Advanced Bayesian Methods for Graph-based Recommendations

Aditi Mavalankar[1], Stephanie Chen[2] and Ajitesh Gupta[3]
University of California, San Diego
December 8, 2017

Abstract— **Previous graph-based recommender systems have been proven to generate good quality recommendations, but have the drawback of high computational complexity. This paper attempts to replicate the work of an existing graph-based recommendation system that employs a probabilistic approach to solve this problem purely based on ratings given to items, and also extends it to incorporate user behavioural patterns into it. The results of including user-user and item-item interactions exhibit an increase in the quality of recommendations generated.**

I. INTRODUCTION

As the number of applications using recommender systems increases, the quality of each recommendation matters more with respect to user satisfaction, as well as merchant revenue. There are several existing techniques used in recommender systems employed by Amazon, Yelp, Netflix, etc. such as one-class collaborative filtering[7], matrix factorization[5], clustering, deep learning and graph-based recommendation.

Graph-based recommendation systems exploit the structure of the graph generated by forming edges between user and item nodes, based on their interaction in terms of rating. They are different from traditional methods in that they do not involve computations involving large matrices, which could be computationally intensive, in case of higher number of users and/or items. For this reason, we extend the paper 'Efficient Bayesian Methods for Graph-based Recommendation'[6]. The authors presented a novel graph-based method to solve general purposed recommendation tasks. Most of the existing graph-based models make use of random walks in order to produce recommendations. These are highly inefficient due to the large number of transition matrices involved leading both high space and time complexity. Cooper et al [1] proposed to do fixed length random walks in order to approximate transition probabilities, but their methods also suffer problems as it can take large number of iterations to converge. Thus the authors proposed a whole different approach in this paper in order to avoid all those problems by evaluating fixed length paths.

II. Original Model Overview

The model proposed in the paper iterates through all 3-step paths present in the graph, where each user and item in the dataset is represented as a node in the graph. Connections between nodes exist if a user has rated an item, or if an item has received a rating from a user. For instance, in Fig.2 the user U1 is our target user. We see that he has bought items I1 and I3. Users U2 and U3 have bought items I1 and I2 while user U4 and U4 has bought items I3 and I4. As such 3 length paths exist between target user U1 and potential recommendations I2 and I4. The labels on the path from U1 to I1, from I1 to U2 and from U2 to I2 indicate one such path. This is the way all neighbors of U1 are evaluated to see if they have bought items that have not been bought by U1, with the neighborhood being defined as the group of users who have bought the same items as U1.

While going through all such 3-step paths, the authors score each item by making use of a probability distribution functions defined over the ratings given to each item. In order to create the scoring functions the authors first define the notion of reliability score of an item. They define a binary random variable such that

$$Y_j = \begin{cases} 0 & \text{if item j assessed positively} \\ 1 & \text{if item j assessed negatively} \end{cases}$$

Using this random variable they defined the reliability of the item as

$$P(Yj = 1) = \theta_j$$

which they model as a Beta distribution. Since beta distribution is a conjugate distribution, the prior is also a beta distribution. Hence

*This work extends the RecSys-2016 paper dl.acm.org/citation.cfm?id=2959132

[1]Aditi Mavalankar `amavalan at eng.ucsd.edu`
[2]Stephanie Chen `syc078 at eng.ucsd.edu`
[2]Ajitesh Gupta `ajgupta at eng.ucsd.edu`

$$P(\theta_j = 1|Ratings) = Beta(R^+, R^-)$$

where
R^+ = Number of positive ratings for the item
R^- = Number of negative ratings for the item

Using these notions the authors defined 3 different scoring functions. We'll make use of Fig.1 to understand the functions. There Item X is clearly the item in consideration to be recommended to the Target user U.

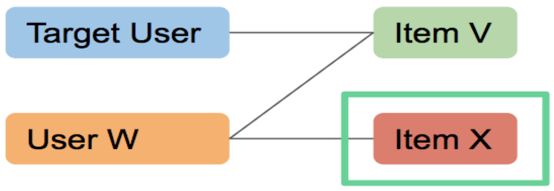


Fig. 1: Sample 3-step path from user to item.

