CS224W Final Report

Cyprien de Lichy, Renke Pan, Zheng Wu

December 9, 2015

1 Introduction

Recommender systems are information filtering systems that provide users with personalized suggestions for products or items. They have become an important research area since the appearance of the first papers on collaborative filtering. Much work has been done on developing new approaches to recommender systems over the last years because of the growing need of applications to help users deal with an increasing amount of information and provide them with personalized recommendations.

Nowadays, recommender systems are usually based on the users past behaviors and the similarity between users and/or between items. But such traditional recommendation systems suffer from several problems, one of them being data sparsity. Some users may have very few recorded (or even no) interactions with the items, and thus it is hard to predict what they might like, which is called cold start. However, there is a noticeable social trend that people that are socially related share interests and tastes, and thus, the knowledge about a person's social neighborhood structure could be used to infer its preferences to some degree. But conventional recommendation systems don't leverage this network structure to make predictions, thus depriving themselves of a rich source of information.

The idea of this project would be to take into consideration the relational structure of the network to improve the recommendation power, for both user network and item network.

2 Prior Work

Recommender systems are usually classified into collaborative filtering methods and content-based methods. There are also hybrid approaches that combine several recommendation method. For instance in [5] the authors used a combination of collaborative filtering and content filtering.

Except the two approaches mentioned above, people are exploring more and more other information such as social ties between people to help improve recommendation system's performance. Zheng *et al.* [1] were the first to introduce the method of social network collaborative filtering. Hill *et al.* [2] introduce a way to use the information of social network in collective inference without heavy computations. However, the computational complexity for conducting such inference technique on traditional social network is high. Thus, Hill *et al.* [2] suggests to only limit our interest to a small subpopulation (those with high commercial potential).On the other hand, features extracted from recommended contents' network could also make the system more robust. According to the study done by Teng *et al.* [6] to build recommendation system for recipes, two features can be extracted from recipes' network.

Sharma *et al.*[4] proposed a network-centric approach to recommendations and introduced PopCore, a platform that implements this approach on Facebook. They implemented 6 different algorithms, some of which are network-centric, to study and compare the effect of the recommendations and found that the network-centric algorithms indeed perform better.

Marlin *et al.*[7] showed that to improve the recommendation accuracy of a recommender system, the algorithm should not ignore the missing data mechanism and the MAR (Missing at Random) assumption is not valid, and we should model the MNAR (Missing Not at Random) phenomenon. In practice we observe that users tend not to give a rating to an item they didn't like than giving a bad rating. As a result the rating distribution is more skewed toward good ratings.

3 Data Collection and Visualization

We use Yelp Data for recommendation system implementation. The dataset contains the following information:

1. Users: Yelping since, votes, review counts, name, userID, friends, fans, average stars, type, compliments, elite.

2. Businesses: business ID, full address, hours, open, categories, city, review count, name, neighborhood, longitude, state, stars, latitude, attributes, type.

3. Reviews: votes, user ID, review ID, stars, date, text, type, business ID.

4. Tips :user ID, text, business ID, likes, date, type.

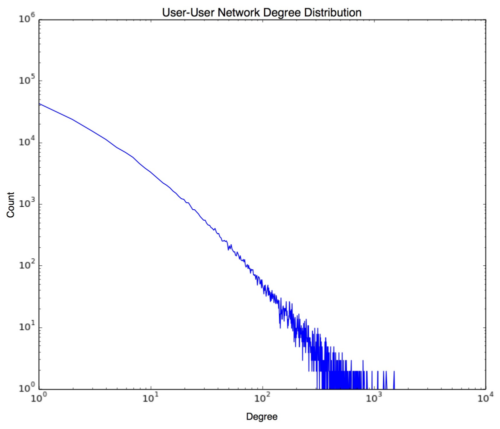
5. Check-ins : check-in info, type, business ID.

It has 1.6 Million reviews in total, covering 10 cities across United States, Germany, Canada and UK. 366k Users in the dataset form a large social network with 2.9 Million connections.

