Evaluation of Graph Based Recommendation with Other Common Collaborative Filtering Methods

Ryan Foley and Qinghui Ji

Abstract

Recommendation systems have come to be of great importance for the online retailer and they are incorporated in to many websites today. However, many different recommendation methods exist and the data collected with which to make recommendations varies from site to site. Thus, in this paper, we evaluate several collaborative filtering based methods, including one graph based method, on samples of data with varying sparseness. To do so, we investigate data from BeerAdvocate.com and draw conclusions about each models efficacy. We believe that this paper may have relevance particularly in the online retail and marketing spaces.

1.0 Introduction

The greatest advantage Internet retailers have over traditional brick-and-mortar stores is the ability to sell more obscure items. “On Amazon.com, somewhere between 20 to 40 percent of unit sales fall outside of its top 100,000 ranked products” [1]. Traditional retailers are limited by what can be stocked on the shelves and therefore often resort to merely displaying the most popular items. On the other hand, online stores are not limited by shelf space and therefore can present a seemingly infinite list of products.

Although this is a great advantage for the online retailer, deciding which items to display to the customer can be a difficult task. For an online shopper looking at a product that they may have never seen or heard of before, it is essential for the recommendation system to suggest those that the user will truly like in order for repeated business and for that customer to trust future recommendations. This challenge has given rise to the creation of advanced recommendation systems. Thus, in this paper, we evaluate several methods, including one graph-based method, on samples of data with varying degrees of sparseness and numbers of items.

Recommendation systems are typically either content-based, using features of the items to recommend those that are similar, or collaborative filtering based where recommendations are made from patterns observed in users' previous interactions, ratings, or purchases of different items [2]. For our study, we chose to evaluate collaborative filtering methods on voting data of various beers from the website BeerAdvocate.com.

2.0 Data Set

The data set for our study comes from the BeerAdvocate.com website where users can rate a beer from 1 to 5 on an increasing scale of preference. In total, our dataset consists of 1,586,614 reviews of 66,055 different beers from 33,388 active users. Although there exists data from the site of each user's written review and other features, we only look at votes, or what we also refer to as ratings, of beers by individual users. To evaluate our models, we select only samples of this data set as described in §5.0.

2.1 Votes

Table 1 shows a summary of the votes, or ratings, throughout the entire dataset. Noticing that the mean and median vote for a beer lie around 4, we decide to evaluate our recommendation models later using 4 and above to indicate that the user preferred the beer. Although this assumption is seems somewhat arbitrary, it allows us to think of the beers as either good or bad for a particular user and gives us a way in which to evaluate a recommended beer for that user.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.000	3.500	4.000	3.816	4.500	5.000

Table 1. Summary of votes throughout entire data set.

2.0 Methods

In this paper, we evaluate a graphical recommendation model against three other common collaborative filtering methods. As a baseline, we also consider mean user rating for each beer. Across all of these models, we use the collaborative filtering idea of taking past user ratings, or votes, to predict and recommend items that the user would prefer. In this section, we describe each of these models.

3.1 Graph-Based Model

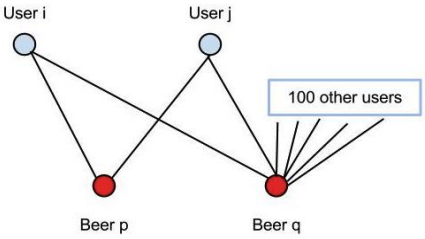

Fig. 1

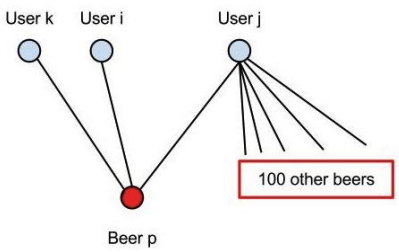

Fig. 2

In this method, we consider the ratings data as a bipartite graph with users and beers as nodes and the users' votes for a beer as edges between them. We use the term weight to mean the similarity between user u_i and user u_j. If we only look at the common beers that two users rated, as in Fig. 1, then a beer b_p that has only been reviewed by u_i and u_j means the same as a beer b_q that has been reviewed by 102 users. Thus, we take into account the degree of the beer with respect to its connections to users. If a beer is very popular, then it is hard to say whether two people voted for it have similar taste. In contrast, if a beer is exclusive to only two voters, then it is more certain that these two have similar taste. The weight formula used here, Eq. 1, is adopted from Zhou et al.