1) Posterior Inequality Scoring - The authors this as a measure of the probability of reliability of item X being higher than the reliability of item V. The value for that can be calculated using integration over the beta distribution.

$$PIS(u, v, w, x) = P(\theta_x > \theta_v|R_x, R_v)$$

2) Posterior Prediction Scoring - This measures the probability of both items X and V being assessed positively, while considering their RVs to independent of each other.

$$PPS(u, v, w, x) = P(Y_x = 1|R_x) * P(Y_v = 1|R_v)$$

The probability of reliability of individual items being 1 comes out to the following equation -

$$P(Y_x = 1|R^+, R^-) = \frac{R^+ + 1}{R^+ + R^- + 1}$$

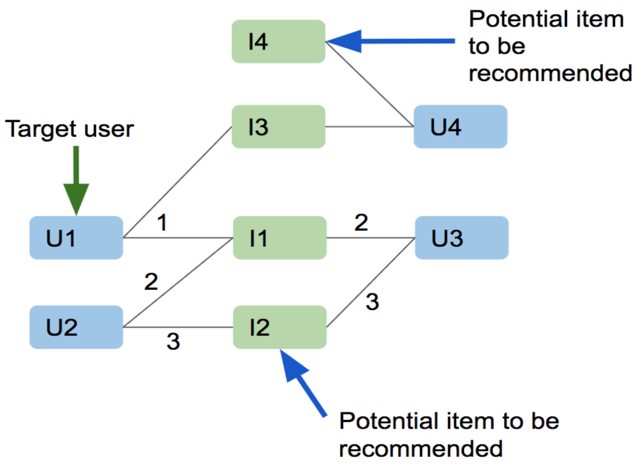
3) Posterior Odds Ratio Scoring - This measure represents how large the odds of x receiving a positive assessment is when compared to the odds of v receiving a positive assessment. It can defined as follows -

$$PORS(u, v, w, x) = \frac{\phi_x}{\phi_v}$$

, where

$$\phi_i = \frac{P(Y_i = 1|R_x)}{P(Y_i = 0|R_x)}$$

is the odds of item i receiving a positive assessment.



I4
Potential item to be recommended
Target user
I3
U4
U1
1
I1
2
U3
2
3
U2
3
I2
Potential item to be recommended

Fig. 2: Potential recommendations for a user

III. REPRODUCING THE PAPER'S RESULTS

The original paper utilized seven datasets on items ranging from books to movies to Amazon products. From these, we chose to use only the FilmTrust dataset[2] given several factors: (1) the computational constraints of computing similarities between all pairs of users as the dataset size increased, (2) the difficulty of reproducing another paper's results without their code, and (3) our time constraints in a 10-week academic quarter.

The FilmTrust dataset[2] contains 35,497 movie ratings from 1,508 users on 2,071 movies, all crawled in June 2011. In contrast, the other datasets had at least 1 million ratings each, making FilmTrust relatively small and more manageable.

As done in the original paper, we removed users who had given fewer than 10 ratings, and items that had received less than 10 ratings. This was in order to avoid the cold start problem, which the authors did not aim at solving. So, from the original 35,000 movie ratings, we ended with 28,317 movie ratings, which is almost exactly the same as that reported by the authors. We then split the data randomly into a training set and a test set with an 80:20 split.

We also conducted the experiments on MovieLens-100k dataset[4]. It contains 100,000 ratings from 943 users for 1682 movies. With each user having rated atleast 20 movies each, we did not have to worry about cold-start issues.

The next step was to implement the three scoring techniques proposed by the authors to generate a score for each candidate item:

1) Posterior Inequality Scoring (PIS)
2) Posterior Prediction Storing (PPS)
3) Posterior Odds Ratio Scoring (PORS)

For us, PPS shows the best results amongst the three scoring functions, which is a conclusion in line with the authors' findings.

We then calculated the four metrics the authors used for the evaluation criteria:

1) Precision@k (P@k)
2) Mean Average Precision (MAP)
3) Mean Reciprocal Rank (MRR)
4) Normalized Discounted Cumulative Gain (NDCG)

For the same baseline method the authors used, we were actually able to achieve a higher MRR and NDCG than they did. For Precision@k and MAP, our results matched those of the original paper.

The overall trends that we obtained in our implementation matched those of the original paper.

IV. PROPOSED EXTENSIONS

Now that the trends in performances of all three scoring functions in the original paper had been replicated, we decided to study factors other than the ratings given to each item. There were 3 major factors we studied - user-user similarity, item-item similarity and user reliability.

A. *User similarity score*

The original paper does not consider using any kind of information about the users and exclusively works in terms of items, hence we decided that the first direction to look at would be to somehow incorporate user information into the algorithm. For this we decided to compute similarity scores for pairs of users by computing how many similar ratings they had among the common set of items rated by them. We believe that this similarity would be useful because users who rate items the same way might share the same preferences and hence information about preferences of one such user may help us give better recommendations to users highly similar to him. This basically measures the degree of agreement in opinions and preferences of the two users.