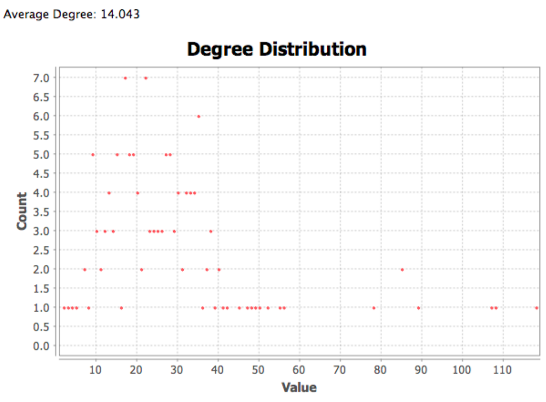
We choose only businesses in Pittsburgh in this case to speed up the calculation for the milestone. And only users who have reviewed one of the businesses included will be considered. Thus the data set we use contains 2724 businesses, 17175 users, 18358 user connections and 62325 user ratings.

3.1 User Network

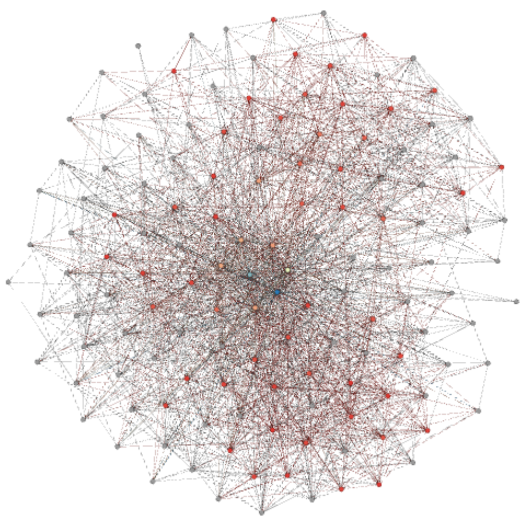
The degree distribution of user network in the whole data set.



The degree distribution of user network in the Pittsburgh data set.



For User network, if we only display nodes with above 33 degrees, below is the network. We could observe that there is no obvious clusterization.



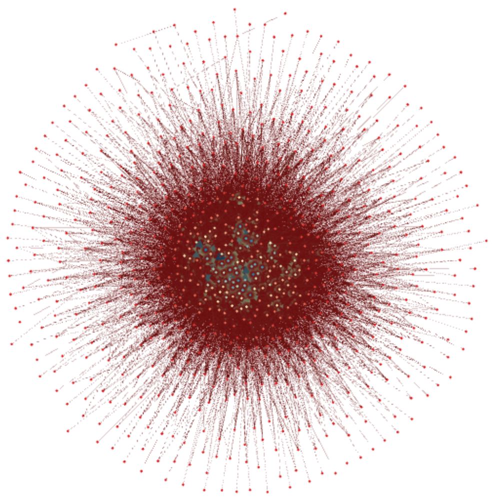
3.2 Business Network

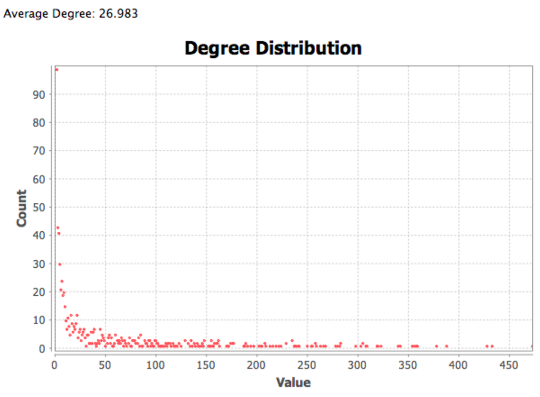
We define our current business network as follows:

Each business is a node in the network. If same user have been to both business A and business B, then we add an edge between business A and business B with weight 1. If business A and business B share k common users, then they have an edge between them with weight k.

To simplify the network, we limit the size of edges by putting certain threshold t for the formation of edges. In other words, we consider an edge exist if the weight is above t.

We use Fruchterman Reingold lay out method, with threshold 5.





