Recommendation System Leveraging Heterogeneity of Trust in Social Networks

Pulkit Agrawal, Jayanth Ramesh and Agrim Gupta

I. INTRODUCTION

Recommendation systems are an integral part of many online services ranging from content streaming websites like Netflix to online shopping websites like Amazon. They improve the online experience by suggesting new products that match users' interests and preferences. Collaborative filtering [4] methods are the most commonly used techniques in recommendation systems which rely on collection and analysis of users' preferences and predicting what users will like based on their similarity to other users. Traditional collaborative filtering based recommendation systems can be categorized into three types: user based, item based and hybrid methods. Many algorithms have been used for measuring user and item similarity in recommendation systems like the k-nearest neighbor (k-NN) approach and the Pearson Correlation as first implemented in [5]. Neighborhood methods are good at detecting very localized relationships but poor at detecting a user's overall user preference. In contrast, latent factor models are best at estimating the overall preferences of a user but poor at incorporating relationships between users. Due to this complementary nature of the two approaches, [3] integrates them to provide better recommendations.

Trust is another aspect of recommendations in real life scenarios where we trust different friends for recommendations in different categories of products. For example, we may prefer movie recommendations from an entirely different set of friends as compared to electronics recommendations. We refer to this difference in trust for different categories of recommendations as *heterogeneous trust*. [7] attempts to improve on traditional methods by studying the relations between trust and product ratings in online consumer review sites. They propose an improvised matrix factorisation technique that can be used to improve rating predictions and estimate true strengths of trust relations at the same time. TrustWalker [8] proposes a random walk model which combines trust-based and item-based recommendations. This improves the performance of recommendations even for cold start users. Our goal in this project is to determine if we can leverage heterogeneous trust to improve recommendations. Our intuition is that what is true in real life should also hold true for online communities. More specifically, given a set of users, their ratings for products in several categories and a social network of users, we will predict ratings for products that have not been reviewed by a particular user. We will then evaluate whether modeling heterogeneous trust information improves the prediction accuracy.

II. DATA

We work with two different datasets to evaluate our models. Our first dataset comes from the Yelp Dataset Challenge [12] which contains business data (business id, location, average rating, category, attributes, etc.), user data (user id, friends, number of reviews so far, etc.) and review data (user id, business id, review text, star rating on an integer scale of $1-5$, etc.) for six cities in the US. The dataset is available in JSON format. We imported the data in a Postgres database which is used as the main datastore for our implementation. We fixed our product categories to restaurants and shopping as they had the maximum number or reviews amongst all categories. We narrowed down to top 6000 users based on the total number of reviews they had in the above categories. We constructed a social network of users using friends information in the user dataset and only kept those users in the network who had reviewed either restaurants or shopping businesses and who were in the list of top 6000 users. After the data wrangling, our final dataset had 5614 users, 27688 businesses (7598 shopping, 20090 restaurants) with 89 attributes, 247551 reviews and 45634 edges (friendships) in the user social network.

Yelp business dataset has a list of attributes for each business which captures information about its characteristics. For example, the most common attributes for restaurants are the ambiance of the restaurant (classy, touristy, romantic, etc.), the type of music played (DJ, Karaoke, jukebox, etc.) and so on. We use these attributes to extract categorical features for the businesses. That is, for the ambiance attribute, 'ambiance classy'is a feature which takes value 1 if the business has classy ambiance and 0 otherwise. We define 89 such features for the restaurants.

Our second dataset is Epinions dataset [6] which contains user and ratings data for products in several categories. The data is available in form of MATLAB .mat files. We select movies and electronics as our two product categories because they have the maximum number of reviews amongst all categories. We select only those users who have reviewed products in both the above categories. In the user dataset, we have explicit trust relations for each user, that is, we explicitly know which users are trusted by a given user for recommendations. We use this data to generate a directed graph where users are nodes and a directed edge exists between user u and user v if u trusts v. This directed graph will be referred to as trust network in subsequent discussion. Our final dataset has 6874 users, 14412 products (4307 electronics, 10105 movies), 74246 reviews and 43733 edges

in the trust network (917 zero in-degree nodes, 641 zero out-degree nodes maximum degree - 314).