B. *Item similarity score*

In the original paper the authors make use of reliability scores of item ratings to generate scores for each item to be recommended. They do not make use of binary interactions between items explicitly as such, hence much like user similarity we decided to use the given data to find item similarity. Analogous to the user similarity we computed the similarity for a pair of items by computing the number of similar ratings the pair of items got from their common set of users. For each similar rating by a user for both items, we increase the similarity score and for each dissimilar rating by a user we decrease the similarity score. As mentioned before, this similarity give us a way of using item information while making recommendations. We got inspiration for this by seeing similar ideas being used in ItemRank[3].

C. *Reliability of Users*

We came up with another method to incorporate user information into the algorithm, by considering the number of items that have been rated by that user. The intuition behind using this information is that the users who have rated more items have more experience and thus might make more informed decisions. So e.g. a user that has rated a 1000 items might be more knowledgeable than a user who has rated 10 items and as such we would regard his opinion with more importance. As such we define reliability of a user as the number of items rated by that user. This is also different from the previously proposed user similarity metric as this is a unary metric and only depends on the user we are looking to draw potential recommendations from. The user similarity metric on the other hand is binary and takes into account both the target user and the similar user.

V. IMPLEMENTING EXTENSIONS

A. *Calculating user similarity scores*

In order to assign a score to the similarity between each pair of users, we look at the items they have rated in common. As shown in Algorithm 1, we increment the similarity score by one, each time an item rated in the same manner (favorably/unfavorably) by both users is encountered. Conversely, we decrement the similarity score by one, each time an item rated differently by the two users is seen. We then divide this similarity score by the total number of common items rated, to get a mean similarity score. Since the extreme cases consist of two users rating all their common items completely similarly, or differently, the mean similarity

Algorithm 1: FindUserSimilarity (R, U, I)

```
Input  : Rating map $R$, User list $U$, Item list $I$
Output: User similarity matrix $U$
1  $user\_similarity \leftarrow matrix\ of\ size|\mathrm{U}| \times |\mathrm{U}|$
2  for $each\ user\ pair(u_1, u_2)$ do
3      $similarity = 0$
4      for $each\ item\ i\ \epsilon\ R(u_1)\ \cap\ R(u_2)$ do
5          if $R(u_1,i)\ equals\ R(u_2,i)$ then
6              $similarity++$
7          else
8              $similarity--$
9          end
10     end
11     $user\_similarity\,[u_1, u_2] =\ similarity\ /\ |R\,(u_1)\ \cap\ R(\,u_2)|$
12     $user\_similarity\,[u_2, u_1] = user\_similarity\,[u_1, u_2]$
13 end
14 # Normalization
15 for $each\ \ (u_1, u_2)\,in\ user\_similarity$ do
16     $user\_similarity\ \ [u_1, u_2] =\ (user\_similarity\ \ [u_1, u_2] +\ 1)\ /\ 2$
17 end
18 return *user_similarity*
```

score lies between -1 and 1. Qualitatively, this score should signify the probability of two users rating an item similarly. Thus, we need to normalize this score to lie between 0 and 1. We do this by employing the following formula for each user-user similarity score.

$$\text{normalized\_score} = \frac{\text{score} - \text{score}_{min}}{\text{score}_{max} - \text{score}_{min}}$$

Intuitively, this matrix will be a symmetric matrix, and so, the values of similarity of (u_1, u_2) will be the same as that of (u_2, u_1).

In order to incorporate this user similarity into the existing scoring functions, as indicated in Algorithm 3, we multiply the scores generated according to the original algorithm with the user-user similarity for the two users in the 3-step path. This modified value is then added to the recommendation score for that particular item.

B. Calculating item similarity scores

We calculated item similarity scores in the same fashion to our strategy for user similarity scores. In order to assign a score to the similarity between two items, we looked at which (if any) users had given both of those two items a rating, be it positive or negative. As described in Algorithm 2, we started with the rating map, user list, and item list. The algorithm fills and returns an item similarity matrix of size $|I|$x$|I|$, where I is the length of the item list. This item similarity matrix will contain the similarity scores for each pair of items. To fill this matrix, we calculate the similarity score for every pair of items.

Taking in U', which contains all users who have rated this particular pair (call it item 1 and item 2), we iterate over each user $u\ \epsilon\ U'$. If u gave the same rating to item 1 and item 2, then the similarity increments. If a different rating were given to each item in this pair, then the similarity decrements. Once we have iterated over all the users in U', we then calculate the item similarity score for each pair of items by taking its similarity and dividing it by the average rating given by the users who rated those items. This is essentially taking the average of the ratings for these items. And, because the item similarity matrix is symmetric, we assign the same score for $(i_1,\ i_2)$ to the cell for $(i_2,\ i_1)$.