4 Evaluating Metric

Social network based top-k recommendation recommends to a user a small number of items(K items) at a time. Most of the existing social network based recommender systems use the root mean square error (RMSE) as the optimization goal.

Top-k recommendations are more commonly used in real world problem. For these recommendation systems, the top-k hit ratio or recall is a more natural and useful metric compared with root mean square error.

The top-k hit ratio or recall is defined as below[8]:

For each user u, we sort the items i according to descending order of the predicted rating $\hat{R}_{i,u}$. When we have same predicted rating to different items, we rank them randomly.

An item is relevant to a user in the test set if s/he likes it (e.g., the assigned rating in the test data is above a certain threshold). For example, with Yelp data, the rating values range from 1 to 5 stars and we consider a 5 star rating as relevant(the user definitely liked these items), while other rating values and missing rating values are considered not relevant.

The top-k hit ratio or recall is the fraction of relevant items in the test set that are in the top-k of the ranking list, denoted by $N(k, u)$, from among all relevant items, $N(u)$.

For each user u, the top-k hit ratio is given by:

$$H(k,u) = \frac{N(k,u)}{N(u)}$$

With user top-k hit rate, we can compute the overall top-k hit ratio/recall by:

$$recall = \frac{\Sigma_u N(k,u)}{\Sigma_u N(u)}$$

A higher top-k hit ratio or recall indicates a better performance of our algorithm.

We also computed the RMSE when running our recommendation algorithms:

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{(u,i)\in T}\left(\hat{R}_{u,i}^{\text{test}} - R_{u,i}^{\text{test}}\right)^2}$$

Where T is a test set of observed ratings, $R_{u,i}^{\text{test}}$ is the ground truth value, $\hat{R}_{u,i}^{\text{test}}$ is the corresponding predicted rating.

5 Nearest Neighbor Methods

In a Nearest Neighbor method, top-k recommendations makes prediction only based on items adored by a subset of users who have some connection (under certain distance metric) with the target user. This kind of connection could be based on collaborative filtering or edges in the user-user network. Then we can integrate social information into Nearest Neighbor top-k recommendation system.

5.1 Collaborative Filtering Approach

The Collaborative Filtering Approach uses AllRank to obtain the user latent features[8]. The distance is computed through the Pearson correlation coefficient. Then we gather items from The U nearest neighbors of the target user. The voting for the candidate items is defined as follows:

$$Vote_{u,i} = \Sigma_{v\in N_u}\Sigma_i sim(u,v)\delta_{i\in I_v}$$

where δ is the Kronecker delta; I_v denotes the set of relevant items of user v; and N_u is the set of U nearest neighbors of user u (as determined by the Pearson correlation). For user u, item i , we have $Vote_{u,i}$; We give weights to neighbors according to their similarity sim(u, v) with the target user, which is defined as the Pearson correlation coefficient.

5.2 PureTrust Approach

PureTrust approach finds neighbors by implementing BFS in the user-user network starting from user u.

The voting is as follow:

$$Vote_{u,i} = \Sigma_{v\in N_u^{(t)}}\Sigma_i w_t(u,v)\delta_{i\in I_v}$$

where $N_u^{(t)}$ is the set of trusted users of u, and $w_t(u,v)$ is the voting weight of user v. The value of $w_t(u,v)$ is set to be $1/d_v$, where d_v is the depth of user v in the BFS tree rooted at user u.

5.3 Trust-Collaborative Filtering Approach

Trust-Collaborative Filtering Approach is designed to hybrid user latent feature space based collaborative filtering approach and social network based approach. In this approach, we first find U closest neighbors from the CF neighborhood, then find U closest neighbors from the trust neighborhood which are not in the first U neighbors. Then we gather information from the above neighbors:

$$Vote_{u,i} = \Sigma_{v\in N_u^{(c)}}\Sigma_i w(u,v)\delta_{i\in I_v}$$

where $N_u^{(c)}$ is the combined neighborhood and w(u,v) is defined as following:

$$w(u,v) = \begin{cases} sim(u,v) & \text{if } v\in N_u \\ w_t(u,v) & \text{if } v\in N_u^{(t)} \end{cases}$$