$$weight(u_i, u_j) = \sum_{b_k \in B} \frac{1\{b_k \in neighbors(u_i) \wedge b_k \in neighbors(u_j)\}}{degree(b_k)} \quad (1)$$

Then, the score given to a certain beer for a user is equal to the sum of weights between user u and all the other users who have voted for the same beer, divided by the degrees of those users, as seen in Eq. 2. We can think of the scores as creating a new bipartite graph with the same nodes, but new scored edges between the users and the beers. If we only add up the weights between other users and user u, then it ignores the activeness of user behavior. For example, in Fig. 2, user u_k only voted beer b_p, and user u_j voted b_p and another 100 beers. From this observation, we increase the weight between u_i and u_k because u_k devotes all of his or her reviews to only one beer. Furthermore, we decrease the weight between u_i and u_j because for u_j, b_p is one of his or her 101 beer reviews.

$$score(u, b) = \sum_{u_i \in neighbors(b)} \frac{weight(u, u_i)}{degree(u_i)} \times vote(u_i, b) \quad (2)$$

In the first trial of evaluation, we simply multiply the vote, as in Eq. 2. The testing set showed both high TPR (True Positive Rate) and high FPR (False Positive rate) while giving few recommendations. It indicated

that the model could predict the beers that showed up in the testing set very well, but failed to distinguish a highly rated beer from a low rated one. Since we defined in the evaluation a good recommendation to be a beer that would be rated 4 or more, we wanted our model to suggest beers with these higher ratings. Thus, for Eq. 2, we updated the vote between user u_i and beer b to be the original vote minus 3.5. Now we define our equation for the score between user u and beer b to be that shown in Eq. 3.

$$score(u,b) = \sum_{u_i \in neighbors(b)} \frac{weight(u,u_i)}{degree(u_i)} \times (vote(u_i,b) - 3.5) \qquad (3)$$

With this updated score, we found better results because beers with poor votes received lower scores than those with good votes.

For a specific user u, this algorithm calculates a score for every beer in the training dataset. If a beer does not appear in the training set, it is assigned a score of 0 because we cannot make any assumption on whether it is a good recommendation or not, but we can be sure that it is better than those with negative scores. Then after sorting this array of scores in a descending order, we obtain a list of beers that ranks from most likely to least likely to be preferred. In some special cases, if a user never appears in the training set but shows up in the testing set, another recommendation list is created. The score for each beer is the sum of its scores from all the users in the training set, Eq. 4. In this case, we take both the rating and popularity of a beer into account.

$$score(b) = \sum_{u_i \in U_{training}} score(u_i,b) \qquad (4)$$

3.2 Matrix Factorization Model

Matrix factorization as a recommendation method was made popular by the Netflix Prize in 2007 and has been shown to have great success [4]. The benefit of this model over item-based or user-based collaborative filtering is that it maps the user-item rating matrix to a latent factor space characterized by patterns in the observed voting data. Then, this mapping can be used to approximate missing values in the data. Each item i is associated with a vector $q_i \in R^f$ and each user u is associated with a vector $p_u \in R^f$ where f is the dimensionality of the latent vector space. The elements of q_i and p_u represent the level at which the corresponding item or user possesses the latent factors. The dot product of the two vectors, ${q_i}^T p_u$, then represents the estimated rating, $\hat{r}_{ui}$, of item i by user u [4].