We then normalize the item similarity scores to lie between 0 and 1, following the same procedure employed for the user similarity scores described previously.

C. Calculating User Reliability

User reliability scores for each user can be very simply calculated by counting the number of items

```
Algorithm 2: FindItemSimilarity $(R, U, I)$
Input  : Rating map $R$, User list $U$, Item list $I$
Output: Item similarity matrix $U$
1  $item\_similarity \leftarrow matrix\ of\ size$ |I| × |I|
2  for $each\ item\ pair(i_1, i_2)$ do
3      $similarity = 0$
4      $U' = \{u \in U \mid (u, i_1) \in R \,\&\, (u, i_2) \in R\}$
5      for $each\ user\ u \in U'$ do
6          if $R(u, i_1)\ equals\ R(u, i_2)$ then
7              $similarity++$
8          else
9              $similarity--$
10         end
11     end
12     $item\_similarity\ [i_1, i_2] = similarity \div |U'|$
13     $item\_similarity\ [i_2, i_1] = item\_similarity\ [i_1, i_2]$
14 end
15 # Normalization
16 for $each\ (i_1, i_2)\ in\ item\_similarity$ do
17     $item\_similarity\ [i_1, i_2] = (item\_similarity\ [i_1, i_2] + 1) \div 2$
18 end
19 return item_similarity
```

```
Algorithm 3: Calculate Item Scores
Input  : User $u_1$, Item $i_1$, User $u_2$, Item $i_2$, where $i_2$ is the candidate for recommendation to $u_1$
Output: List of all scores calculated to determine suitability of recommending $i_2$ to $u_1$
1  for $each\ 3-step\ path(u_1, i_1, u_2, i_2)$ do
2      $original\_score\,[i_2] \mathrel{+}= scoring\_function\,(u_1,\ i_1,\ u_2,\ i_2)$
3      $USS\_score\,[i_2] \mathrel{+}= original\_score * user\_similarity\,[u_1,\ u_2]$
4      $ISS\_score\,[i_2] \mathrel{+}= original\_score * item\_similarity\,[i_1,\ i_2]$
5      $USS\_ISS\_score\,[i_2] \mathrel{+}= original\_score * user\_similarity\,[u_1,\ u_2] * item\_similarity\,[i_1,\ i_2]$
6      $rel\_score\,[i_2] \mathrel{+}= original\_score * reliability\,[u_2]$
7      $rel\_USS\_score\,[i_2] \mathrel{+}= rel\_score * user\_similarity\,[u_1,\ u_2]$
8      $rel\_ISS\_score\,[i_2] \mathrel{+}= rel\_score * item\_similarity\,[i_1,\ i_2]$
9      $rel\_USS\_ISS\_score\,[i_2] = rel\_score * user\_similarity\,[u_1,\ u_2] * item\_similarity\,[i_1,\ i_2]$
10 end
```

rated by each user and then normalizing it by dividing by the maximum such number.

$$Rel(U_i) = \frac{NumberOfItemsRated_i}{\max_{U_k}(NumberOfItemsRated_k)}$$

VI. RESULTS

A. User Similarity and Item Similarity

As seen in Table I, incorporating the user similarity and item similarity separately into the scoring function leads to an improvement over the original results. Furthermore, when we include both these similarities in the scoring function, the performance improves in most cases, compared to including them individually or not including them at all. This goes on to show that both these features contribute to an improved