5.4 Result

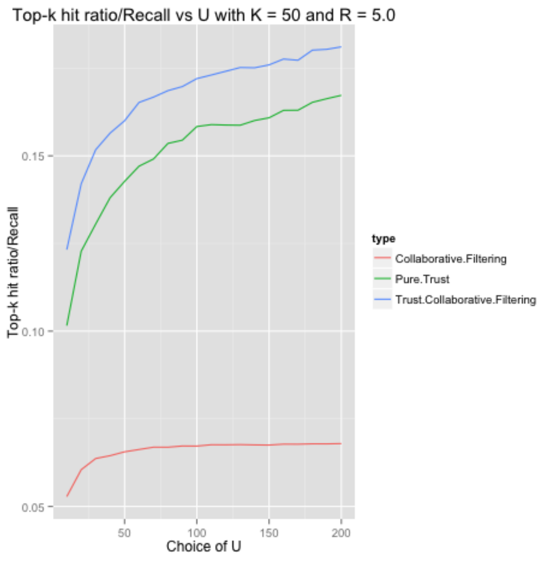
Our results show that trust information significantly improves top-k hit ratio when incorporated properly in Nearest Neighbor models.

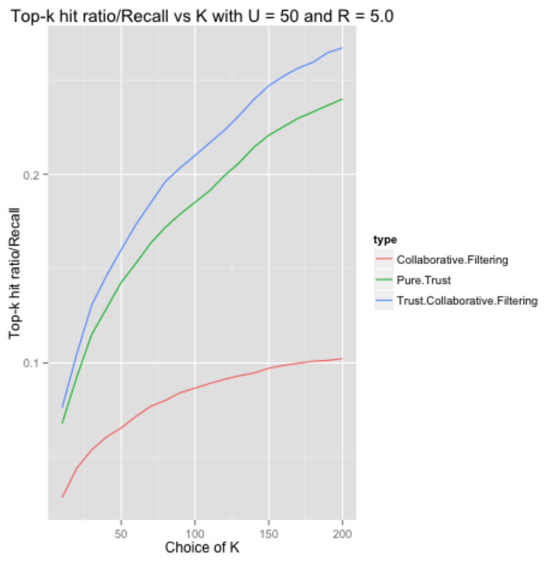
The top-k hit ratio for Trust-Collaborative Filtering Approach is the highest, followed by that of PureTrust Approach, which is significantly higher than that of Collaborative Filtering Approach.

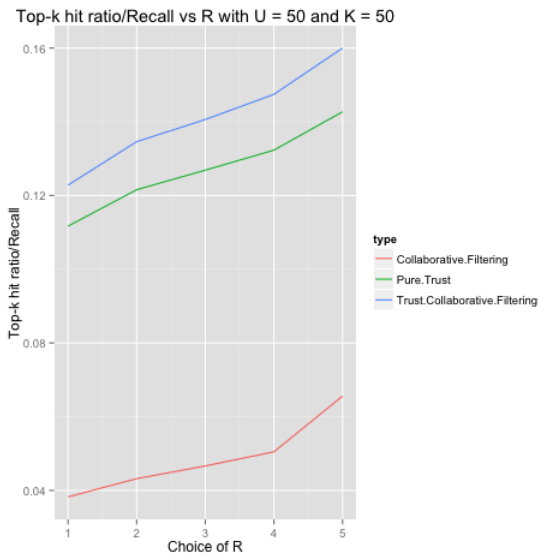
We have three parameters:

1. U, the number of users considered in the neighborhood, both in the collaborative filtering approach and the network based approaches.
2. K, the upper limit of the number of items recommended to a user.
3. R, the rating threshold used to determine whether a recommendation is relevant.

By changing these parameters, we got different top-k hit ratios. The next three figures show how the top-k hit ratio change with respect to different values of U, K and R:







We can see that recall increases as U increases, fixing K and R. The recall for collaborative filtering approach reaches a plateau after certain threshold. This is probably because there is not enough users that rated the same restaurants and thus we cannot get a neighborhood of size U after certain large enough U. On the other hand, the recall for the two network-based approaches keep increasing as U increases. This is probably because the largest connected component is pretty large (it turns out 46% percent of all users are in the largest connected component) so we are always able to find a large enough neighborhood in the network. From the period before the collaborative filtering approach reaches the plateau, we can conclude that introducing network features significantly improves the recommendation.