Now, consider all of the observed ratings as a matrix, X, whose number of rows equal that of the total number of users, number of columns equal the total number of items, and each entry is the corresponding user-item rating, r_{ui}. We can make an approximation of the entire ratings matrix, which we denote as $\hat{X}$, by taking the truncated singular value decomposition of X, as described by Eq. 5.

$$\widehat{\boldsymbol{X}}_{\boldsymbol{f}} = \boldsymbol{U_f D_f V_f^T} \qquad (5)$$

Here $\widehat{\boldsymbol{X}}_{\boldsymbol{f}}$ is the rank f approximation to X and each entry of $\widehat{\boldsymbol{X}}_{\boldsymbol{f}}$ represents $\hat{r}_{\boldsymbol{ui}}$. From this notion, we can impute the missing ratings of the mean centered ratings matrix using the SVD imputation algorithm defined by Hastie et. al [5],

1. Initialize all missing values to the mean of the non-missing entries of each row, producing a complete matrix X^0. Set $i = 0$.

2. Compute the rank f approximation of X^i, and produce X^{i+1} by replacing any of the cells that were originally missing in X with the fitted values of X^i.

3. Update i to $i + 1$ and repeat step 2 until convergence where the criteria is that

$$\frac{||X^i - X^{i+1}||}{||X^i||} \leq some\ threshold, \epsilon\ (10^{-6}) \qquad (6)$$

To determine the dimensionality of the latent factor space, or the value of f, we choose the value that minimizes the mean squared error of our estimated ratings and some hold out set. However, because the purpose of this paper is to compare several models, we take a training sample separate of the sample used to evaluate all the models - here separation implies that no two users are in both samples.

One difficult aspect of using SVD imputation is that it can be computationally expensive to compute the entire SVD at every iteration. To work around this, we used the irlba package for R which calculates an approximation of the partial SVD in a way that is more memory and computationally efficient [Lewis]. After some preliminary testing, we found this package to be faster and comparable to the R base package. Furthermore, it would be interesting to note that we investigated other matrix factorization models, such as with regularization, but did not implement them [4].

3.3 User and Item Based

Two common collaborative filtering methods that we also consider are user-based and item-based models. For user-based collaborative filtering, we estimate a missing rating between user u and item i by finding the k most similar users to u and averaging their votes for item i. Thus,

$$\hat{r}_{ui} = \frac{1}{k}\sum_{u_j \in U^k} r_{u_j,i} \qquad (7)$$

where we define the set of k users in U^k to be the users which minimize the Euclidean distance between their voting history and that of u. Similarly, item-based collaborative filtering estimates the missing rating, r_{ui}, using the average of the k most similar items to item i. Other item and user based collaborative filter methods often use different distance metrics, such as cosine distance, or take a weighted average by the similarity of the items in computing $\hat{r}_{ui}$. However, these additions were not evaluated on our data set, so we only use our slightly simpler but fundamentally similar user and item based models. The parameter k is determined by tuning on a held out training set similar to that of the method used in determining f for our matrix factorization model.

4.0 Evaluation and Results

The entire dataset of the BeerAdvocate.com site consists of 66,055 beers. Thus, we decided to use only the most reviewed subcategory of the beers, which happened to be IPAs. This category consisted of 3611 beers, which gave a more manageable number of items with which to evaluate our models. Furthermore, because this set was the most reviewed, there were enough users and reviews to evaluate our models under varying degrees of sparseness. In building a recommendation system, it is often the case that the number of available items far exceeds that with which the average user has any interaction. Thus, we thought it would be valuable to look at the performance of each model on three different random samples of 100 users where the number of reviews from each user was from 3 to 19, 20 to 100, and 80 or more. Note that there was some overlap in our data sets due to the need to increase the size of the third to be large enough for both training and testing of the matrix factorization, user, and item based models. To identify each of these data sets individually, we refer to them as “sparse”, “average”, or “dense”. Table 2 shows the mean, median and quartiles of the number of reviews made by users in each set.

Dataset	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
"Sparse"	3.0	4.0	5.5	6.7	9.0	19.0
"Average"	21.00	27.00	33.00	38.18	45.25	93.00
"Dense"	80.0	86.0	103.0	116.7	130.5	354.0

Table 2 Summary of number of reviews per user in each sample data set.