The rating distribution for both the datasets is shown in Figure I. It can be seen that the data is highly skewed towards positive ratings (4 and 5).

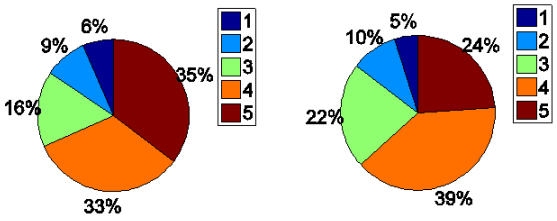

Fig. 1. Rating Distribution of Yelp(left) and Epinions(right)

III. Methods

We propose two different approaches to model heterogeneous trust information in recommendation systems. In our first approach, we use k-means clustering to find communities of users with similar preferences in each product category. Thus, for each product category, we will have a different clustering of users into communities. A new rating is then predicted by a collaborative filtering algorithm which computes a weighted average of ratings of all the users for a particular product in a given user's community.

In order to cluster the users based on user preferences, we generate user profiles based on attributes of the businesses reviewed by them as follows. Let $\{b_1, b_2, \ldots, b_m\}$ be the m businesses reviewed by the user u with ratings $\{r_1, r_2, \ldots, r_m\}$. Let $\{f_1, f_2, \ldots, fm\}$ be the set of 89×1 binary feature vectors corresponding to the businesses. The user profile of user u is the 89×1 vector $\frac{\sum_j f_j r_j}{m}$. Intuitively, if a user gives high ratings to restaurants with classy ambiance and jukebox music, these features would have higher weights in the user profile vector, thus indicating the preference of the user to such restaurants.

We also try to model user preferences by learning a Naive Bayes and SVM model for each user where the attributes of businesses reviewed by the user are used as features and a positive label (1) is used if star rating of the business $>= threshold$ and a negative label (0) is used otherwise. The parameters of the learnt model would serve as user profiles in this scenario which we intended to use for clustering in the community detection stage. We implemented Naive Bayes in MATLAB while scikit learn [1] was used for SVM and k-means. The algorithms were implemented with the defaults from scikit, which can be found in their user guide [2]. We tested our first approach only on Yelp dataset.

In our second approach, we estimate the strength of trust a user places in all the users in his trust network for each product category using category specific and general trust models described below. Thus, for each product category, we will have a different value of trust for all the users in a given user's trust network. We then use a collaborative filtering algorithm to predict the unknown ratings (for trust models A, B and E) as follows:

$$\hat{\mathbf{R}}_{u,b} = \mu_u + \frac{\sum_v \mathbf{S}_{u,v}(\mathbf{R}_{v,b} - \mu_v)}{\sum_v \mathbf{S}_{u,v}} \quad (1)$$

In the equation above and all the models that follow, $\mathbf{R}_{u,b}$ and $\hat{\mathbf{R}}_{u,b}$ are the actual and estimated (from data) ratings of user u to product b, μ_u is the average rating of user u across all products and $\mathbf{S}_{u,v}$ is the strength of trust between users u and v computed using one of the models described below. For models which measure the strength of trust between users for each category separately, we define a category specific trust strength matrix $\mathbf{S}^l_{u,v}$, where l denotes the product category. We also define a trust matrix $\mathbf{T}_{u,v}$ for our network of users which takes value 1 if user u trusts v and 0 otherwise. For all the trust models we consider, we define $\mathbf{S}_{u,v}$ and $\mathbf{S}^l_{u,v}$ to be 0 if $\mathbf{T}_{u,v} = 0$. In addition to these matrices, we define a product category matrix $\mathbf{C}_{b,l}$ which takes value 1 if product b belongs to category l and 0 otherwise. We refer to our second approach as *trust modeling* in the discussion that follows. Below, we describe the various trust models we experimented with.