We can see that recall increases as K increases, fixing U and R. This is expected because we are increasing the nominator of recall while keeping denominator fixed.

We can see that recall increases as R increases, fixing U and K. This is expected because we are increasing the nominator of recall slightly but the denominator increased drastically if we decrease R.

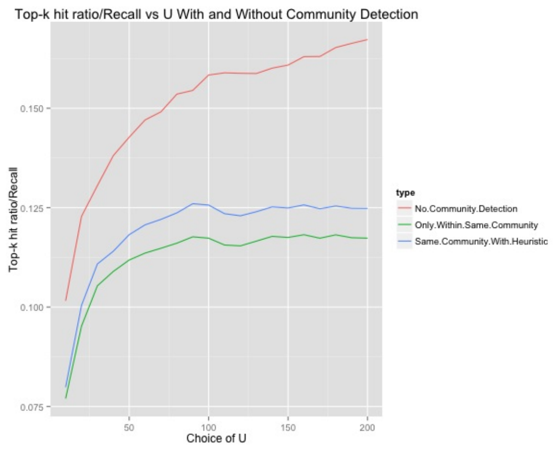
So overall we see the recall increases with U, K, R. Larger U, K, R always lead to better recall. However, the choice of U, K, R cannot be arbitrarily large. For U, the larger the U, the longer it takes to finish the computation. For K, we cannot recommend say 1000 restaurants simultaneously to a user. A reasonable K would be somewhere below 50. For R, 5 seems to be the best not only because it is the largest possible value, but also because it makes the most sense in practice. Giving a 5 star review suggests the user absolutely loved the restaurant.

5.5 Community Detection

We hypothesized that adding in community detection to pure trust approach is going to improve the hit rate.

The assumption is that users within the same community share the same taste. Thus giving recommendation based on users from the same community will likely increase the hit rate/recall.

We first implemented a version of PureTrust that restricts neighbors to only those that are within the same community, detected by Clauset-Newman-Moore community detection method.



As can be seen from the above graph, it turned out that the hit rate actually decreased, which is quite unexpected. Then we dig deep into this and brainstormed what could be causing this. We think the reason is that not every one of a user's nearest neighbor falls within the same community with the user. Thus these users, which actually has the best predictive power, are excluded when making the recommendation.

As an attempt to justify this hypothesis and also improve the recommendation, we introduced a heuristic: all the users who are in the same community as the user of interest are closer to this user in the network. So once we skipped through number of users in the same community, we no longer enforce same community rule when determining if a user should be considered, i.e. always consider them. This heuristic in fact increased the recall. Our previous hypothesis is thus justified. However, the improvement in the recall is not enough. It still falls much lower than the original approach. Further more, both curves with community features reach a plateau. This is because they both skipped some close users in the network but are not in the same community and these users have the most predictive power. Including more users further out doesn't really help the prediction much because the similarity between users is much lower.

6 Matrix Factorization

The Matrix Factorization approach was found to be the most accurate approach to reduce the dimensionality of the problem when a high level of data sparsity is observed.

6.1 Basic approach

In this model, the rating matrix R of dimensions $u_0 \times i_0$ (u_0 being the number of users and i_0 the number of items) is represented as the product of 2 low-rank matrices P and Q. The matrices P and Q are latent-features representation of items and users respectively, and they are said to be low-rank: $\text{rank}(P) = \text{rank}(Q) = d$, with $d \ll u_0$ and $d \ll i_0$. The MF model is a kind of dimension reduction, the objective is to find the matrix $\hat{R} = QP^T$ that best fits the original matrix R (which is sparse because we have only a few items rated by each user). Q is of dimensions $u_0 \times d$ and P is of dimensions $i_0 \times d$. To take into account the phenomenon of MNAR (Missing Not at Random) of the ratings, an offset rating value r_m is added in the model in the following form:

$$\hat{R} = r_m + QP^T$$

One way to proceed to maximize the recall on the top-k recommendations would be to optimize directly on the top-k hit ratio but this is too costly computationally. Instead, we can optimize a modified squared error criterion that will lead to a better top-k hit rate than the classical squared error:

$$\min_{P,Q} \sum_u \sum_i W_{u,i} \left(\widetilde{R}_{u,i} - \hat{R}_{u,i} \right)^2 + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 \right) \tag{1}$$