Results for Filmtrust dataset							
Scoring	Method	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
PIS	Original	0.4042	0.5273	0.2904	0.2479	0.4601	0.5344
	+USS	0.4033	0.5251	0.2906	0.2476	0.4601	0.5339
	+ISS	0.4079	0.5322	0.2913	0.2493	0.4633	0.5388
	+USS+ISS	0.4083	0.5323	0.2910	0.2493	0.4635	0.5392
	+REL	0.3957	0.5192	0.2799	0.2466	0.4434	0.5243
	+USS+REL	0.3950	0.5185	0.2813	0.2468	0.4440	0.5242
	+ISS+REL	0.3998	0.5232	0.2841	0.2483	0.4503	0.5298
	+USS+ISS+REL	0.4000	0.5240	0.2851	0.2486	0.4491	0.5287
PORS	Original	0.4121	0.5315	0.2984	0.2534	0.4728	0.5487
	+USS	0.4124	0.5320	0.2991	0.2530	0.4729	0.5484
	+ISS	0.4149	0.5338	0.3014	0.2540	0.4787	0.5529
	+USS+ISS	0.4163	0.5359	0.3037	0.2537	0.4795	0.5525
	+REL	0.4012	0.5185	0.2828	0.2519	0.4459	0.5326
	+USS+REL	0.4003	0.5174	0.2826	0.2518	0.4454	0.5319
	+ISS+REL	0.4023	0.5185	0.2870	0.2525	0.4522	0.5355
	+USS+ISS+REL	0.4026	0.5192	0.2885	0.2523	0.4529	0.5356
PPS	Original	0.4247	0.5410	0.3087	0.2571	0.4972	0.5723
	+USS	0.4280	**0.5451**	0.3085	**0.2574**	0.4984	**0.5742**
	+ISS	0.4258	0.5410	0.3083	0.2570	0.4976	0.5720
	+USS+ISS	**0.4284**	0.5431	**0.3104**	**0.2574**	**0.4994**	0.5738
	+REL	0.4144	0.5312	0.2980	0.2546	0.4760	0.5552
	+USS+REL	0.4163	0.5350	0.2984	0.2546	0.4770	0.5554
	+ISS+REL	0.4145	0.5315	0.3037	0.2538	0.4804	0.5535
	+USS+ISS+REL	0.4152	0.5326	0.3014	0.2544	0.4771	0.5528

TABLE I: The results obtained for all methods on FilmTrust Dataset.

Results for ML-100k dataset							
Scoring	Method	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
PIS	Original	0.1459	0.4173	0.2049	0.1654	0.2482	0.2310
	+USS	0.1472	0.4209	0.2049	0.1667	0.2486	0.2319
	+ISS	0.1476	0.4266	0.2023	0.1653	0.2464	0.2307
	+USS+ISS	0.1479	0.4264	0.2023	0.1657	0.2465	0.2309
	+REL	0.1253	0.3726	0.1739	0.1478	0.2102	0.1959
	+USS+REL	0.1264	0.3751	0.1758	0.1487	0.2123	0.1972
	+ISS+REL	0.1251	0.3680	0.1739	0.1468	0.2082	0.1936
	+USS+ISS+REL	0.1261	0.3680	0.1758	0.1485	0.2098	0.1953
PORS	Original	0.1147	0.2949	0.1525	0.1330	0.1694	0.1643
	+USS	0.1149	0.2931	0.1529	0.1326	0.1693	0.1639
	+ISS	0.1188	0.3054	0.1540	0.1372	0.1737	0.1704
	+USS+ISS	0.1195	0.3038	0.1559	0.1384	0.1751	0.1716
	+REL	0.1011	0.2466	0.1419	0.1207	0.1510	0.1450
	+USS+REL	0.1012	0.2447	0.1419	0.1212	0.1502	0.1447
	+ISS+REL	0.1040	0.2564	0.1421	0.1261	0.1535	0.1506
	+USS+ISS+REL	0.1045	0.2561	0.1423	0.1265	0.1527	0.1502
PPS	Original	0.1531	0.4213	0.2102	0.1724	0.2546	0.2410
	+USS	0.1546	0.4235	**0.2106**	**0.1743**	**0.2554**	**0.2432**
	+ISS	0.1546	0.4304	0.2095	0.1727	0.2544	0.2412
	+USS+ISS	**0.1558**	**0.4317**	0.2076	0.1735	0.2534	0.2415
	+REL	0.1291	0.3660	0.1745	0.1494	0.2126	0.1994
	+USS+REL	0.1311	0.3735	0.1767	0.1519	0.2146	0.2021
	+ISS+REL	0.1297	0.3659	0.1784	0.1508	0.2121	0.1987
	+USS+ISS+REL	0.1312	0.3681	0.1775	0.1513	0.2124	0.1996

TABLE II: The results obtained for all methods on ML-100k Dataset.