A. Cosine Trust

This is our simplest model for estimating the strength of trust between two users. We assume that two users trust each other in a particular category if they rate same products in that category similarly. For each category l, we define strength of trust between u and v as follows:

$$\mathbf{S}^l_{u,v} = \frac{\sum_b \mathbf{R}_{u,b}\mathbf{R}_{v,b}\mathbf{C}_{b,l}}{\sum_b \mathbf{R}_{u,b}\mathbf{C}_{b,l} \sum_b \mathbf{R}_{v,b}\mathbf{C}_{b,l}} \quad (2)$$

B. Recommendation Power

This measure of trust is inspired from [11]. For each category l, the strength of trust between users u and v is defined as:

$$\mathbf{S}^l_{u,v} = \sum_b \frac{\mathbf{R}_{u,b}\mathbf{R}_{v,b}\mathbf{C}_{b,l}}{\mathbf{R}_u\mathbf{R}_b\mathbf{C}_{b,l}} \quad (3)$$

where $\mathbf{R}_u$ is sum of all ratings by user u and $\mathbf{R}_b$ is sum of all ratings given to product b. We can interpret this trust measure intuitively in terms of a two step random walk on a weighted bipartite graph of the set of all users and the set of all products of a particular category, where an edge exists from a user to a product if the user rated the product. The weight of the edge is the rating given by the user to the product. Under these definitions, $\frac{\mathbf{R}_{u,b}}{\mathbf{R}_u}$ is the probability of traveling from u to b and $\frac{\mathbf{R}_{v,b}}{\mathbf{R}_b}$ is the probability to traveling from b to v in a random walk on the graph. The recommendation power is thus the total probability of traveling from u to v. The probability is high if u rates all the businesses which are most liked by v (a high value of $\frac{\mathbf{R}_{v,b}}{\mathbf{R}_b}$) highly (a high value of $\frac{\mathbf{R}_{u,b}}{\mathbf{R}_u}$).

C. Latent Factor+General Trust Hybrid

This model is inspired from [7]. In this model, we assume that a user's rating for a particular product depends partly on his inherent preferences and partly on the ratings of users in his trust network. The user's preferences are modeled using a latent factor model where we assume that users rate products using K hidden features. For each product b, a $K \times 1$ vector $\mathbf{q}_b$ denotes the values of these K features for the product and for each user u, a $1 \times K$ vector $\mathbf{p}_u$ denotes the importance the user places in these K factors and u's estimated rating on b is given by $\mathbf{p}_u.\mathbf{q}_b$. In order to model the influence of the trusted users, we add a second component to this rating estimate by averaging the ratings of all the users in a given user's trust network using the strength of the trust between the users as a weight. Thus, the estimated rating, $\hat{\mathbf{R}}_{u,b}$ of a user u for a product b is given by:

$$\hat{\mathbf{R}}_{u,b} = \alpha \mathbf{p}_u.\mathbf{q}_b + (1-\alpha)\frac{\sum_v \mathbf{R}_{v,b}\mathbf{S}_{u,v}}{\sum_v \mathbf{S}_{u,v}} \tag{4}$$