Where $\widetilde{R}_{u,i} = R_{u,i}$ if the rating is observed and $\widetilde{R}_{u,i} = r_m$ otherwise, r_m is the imputed rating when there is a missing value, it is a tuning parameter of the model, and according to the study by Yahoo! (See [7]) we expect its optimal value to be rather small. The matrix W is introduced to balance the dataset (which is very sparse in reality) by giving more weight to the observed ratings. In addition to that we also have the typical L2 regularization term to avoid over-fitting.

6.2 Including the social network

Now, we can include the networks (user-user and business-business networks) information by including more terms i the following way. We introduce two matrices U and B representing the relationship between two users and two businesses respectively, U and B could be weighted adjacency matrix, it is proposed in [8]:

$$U_{u,v} = A_{u,v} \sqrt{\frac{\deg(v)}{\deg(v) + \deg(u)}}$$

where A is the user-user adjacency matrix. The objective function to minimize becomes the following:

$$\begin{aligned} \min_{P,Q,V} \sum_u \sum_i W_{u,i} \left(\widetilde{R}_{u,i} - \hat{R}_{u,i} \right)^2 \\ + \gamma \sum_u \sum_v W^U_{u,v} \left(U_{u,v} - \hat{U}_{u,v} \right)^2 \\ + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 + \|V\|_F^2 \right) \end{aligned} \tag{2}$$

Where $\hat{U} = QV^T$, V is of size $u_0 \times d$.

6.3 Including a business-business network

In our approach we imagined doing the same for businesses, by constructing a business-business network using the data about the businesses (like descriptions, reviews) and the users. Basically, we would build a network by linking two businesses if they have mutual customers and similar descriptions. In that way we use more available data than just the ratings history to drive the CF computations for the recommender system. The objective function to minimize would be the following:

$$\begin{aligned} \min_{P,Q,V,C} \sum_u \sum_i W_{u,i} \left(\widetilde{R}_{u,i} - \hat{R}_{u,i} \right)^2 \\ + \gamma \sum_u \sum_v W^U_{u,v} \left(U_{u,v} - \hat{U}_{u,v} \right)^2 \\ + \beta \sum_i \sum_j W^B_{i,j} \left(B_{i,j} - \hat{B}_{i,j} \right)^2 \\ + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 + \|V\|_F^2 + \|C\|_F^2 \right) \end{aligned} \tag{3}$$

Where $\hat{B} = PC^T$, and C is of size $i_0 \times d$.

6.4 Social influence weighting

By varying the weights $W_{u,i}$ of the observed ratings according to the eigenvector centrality measure of user u, we are giving more weights to the important/influential users, this forces the algorithm to better fit the matrix factorization to these users and these better results may diffuse in the network (via the social part of the objective function) and yield better overall results. This comes from the intuition that better fitting the more influential users is more useful for the overall performance.

6.5 Optimization algorithm for the MF models

We implemented all of these MF models using an alternating optimization algorithm: Alternating Least Squares, which is an efficient algorithm for this task. At each step of the algorithm 3 of the unknown matrices (P, Q, V and C) are fixed and the other one can be updated to maximize the objective function via an exact formula.

6.6 Results

Model	RMSE
MAR (r_m, $w_m = 0$)	1.61
MNAR (r_m, $w_m = 2$, 0.0001)	1.11
MNAR Social Network	1.02
MNAR Social & business Network	1.01

From this table we can see that by taking into account the phenomenon of MNAR we can greatly improve the RMSE at the expense of finding the optimal values of 2 tuning parameters (r_m and w_m). We can also see that using information about the structure of the social network (the adjacency matrix and the eigenvector centrality measure) can also improve the RMSE. However, we didn't get a noticeable improvement by considering our business-business network maybe because our basic construction of the network does not bring new information, the convergence of our algorithm is also longer.