In order to validate the results from each model, we randomly selected 10% of the ratings from each data set to be held out and developed recommendations from the remaining 90%. We were careful to use the same hold out set for all of the models. Then, each model was able generate an ordered list of the beers from most recommended to least for each individual. In order to create the ROC plots shown in Fig. 5, a recommendation list the length of the total number of items had to be created. Therefore, any item that had already been rated was placed at the end of the list in random order. As the length of the recommendation list is varied for a particular model, we count the number of true positives and false positives made by the model. We define the total number of true positives in Eq. 8 as the number of items in a recommendation list for a user u where that user-item pair or rating for that item, r_{ui}, is in the held out set and that $r_{ui} \geq 4$. We chose the rating of a 4 to be the threshold for a "good" rating because it was close to the mean of all ratings and seemed appropriate. Similarly, we define the total number of false positives in Eq. (9).

$$TP = \sum_{u \in U} \sum_{i \in L_u} 1\{r_{ui} \in H \text{ and } r_{ui} \geq 4\} \quad (8)$$

$$FP = \sum_{u \in U} \sum_{i \in L_u} 1\{r_{ui} \in H \text{ and } r_{ui} < 4\} \quad (9)$$

$$Hitting\ Rate = TPR = \frac{TP}{P} = \frac{TP}{\sum_{u,i} 1\{r_{ui} \in H \text{ and } r_{ui} \geq 4\}} \quad (10)$$

$$FPR = \frac{FP}{P} = \frac{FP}{\sum_{u,i} 1\{r_{ui} \in H \text{ and } r_{ui} < 4\}} \quad (11)$$

where U is the set of sampled users, L_u is the set of items recommended for user u, and H is the held out set of ratings.

Also, from the total number of true positives, we define the true positive rate in Eq. 10. This is also termed the hitting rate and it was used in some of our readings to evaluate recommendation methods [3]. Fig. 5 shows the hitting rate as a function of the length of the recommendation list for each of our models on the three data samples.

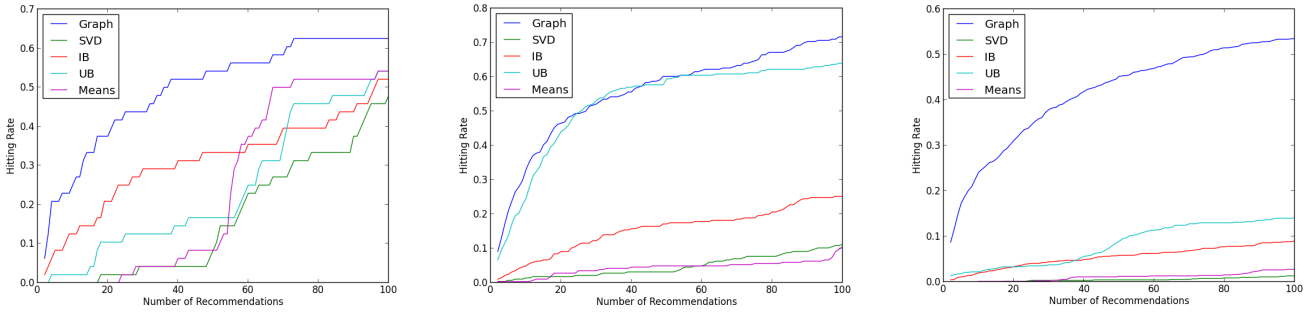

Figure 3. Hitting Rate or TPR as a function of the length of the recommendation list with number of reviews per user in the sample increasing from left to right.

Fig. 3 displays the graph based method to have a much higher hitting rate than the other models for the first one hundred recommendations. This is one of the primary criteria by which Zhou et. al measure the performance of the graph based method and base their conclusion that it is the superior model [3]. However, we find that this is an incomplete evaluation of the graph based model and a possibly misleading comparison of its performance against the other recommendation methods. First off, we also plot the false positive rate of each model as a function of the number of recommendations to each user in Fig. 4.

