

5.4 Result Interpretation

With increasing number of trees in RF regression, Figure 5 shows training and test error decreases initially. We point out that there exists a large gap between training and test error. This observations indicates that our regressions models are over-fitting the data due to excess number of features.

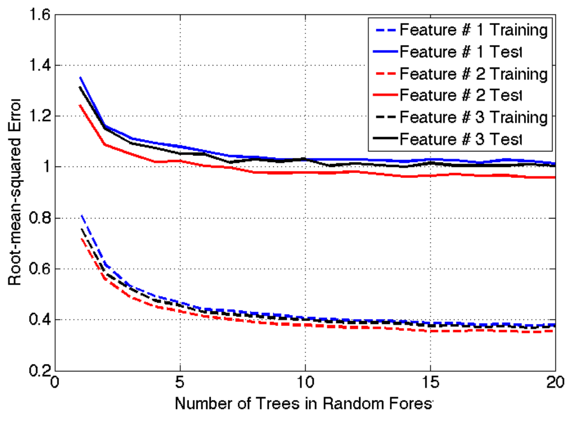

Figure 5: *Training and Test Errors of Random Forest Regression*

We concluded that the excess number of features introduced a high noise-to-signal ratio. In particular, we observed that the bag-of-words model cannot capture the different meanings and context of a qualifier like "very". This has the potential to add significant noise to our model. This notion was supported by the fact that our lowest RMSE was obtained from the feature set without tf-idf features.

6 Improved Model

6.1 Feature Selection

Instead of using tf-idf, we used the GenSim Python library, which implements Google's word2vec, which is fine-tuned to learn the relationship between words given their context. We removed the stop words from our reviews, which decreases the noise of irrelevant words. We also removed noisy parts of the metadata such as the restaurant attributes that are sparse (e.g. "Good for brunch", "Free Wi-Fi").

Our modified feature sets were:

1. Only doc2vec
2. Only metadata (minus attributes)
3. Both doc2vec and metadata

6.2 Rating Prediction

Once more, used linear regression, support vector machines and random forest regression models to predict users' ratings. To reduce noise from useless features, we also applied a neural network model that should theoretically filter out unhelpful features. The layers of neural network model include a layer of 100 rectifiers and another layer of linear neurons. Using the new feature sets above, we obtained the following results:

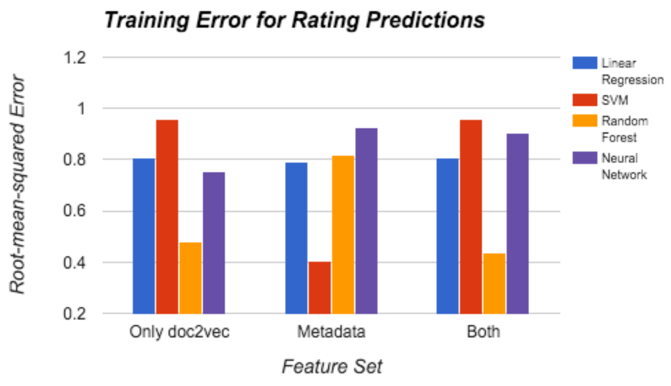

Figure 6: *Training Errors of New Models*

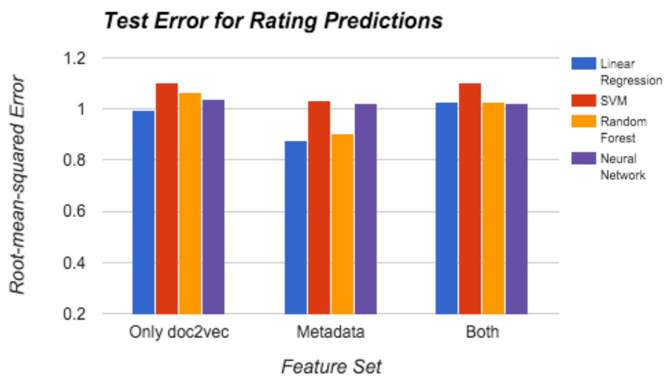

Figure 7: *Test Errors of New Models*

6.3 Similarity Metrics

Because cosine similarity from our early model was trained with the high noise-to-signal tf-idf bag of words feature vector it generated a rather high RMSE of 0.210. It also had the disadvantage of being untrainable. Instead, we decided to use various machine learning models with a feature vector composed of the word2vec vectorized texts from each user-restaurant pair to predict our similarity heuristic: from Eqn. (1):

We apply Linear Regression, SVM, Random Forest, and Neural Networks to learn the similarity metric based on the same feature sets. The below table summarizes the cross-validation root-mean-squared errors of different models. We also show the true and predicted values of similarity metric based on Random Forest of the training and test sets in Figure 8 and Figure 9.