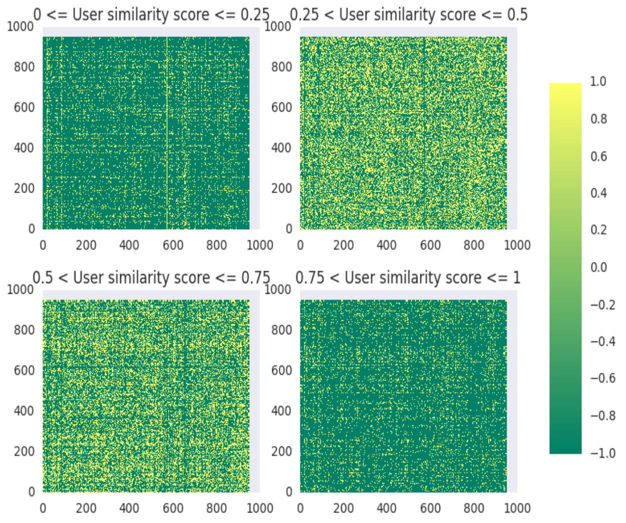


Fig. 3: Heat map of user similarity - FilmTrust

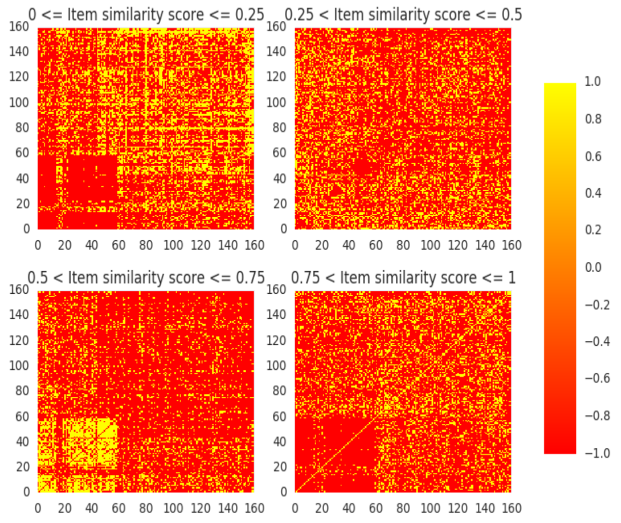


Fig. 4: Heat map of item similarity - FilmTrust

recommendation quality.

To analyze why these similarity matrices give better results, we look at the heat map of the user similarities in Fig. 3. This heat map represents the user similarities that fall in specific brackets. Pairs of users who are highly dissimilar to each other are highlighted in the first subplot; those who are mildly dissimilar to each other are highlighted in the second plot. The users who are moderately similar to each other are highlighted in the third plot and those highly similar to each other are highlighted in the fourth plot. The item similarity heat map follows the same analogous structure for items.

We see that the majority of users are neither too similar nor too dissimilar to each other. This makes the remaining user pairs special i.e. they can be used to generate recommendations with a stronger confidence. On the other hand, item similarity scores, shown in Fig. 4, seem to be evenly spread out, making it more difficult to understand their influence in improving the results over the baseline performance. The same analysis can be extended to the user and item similarity heat maps for the ML-100k dataset.

B. *Precision@k*

Precision@k refers to the number of relevant items in the list of top k items recommended. The values of precision@k for $k = 5$ and $k = 10$ for all combinations of methods used, are plotted in Fig. 7. For P@5, we can see that PPS, combined with the user similarity and item similarity scores, outperforms all other methods. For P@10, we see that PPS combined with user similarity gives almost the same performance

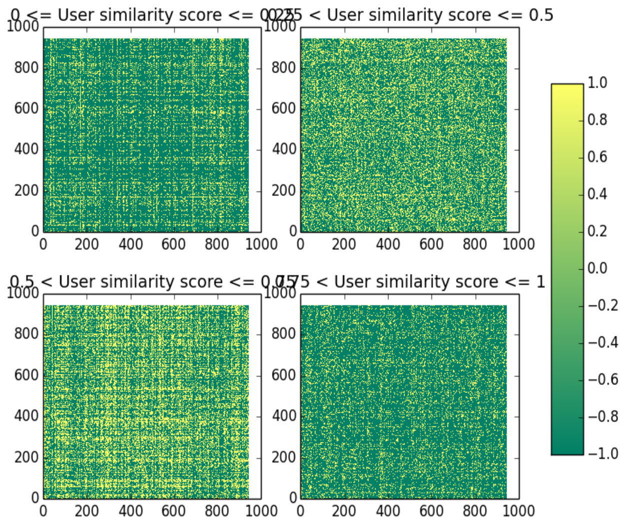


Fig. 5: Heat map of user similarity - ML-100k

as PPS combined with both user and item similarities.

If we consider the results on the ML-100k dataset, we see that the highest precision value for both $k = 5$ and $k = 10$ is obtained when PPS combined with user similarity scores, is used for ranking. The results on both these datasets indicate that user similarity is an important addition to the existing scoring functions.