We constrain $\mathbf{S}_{u,v}$ to lie between 0 and 1, larger value indicating more trust. α is a hyper-parameter which determines the importance of each component. In order to learn the parameters of this model, we solve the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{P},\mathbf{Q},\mathbf{S}} \sum_{(u,b)\in O} (\mathbf{R}_{u,b} - \hat{\mathbf{R}}_{u,b})^2 \\ &+ \frac{\beta}{2}(\sum_u \sum_k p_{u,k}^2 + \sum_b \sum_k q_{b,k}^2) \\ &\textit{s.t. } \forall u, v \ \ \mathbf{S}_{u,v} \in [0,1] \end{aligned} \tag{5}$$

where O is the set of all user-product pairs for which we know the ratings, $\mathbf{P}$ is the matrix formed by stacking the $\mathbf{p}_u$ vectors row-wise and $\mathbf{Q}$ is the matrix formed by stacking the $\mathbf{q}_b$ vectors column-wise. Here β is a regularization parameter which doesn't allow $\mathbf{p}_u$ and $\mathbf{q}_b$ to become too large and avoids over-fitting. We initialize the parameters to random values and solve the optimization problem by using stochastic gradient descent with the following update rules,

$$\begin{aligned} p_{u,k} &\leftarrow p_{u,k} + \gamma(\alpha \mathbf{E}_{u,b} q_{b,k} - \beta p_{u,k}) \\ q_{b,k} &\leftarrow q_{b,k} + \gamma(\alpha \mathbf{E}_{u,b} p_{u,k} - \beta q_{b,k}) \\ \mathbf{S}_{u,v} &\leftarrow \mathbf{S}_{u,v} + \gamma((1-\alpha) \\ &\quad \mathbf{E}_{u,b}\frac{\mathbf{R}_{v,b}\sum_v \mathbf{S}_{u,v} - \sum_v \mathbf{R}_{v,b}\mathbf{S}_{u,v}}{(\sum_v \mathbf{S}_{u,v})^2}) \end{aligned} \tag{6}$$

γ is the learning rate and $\mathbf{E}_{u,b}$ is the estimation error $(\mathbf{R}_{u,b} - \hat{\mathbf{R}}_{\mathbf{u},\mathbf{b}})$. If the updated value of $\mathbf{S}_{u,v}$ exceeds 1 or falls below 0, we set it to 1 and 0 respectively.

D. Latent Factor+Heterogeneous Trust Hybrid

Apart from some minor implementation differences, this model is similar to the model proposed in [7]. It is same as the previous model with the difference that instead of learning a common trust strength $\mathbf{S}_{u,v}$ between users u and v for all products, we learn a different trust for each category of products we have in our data set. Thus, the estimated rating in (4) changes to the following:

$$\hat{\mathbf{R}}_{u,b} = \alpha \mathbf{p}_u.\mathbf{q}_b + (1-\alpha)\frac{\sum_l \sum_v \mathbf{C}_{b,l}\mathbf{R}_{v,b}\mathbf{S}^l_{u,v}}{\sum_l \sum_v \mathbf{C}_{b,l}\mathbf{S}^l_{u,v}} \tag{7}$$

Using this model we are able to estimate strength of trust between users for each category separately thus allowing us to estimate heterogeneous trust. We solve the same optimization problem as in (5) replacing $\hat{\mathbf{R}}$ with its value as defined in (7). Consequently, the update equations for $\mathbf{S}^l_{u,v}$ also change as follows:

$$\begin{aligned} \mathbf{S}^l_{u,v} &\leftarrow \mathbf{S}^l_{u,v} + \gamma((1-\alpha)\mathbf{E}_{u,b} \\ &\quad \frac{\mathbf{C}_{b,l}\mathbf{R}_{v,b}\sum_v \mathbf{S}^l_{u,v} - \sum_v \mathbf{C}_{b,l}\mathbf{R}_{v,b}\mathbf{S}_{u,v}}{(\sum_v \mathbf{C}_{b,l}\mathbf{S}^l_{u,v})^2}) \end{aligned} \tag{8}$$

To reduce computation time, we implemented both our hybrid latent factor models in C++. All the other collaborative filtering algorithms were implemented in MATLAB.