	Model Selection			
	LR	SVM	RF	NN
sim Train. Err	0.011	0.025	0.005	0.020
sim Test Err	0.013	0.026	0.011	0.022

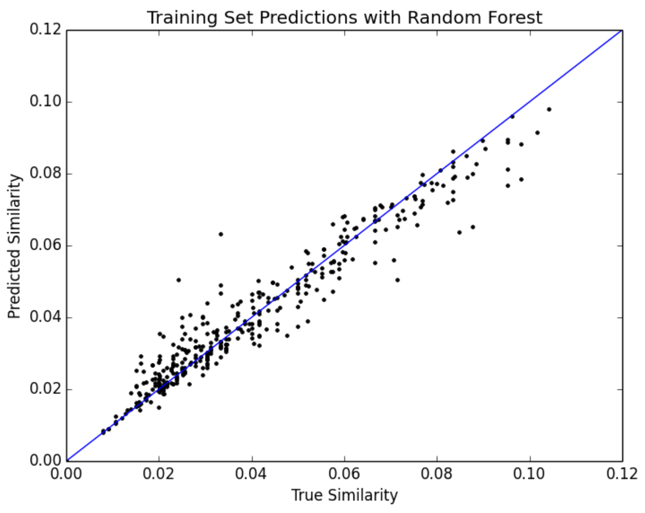

Figure 8: *Training of Random Forest on similarity*

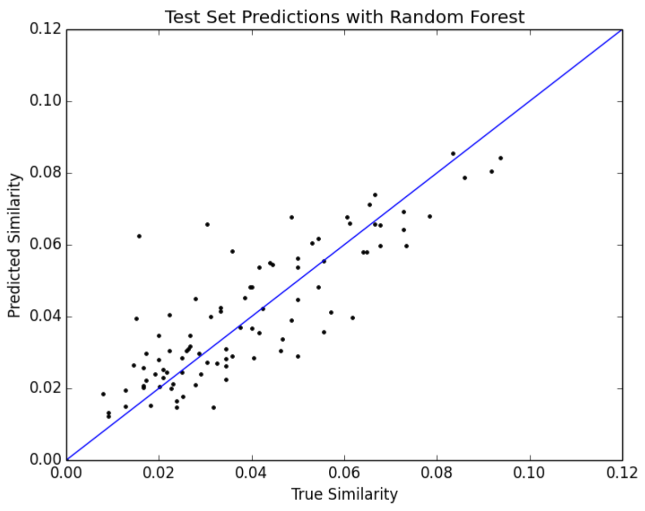

Figure 9: *Testing of Random Forest on similarity*

6.4 Result Interpretation

The disparity between the neural network training and test error is much lower than the other models, so we have less overfitting. The overall RMSE for linear regression, SVM, and Random Forest was lower for the feature sets that used metadata, since we removed large parts of the noise. However, the RMSE for only doc2vec was similar to the early model's RMSE for only tf-idf and this leads us to conclude that review texts are just so noisy and give little useful information about a restaurant similarity with a user as compared to metadata.

7 Overall Evaluation

We have shown the evaluation for individual parts of the model (rating predictor and similarity metric), whose score was given in section 4.5:

$$\text{score}_{\text{final}} = \frac{\alpha}{5} \cdot \text{rat}(u, i) + (1-\alpha)\text{sim}(u, i) \tag{7}$$

Remember that we divide rating predictor by 5 since it is in range of 1 to 5 while similarity is in range of 0 to 1. However, we are not able to evaluate the composite model accurately because we do not have access to the users in the data set and there is no label that we can use from it. With access to the users, we could query them for a score for the overall value of the divergent recommendation (taking into account both the similarity to a given restaurant and how accurate the predicted rating is). In this instance we could either learn α for each user or ask them to consider a particular value of α when they evaluate the divergent recommendation. Unfortunately, the cities that form part of the data set and the businesses therein are unknown to us and we could not give meaningful evaluation to the model.

Nevertheless, we approximate the overall evaluation RMSE by summing the RMSE error for the rating predictor divided by 5 (to normalize it as with the above equation) and the similarity metric.

Taking the best errors of both models we get:

```
Early Model (LR) = 0.9036/5 + 0.2102 = 0.3909
Improved Model (LR) = 0.8798/5 + 0.013 = 0.1890
```

As we can see, the improved model has a lower RMSE and thus is more accurate than the early model, which is to be expected considering the improvements added.

8 Future Improvements

We have iterated on different ways to do rating prediction and the similarity metric. Some additional methods that we could try applying is:

1. Using other similarity metrics such as Pearson Correlation and KNN.

2. Predicting ratings as a classification problem rather than an unsupervised learning problem. Here we can think of each review as having 5 classifications, one for each possible review score. We could then apply Naive Bayes to solve this problem.

3. Content-based filtering is another promising improvement. This is similar to collaborative filtering, except that it deals with user-to-item similarity rather than just user-to-user similarity.

References

1. http://www.yelp.com/dataset_challenge

2. http://grouplens.org/site-content/uploads/Item-Based-WWW-2001.pdf

3. Jafarkarimi, Hosein, Alex Tze Hiang Sim, and Robab Saadatdoost. "A naive recommendation model for large databases." International Journal of Information and Education Technology 2.3 (2012): 216-219.

4. Koren, Yehuda. "Collaborative filtering with temporal dynamics."Communications of the ACM 53.4 (2010):

5. Sarwar, Badrul, et al. Application of dimensionality reduction in recommender system-a case study. No. TR-00-043. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

6. G. Karypis, "Evaluation of item-based top-N recommendation algorithms," in ACM CIKM '01, pp. 247–254, ACM, 2001.

7. B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in ACM EC '00, pp. 158–167, ACM, ACM ID: 352887, 2000.

8. I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. OfekKoifman, "Personalized recommendation of social software items based on social relations," in ACM RecSys '09, pp. 53–60, ACM, 2009.

9. R. Burke, "Hybrid recommender systems: Survey and experiments," User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331–370, November 2002.

10. N. Lathia, "Evaluating Collaborative Filtering Over Time," PhD thesis, University College London, London, UK, June, 2010.