C. *Mean Average Precision*

Mean average precision is obtained by averaging the precision@k values for each $k \in$ {positions of relevant documents}. We can see in Fig. 8 that the highest performance in terms of mean average precision is exhibited by the PPS combined with both the user similarity and item similarity scores. This is closely

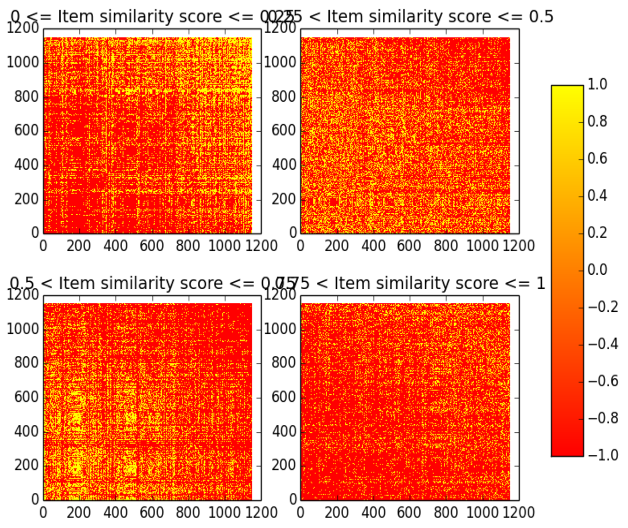


Fig. 6: Heat map of item similarity - ML-100k

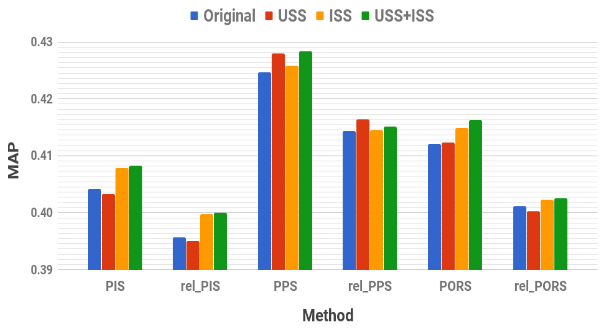


Fig. 8: Mean Average Precision (MAP) - FilmTrust

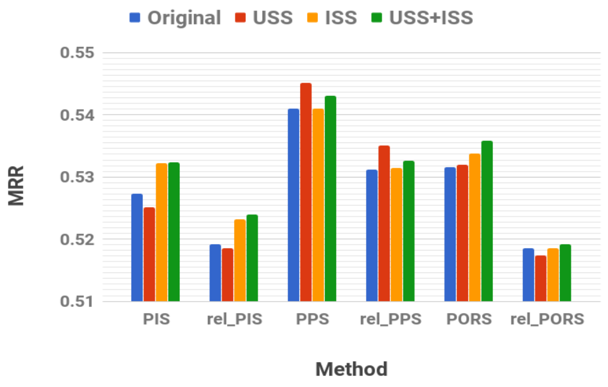
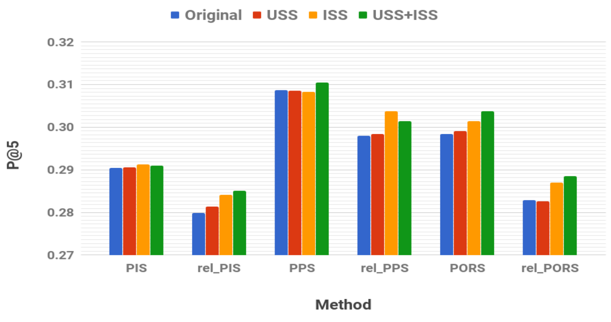
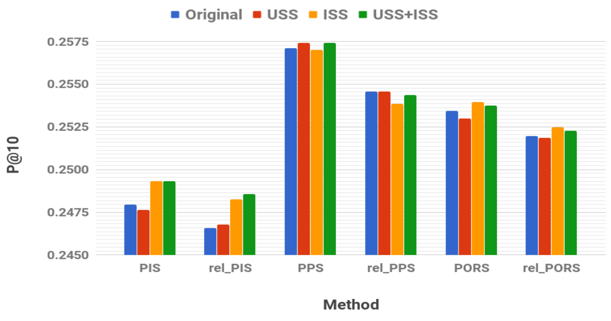