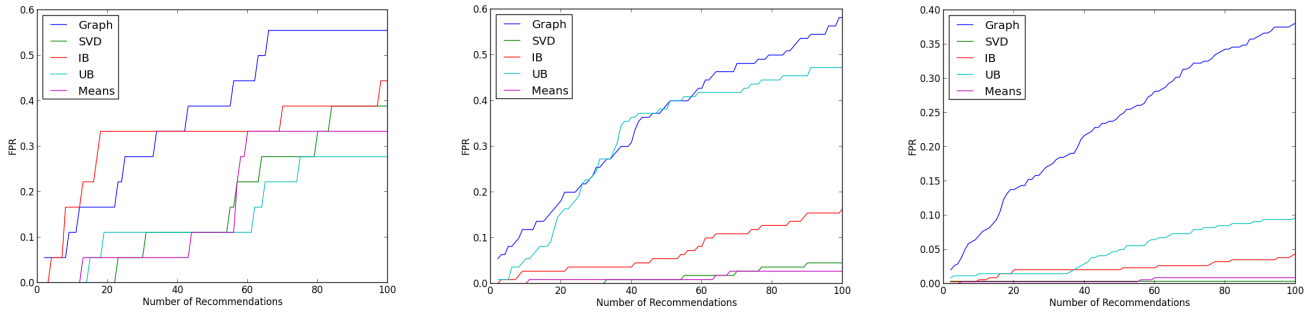

Figure 4. False Positive Rates as a function of the number of recommendations to each user across all of the models on sample data sets with increasing density from left to right.

It is clear from Fig. 4 that the graph based method also has the highest FPR when making 1 to 100 recommendations in each sample. We believe that the claim that the graph based method is a superior recommendation system is not substantiated by looking at the hitting rate. Thus, we will show several other metrics that we believe more completely describe each model's performance and characteristics.

What can be derived from Figs. 3 and 4 is that the graph based method is more quickly identifying which items would be rated by a user, disregarding whether that rating is high or low. However, to better assess the overall performance of the models we will look at their respective ROC curves and precision as a function of the number of recommendations. Fig. 5 shows the complete ROC curve for each model and Table 3 contains the Area Under the Curve of each model in each sampled data set. The ROC curve was derived by plotting the TPR against the FPR as we increased the recommendation list from one item to the total number of items reviewed by members within the dataset.

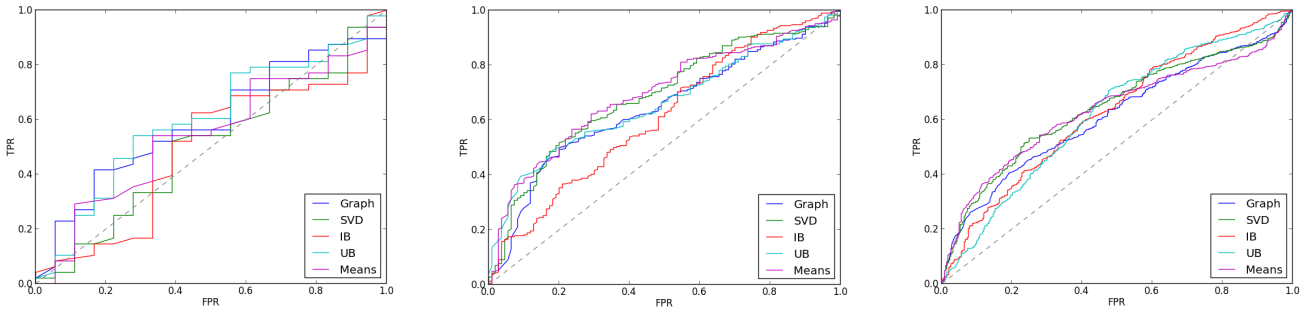

Figure 5. ROC curves for each model on sample recommendation data with increasing density from left to right.

Dataset	Graph Based	SVD	Item Based	User Based	Means
"Sparse"	**.5880**	.4954	.4896	**.5961**	.5405
"Average"	.6599	**.6851**	.6102	.6579	**.6925**
"Dense"	0.6111	**0.6428**	0.6269	0.6205	**.6339**

Table 3. Models with associated AUC on each sample data set.