E. Global Trust

All the trust models we discussed so far are local trust models in the sense that they measure how much a particular user is trusted by another user, thus representing a local relationship between the two users. However, we can also have a global trust model where we assign a trust value $\mathbf{S}_u$ to each user u which measures the overall trustworthiness or importance of the user u solely based on the structure of the trust network. PageRank [10] is an ideal choice for this global trust model because in essence it measures the importance of each node in a network based on how many important nodes (in our case trustworthy users) are linked to it. We compute the PageRank of all the users (nodes) in our trust network and assign it as the global trust value $\mathbf{S}_u$. We contrast global trust measure with the local trust measures to understand how the opinion (or trust) of locally trusted users compares to the opinion of 'experts'when it comes to predicting ratings. We use Snap.py [9] for all our network analysis including PageRank computation.

IV. Experiments/Discussion

A. K-means

For our first approach, we ran k-means algorithm for 5614 users with user profiles generated using the 89 business attributes for restaurants. Determining the number of clusters in a data set is a common problem in data clustering. We determined the number of clusters using Silhouette Coefficients. Shown in Table I are the top three features for 3 sample clusters for $k = 11$. As we can observe, it's difficult to gain intuitive insight into the preferences of the users clustered together based on these basic features. From this table, we inferred that it is not possible to detect communities of users with similar preferences using k-means algorithm. Further analysis of business and their attributes revealed that many business types were combinations of other popular groups; for example the health group contains every business which is categorized as spa and health. On the other hand,

TABLE I

K-MEANS : TOP FEATURES FOR SAMPLE CLUSTERS

One	Two	Three
Price Range_2	Good For_dinner	Good for Kids
Good For Groups	Takes Reservations	Take-out
Take out	Waiter Service	Good for Groups

because of crossovers such as a food court residing within a shopping mall, the business attributes were often overlapping and failed to capture the inherent differences in the factors which affect ratings for these very different business types. Hence, we decided to abandon this approach to model heterogeneous trust.

B. Naive Bayes and SVM

Our results for learning user profiles using Naive Bayes and SVM are shown in Table II. The values shown are average test errors across all users. Threshold in the table refers to the rating above which all ratings are given a positive label. In spite of our efforts to diversify the above two algorithms, (changing the threshold or feature engineering) the results of the algorithms were still very close to a baseline algorithm which predicts an unknown rating as average rating of the user. This can be attributed to the inherent skew in the rating distribution as shown in Figure (1). We can observe that almost 68% of the ratings are either 4 or 5. Also, we have very few bad ratings making it difficult to predict bad ratings. Since we could not obtain a representative model of user profiles using either Naive Bayes and SVM we decided to move on to our second approach.

C. Trust Modeling

For all the trust models, we split our dataset into 70% data for training and 30% data for testing. We chose to evaluate our trust models through the root mean square error (RMSE) between the predicted ratings and the actual ratings to measure accuracy. The RMSE error is computed as follows:

$$RMSE = \sqrt{\frac{\sum_i^n (p_i - a_i)^2}{n}} \tag{9}$$

where p_i is the predicted rating, a_i is the actual rating and n is the number of predicted ratings. To evaluate the effect of modeling category specific trust (referred to as *heterogeneous trust* in the results) on prediction accuracy, we compute cosine trust and recommendation power for all the categories considered together, that is, we assume that all the products belong to the same category and compute the trust measures as defined above. These category independent trust models are referred to as *general trust* models. For the general trust models, we compute the within category RMSE values along with the overall RMSE values. That is, after using the general trust model to predict ratings, we compute the RMSE error separately for restaurants and shopping in Yelp dataset and movies and electronics in Epinions dataset. For RMSE computation, we only consider those test data points for which the prediction component due to averaging from trusted users is non-zero. For Yelp dataset, we did not have explicit data indicating which users are trusted by a given user. Hence, we experimented with two different approaches. In the first approach, we assumed that a user trusts all the other users, (referred as *AllTrust*) and in the second approach, we assumed that a user only trusts his friends (direct connections in the social network, referred as *SocialTrust*).