7 Temporal Effects

When we first proposed the project, we were thinking about incorporating the temporal effects of the ratings and the network. However, when we actually want to go ahead and do it, we found out there is no timestamp of the friendship formation. So the network is a static snapshot, not evolving over time. However, we do have the review timestamp. The usage of the review timestamp is limited though, besides splitting the data into train and test set, predicting future reviews from past reviews, there is not an intuitive way of utilizing this information. Thus studying the temporal effects is not well suited for this data set.

8 Reproducibility/Robustness

We focused our analysis with Pittsburgh because Pittsburgh's data is diverse enough and complete yet small enough for fast prototyping.

One concern is that the overall network structure is different than the one in Pittsburgh. This is a totally valid concern. However, the data consists of restaurants from the following cities:

- U.K.: Edinburgh
- Germany: Karlsruhe
- Canada: Montreal and Waterloo
- U.S.: Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison

First of all, this is not the complete graph of the overall yelp network anyway. It's only a small portion of the global graph structure. Further reducing the graph to one city isn't any fundamentally different.

Secondly, the cities are in disjoint regions of the country or even different countries. we would expect the graphs of each city to be a relative independent connected component and there is much less inter-city connections. Therefore, studying each individual city by itself is a valid approach. In fact, to make the most relevant prediction, I suspect Yelp would use local information heavily as well.

Therefore, it doesn't really make sense to run the analysis with the whole data set. If we did that, We would spent way too much computation time on an improperly defined network.

To make sure our analysis is robust and can be reproduced. We did the exact same analysis on Phoenix data, which is diverse and complete yet substantially larger than Pittsburgh. All the trends, relationships, orderings are retained with Phoenix data. To save space, we did not include the plots here. We have thus gained confidence that our approach can be reproduced in an independent data set and is robust.

9 Difficulties

1. The existing recommendation packages do not satisfy our need (mostly because the flexibility of the algorithms is not satisfying) and our data is not ready to use for recommendation, so we had to implement all the algorithms and data processing from scratch.
2. The size of the data poses problems for details of implementing the algorithms. For example, we run out of memory when doing matrix operations and had to use sparse matrices and resort to computational tricks.
3. The dataset is huge and the running time for the algorithms is rather long.

10 Future Works

1. Further optimize the model, minor tweak the algorithms to have the best performance.
2. Have another data set, different from Yelp, which has friendship formation timestamps, to study temporal effects.

11 Acknowledgment

We would like to thank Lada and the TAs for providing a wonderful quarter of learning experience and insightful feedback on the project.

References

[1] R. Zheng, F. Provost and A. Ghose. Social Network Collaborative Filtering: Preliminary Results. *Proceedings of the Sixth Workshop on eBusiness (WEB2007)*, New York, USA, December 2007

[2] S. Hill, F. Provost and C. Volinsky. Learning and Inference in Massive Social Networks. *5th International Workshop on Mining and Learning with Graphs*, August 2007

[3] G. Adomavicius and A. Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Trans. Knowl. Data Eng . 17(6): 734-749, 2005

[4] A. Sharma and D. Cosley, Network-Centric Recommendation: Personalization with and in Social Networks, *IEEE International Conference on Privacy, Security, Risk, and Trust*, 2011

[5] A. B. Barragns-Martnez, E. Costa-Montenegro, J.C. Burguillo, M. Rey- Lpez, F. A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Information Sciences, 180(22), 4290-4311. 2010

[6] C.-Y. Teng, Y.-R. Lin, and L. A. Adamic. Recipe recommendation using ingredient networks. *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 298–307, 2012.

[7] Marlin, B.; Zemel, R.; Roweis, S; Slaney, M., Collaborative Filtering and the Missing at Random Assumption

[8] Yang, X.; Steck, H.; Guo, Y; Liu, Y., On Top-k Recommendation using Social Networks