From the AUC values in Table 3, we see for the "sparse" set that the graph based and user based methods more often recommend beers that the user rated highly whereas the matrix factorization model, or denoted as SVD in the figure and table, performs best for the "denser" set. We include the mean user rating as a recommendation

method, Means in Fig. 5, as a baseline to represent recommendations made merely on popularity and see that it performs fairly well throughout the data sets. We believe that if the number of items, or beers, were increased significantly, then the relative performance of mean user rating recommendations would decrease. However, this experiment was not conducted and is included as a suggestion for future study in §5.0 of this paper.

Although the ROC curves and AUC display a good measure of each model's performance as we increase the length of the recommendation list to the total number of items, most recommendation systems are built in practice to only recommend a top few items to a user. Thus, we examine the precision of each method as a function of the number of recommended items. Here we define precision as

$$Precision = \frac{TP}{TP + FP} \qquad (12)$$

where TP and FP are as defined in Eqs. 8 and 9 - if the denominator in Eq. 12 was observed to be zero, which was often the case for small numbers of recommendations, precision was calculated as zero. It may be important to note that accuracy, recall, and specificity are also common metrics for evaluating most classification methods. However, they were not included here because we do not believe that the notion of a false negative could be applied to the recommendation model. For a particular user, if an item was left out of the list, then it was because the model deemed that item not to be as highly recommended relative to the others and the model is not necessarily making a binary classification by discouraging the item for the user. Fig. 6 shows the precision of each model as a function of the recommendation list as we vary the number of recommendations from one to one hundred and Fig. 7 shows the same for the number of recommendations up to the number of the total items.

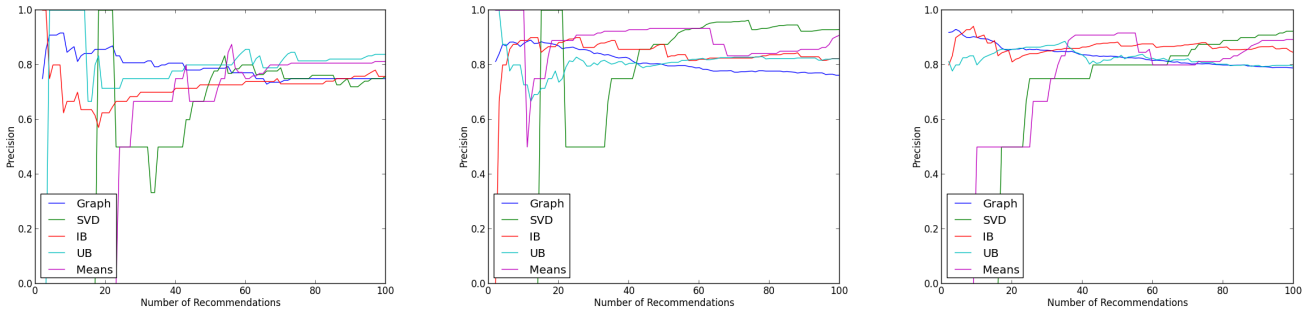

Fig. 6 Precision as a function of the number of recommendations for each model on samples with density increasing from left to right. Note higher values for the graph based method in all cases for up to about 20 to 40 recommendations where inflection between model performances occurs. If the total number of true positives and false positives was observed to be zero, then the precision was calculated as zero.