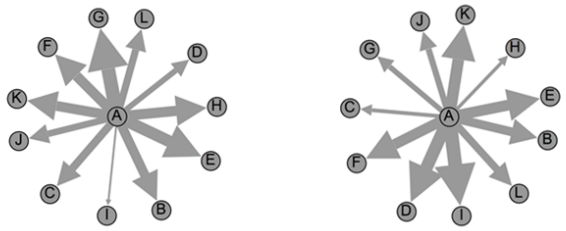

Fig. 2. Heterogeneous Trust learned from Electronics (left) and Movies (right)

The results for trust modeling using cosine trust and recommendation power are summarized in Table III for Yelp dataset and Table IV for Epinions dataset.

Both the heterogeneous *AllTrust* models improve over general trust models in the restaurant category for Yelp dataset. However, the performance in shopping category declines. One of the reasons for this behavior could be that we have fewer shopping businesses as compared to restaurants in our trust. In addition, many shopping places (malls) are also tagged as restaurants and user preferences/trust in restaurants should be similar to shopping to some extent. Hence, when we limit ourselves to shopping category for trust estimation, we lose the similarity information provided by the restaurant part of the user rating vector.

The *SocialTrust* heterogeneous models do not preform better than general trust models on Yelp dataset. This could mean that social connections alone are not a good indication of trust when it comes to recommendations. Also, most of the friends of active users on Yelp are themselves not active. Hence, lack of enough correlation between active users and their friends (due to their inactivity) makes it unreliable to treat social connections as trusted connections.

Epinions dataset doesn't suffer from the above problems with Yelp dataset. Movies and electronics, unlike shopping and restaurant, are mutually exclusive and uncorrelated categories. Also, in Epinions dataset, we explicitly know every user's trusted users (who are also active reviewers on Epinions). Hence, both the heterogeneous trust models improve over general trust models, corroborating our hypothesis.

TABLE II

SVM AND NAIVE BAYES ON YELP DATASET

Model	**Threshold = 3**	**Threshold = 4**
Naive Bayes	0.3609	0.1548
SVM	0.3425	0.1524

TABLE III

TRUST MODELING ON YELP DATASET

Model	Restaurant	Shopping
AllTrust Models		
Cosine (General)	0.9098	0.8604
Cosine (Heterogeneous)	0.8166	0.9133
Recommendation Power (General)	0.9080	0.8500
Recommendation Power (Heterogeneous)	0.8269	0.9124
SocialTrust Models		
Cosine (General)	0.7315	0.8543
Cosine (Heterogeneous)	0.7498	0.8602
Recommendation Power (General)	0.7324	0.8582
Recommendation Power (Heterogeneous)	0.7288	0.8632

TABLE IV

TRUST MODELING ON EPINIONS DATASET

Model	Movies	Electronics
Cosine (General)	0.8310	1.0735
Cosine (Heterogeneous)	0.8053	0.9088
Recommendation Power (General)	0.8138	1.6027
Recommendation Power (Heterogeneous)	0.7911	0.9726

D. Latent Factor Hybrid Models

For the latent factor+trust hybrid models, we divide the dataset into 70% data for training, 15% data for validation and 15% data for test. The learning rate γ was set to 0.01. Lower values of γ slowed down the training and higher values of γ led to divergence. We selected the values of α, β and K which led to the least RMSE on the validation set. This led to a value of $\beta = 0.01$ and $K = 20$. In Figure (3), we show the validation analysis for α ranging between 0.1 to 0.9 for $\beta = 0.01$ and $K = 20$. Similar to the other general trust models, we computed within the category RMSE values for the latent factor and general trust hybrid model. Figure (2) shows the strengths of trust learnt using latent factor+heterogeneous trust model for user A. The edge widths correspond to trust strength. It can be seen that while I is a highly trusted user for movies recommendations, he is least trusted for electronics.