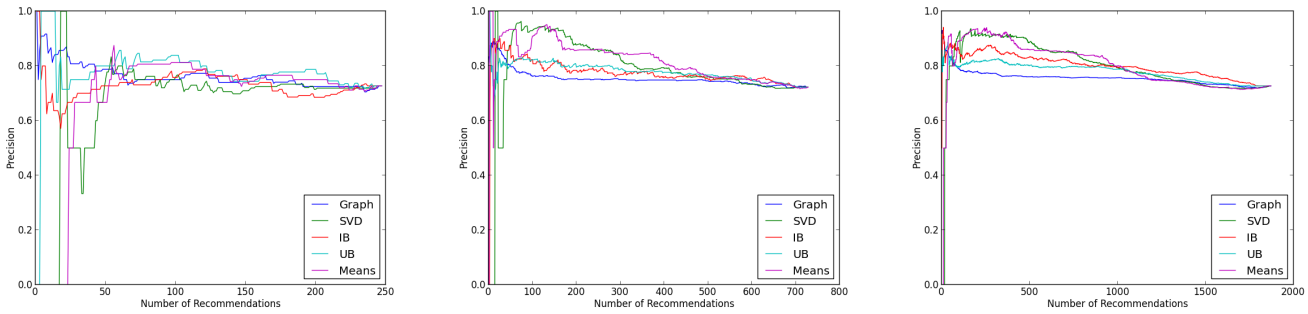

Fig. 7 Precision as a function of the number of recommendations for each model on samples with density increasing from left to right. Note that these are the same curves as in Fig. 8, but extended for the number of recommendations up to the total items rated.

From the plots in Fig. 6, it is now clear that the graph based method outperforms the other methods in the sparse data set for up to about 40 recommendations, after which the graph based method underperforms in comparison to the others and the user based method and rating means perform slightly better. Interestingly, as the density increases in the data, we see that this inflection in relative model performance repeatedly occurs close to 40 recommendations but earlier in the number of recommendations. After the inflection and by 100 recommendations in the denser sets, the performance of the SVD imputation method rises to the top whereas it did not in the sparser sample. In fact, if we look at Fig. 7, the SVD method underperforms throughout in the sparser set.

From these findings we conclude that the graph based method performs best for a shorter number of recommendations, especially for under 20 in our data, and particularly in sparser data sets whereas it remains a top performer in denser data sets but is comparable to item based recommendation. Furthermore, the graph based method more accurately identifies which items would be rated at all by users. For larger numbers of recommendations, matrix factorization performs best. Although mean rating was observed to perform well, we believe that mean rating would prove to under perform in the presence of much greater numbers of items whereas the dimension of the factors in the matrix factorization method could be increased to accommodate the extra complexity in the data. Lastly, we observed the user based method to perform well in the sparse data because of the size of our sample and the smaller number of items. As the density increased, the number of items increased significantly and the item based method outperformed the user based.

5.0 Further Improvement

The data sets we used to evaluate our recommendation models are all within one style of beer. In reality, a website or online retailer would likely have to make recommendation across a much larger set of items. Thus, we believe that the above analysis could be repeated on a much larger item space to further substantiate or revise our claims. Given the computational complexity and memory required for some of these models, this would be a harder task. Furthermore, the models that we used were more basic than some others that have been developed [4][7]. However, most of these others build upon similar principles as those used here and are likely to be comparable. For instance, the matrix factorization method could include regularization to combat overfitting and the item and user based methods could be tested with similarity weightings and cosine distance [4]. Furthermore, other interesting graphical methods have been developed that were not considered in this paper [7].

References

[1] J. Leskovec, L. Adamic, and B. Huberman. The Dynamics of Viral Marketing. 2005.

[2] Ullman, Jefferey David. "9 Recommendation Systems." *Mining of Massive Datasets*. By Anand Rajaraman. N.p.: Cambridge UP, 2011. N. pag. Print.

[3] T. Zhou, J. Ren, M. Medo, and Y-C. Zhang. *Bipartite Network Projection and Personal Recommendation.* The American Physical Society. 2007.

[4] Y. Koren, R. Bell, and C. Volinsky. *Matrix Factorization Techniques for Recommender Systems. Computer* 42.8 (2009): 30-37.

[5] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown, and D. Botstein. *Imputing Missing Data for Gene Expression Arrays.* Stanford University, 1999.

[6] B. W. Lewis. *The Irlba Package*. Program documentation. *Http://illposed.net/irlba.pdf*. N.p., 27 May 2011.

[7] X. Li, and H. Chen.*Recommendation as Link Prediction in Bipartite Graphs: A Graph Kernel-based Machine Learning Approach.* ScienceDirect.com. 2012.