The minimum validation error is obtained for $\alpha = 0.8$, $\beta = 0.01$ and $K = 20$. Total RMSE on test dataset for these values of hyper-parameters is 1.2024. For values of α between $0.7 - 0.9$, both the trust models perform better than pure latent factor models (dotted lines on the graph). Moreover, for too small values of α, the hybrid models perform poorer than pure latent factor models. This shows that user ratings are affected both by inherent user preferences and ratings given by trusted users. However, the heterogeneous trust hybrid doesn't improve considerably over the general trust hybrid. One possible explanation for this could be that the ratings distribution is highly skewed towards 4 and 5. Thus, for a given user, the strengths of trust might vary over movies and electronics. But all the trusted users might rate many products as either 4 or 5 making it difficult to improve the performance by modeling heterogeneous trust beyond a point.

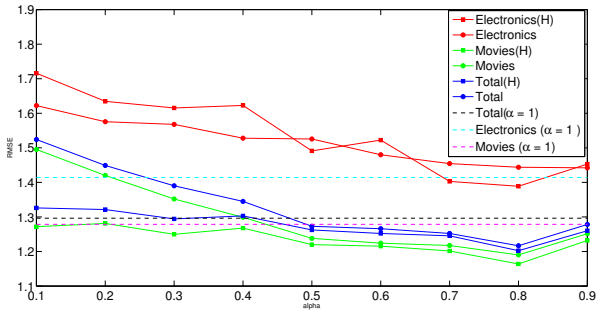

Fig. 3. Validation Error Analysis of Latent Factor Hybrid Models

E. Global Trust

Finally, we compare the performance of global trust measure (as defined in Methods section) with local trust measures (cosine and recommendation power) with respect to rating prediction on Epinions dataset. The total RMSE of rating prediction using global trust is 0.9876 whereas the local trust measures achieve an RMSE of 0.8590 (cosine) and 0.8435 (recommendation power) on the same test dataset. Thus, it looks like local trust connections affect user ratings more than ratings from experts. However, a more thorough analysis is required to make this claim with certainty.

V. CONCLUSIONS AND FUTURE WORK

In this project, we analysed the effect of incorporating heterogeneity in trust relations among users in a social network on traditional recommendation systems. We observed that heterogeneous trust improves performance when compared to general trust for recommendation power and cosine trust, when we have explicit information about user-user trust and uncorrelated product categories. The hybrid model combining latent and heterogeneous trust gives best performance for $\alpha = 0.8$ on the Epinions dataset. We could not achieve similar improvements on the Yelp dataset because we didn't have explicit information about trust between users and most of the reviews on Yelp were for restaurants making it difficult to find an uncorrelated category with sufficient number of reviews.

In this project, we only considered directly connected users as trusted users. However, it is possible to propagate the trust from directly connected users to other connected users using random walks on the user network. It would be interesting to explore these trust propagation algorithms. Another possible line of future work could be extending the work for multiple business categories on a larger dataset to obtain more concrete results.

REFERENCES

[1] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[2] scikit learn user guide [Online]. Available: http://scikitlearn.org/stable/user guide.html

[3] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.
[4] Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998.
[5] Allen, Robert B. "User models: theory, method, and practice." International Journal of man-machine Studies 32.5 (1990): 511-543.
[6] http://www.public.asu.edu/jtang20/datasetcode/truststudy.htm
[7] Au Yeung, Ching-man, and Tomoharu Iwata. "Strength of social influence in trust networks in product review sites." Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011.
[8] Jamali, Mohsen, and Martin Ester. "Trustwalker: a random walk model for combining trust-based and item-based recommendation." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.
[9] Leskovec, Jure, and Rok Sosic. "SNAP: A general purpose network analysis and graph mining library in C++." (2014).
[10] https://en.wikipedia.org/wiki/PageRank
[11] Zhou, Tao, et al. "Bipartite network projection and personal recommendation." Physical Review E 76.4 (2007): 046115.
[12] http://www.yelp.com/dataset