Fig. 9: Mean Reciprocal Rank (MRR) - FilmTrust



(a) Precision@5 - FilmTrust



(b) Precision@10

Fig. 7: Precision@10 - FilmTrust

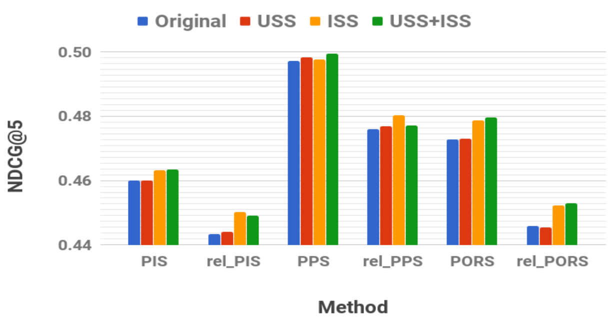
followed by PPS combined with only the user similarity score. The top performer in the ML-100k dataset is PPS combined with user and item similarities. This result again strengthens the importance of user similarity in boosting the quality of predictions.

D. Mean Reciprocal Rank

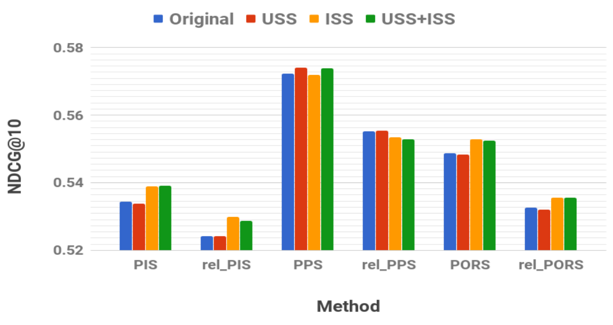
Reciprocal rank refers to the inverse of the position of the first relevant item in the list of recommended items. The average of the reciprocal rank across all users gives the mean reciprocal rank. We see in Fig. 9 that the PPS combined with the user similarity scores is the best performer in terms of MRR, and the second best performer is PPS combined with user and item similarities. In the ML-100k dataset, PPS combined with user and item similarities gives the best performance.

E. Normalized Discounted Cumulative Gain

Discounted Cumulative Gain measures the usefulness of the predictions in terms of the ranks of items in the predicted list. The lower a relevant item appears in the list, the lower its usefulness is and the lesser information it gives. Normalized discounted cumulative



(a) Normalized Discounted Cumulative Gain@5 - FilmTrust



(b) NDCG@10

Fig. 10: Normalized Discounted Cumulative Gain@10 - FilmTrust

gain (NDCG) as such measures the ratio of the gain in information between the predicted ranking and ideal rankings of the items in the predicted list. So As we can see in Fig. 10, for both values $k = 5$ and $k = 10$, the best performers are PPS combined with user similarity, and PPS combined with both user and item similarities. For the ML-100k dataset, for both $k = 5$ and $k = 10$, PPS combined with user similarities beats all other methods.

F. *Overall Analysis*

As is evident in the individual evaluation for all the metrics listed above, PPS remains the best scoring function out of the three scoring functions suggested in the original paper. This is in synchronization with the results of the original paper. Furthermore, our extensions have shown that user similarity is an important factor that boosts the performance in terms of all the metrics on both datasets. Item similarity individually does improve the quality of recommendations, but never beats the user similarity. Furthermore, combining these two similarities sometimes leads to a higher quality of recommendations than incorporating them individually into the scoring functions. (Plots for ML-100k have not been shown due to insufficient space.)

VII. FUTURE WORK

Since the algorithms for calculating user and item similarities are almost trivial, it could be a good idea to implement different algorithms to calculate them. Additionally, ratings alone may not be the best data on which to base this similarity score. We suggest using the timestamp associated with each rating to generate potentially better recommendations. Another major extension could be using the raw ratings (on a scale of 1 through 5) instead of reducing it to binary form.

VIII. CONCLUSION

Through the process of examining the original paper, reproducing its findings, implementing our three extension ideas, analyzing the results, and giving careful consideration to the ways our methods could be improved, we discovered several findings:

- User-user similarity is observed to be more useful than item-item similarity.
- Introducing either kind of similarity, be it user or item, improves the quality of recommendations.
- User reliability score seems to be too naive in that including it in our method actually reduced the quality of recommendations.
- PPS still remains the top performer among all of the scoring techniques.

Since the results we found are consistent on the FilmTrust dataset, as well as on the ML-100k dataset, we believe it is safe to say that similar results would be exhibited on the remaining five datasets used in the original paper.

APPENDIX

The code for our implementation of the original paper, as well as our extensions, can be found at github.com/syychen/fast_bayesian_graph_recommender

ACKNOWLEDGMENT

We would like to thank Prof. Julian McAuley for guiding us throughout the project.

REFERENCES

[1] Colin Cooper et al. "Random walks in recommender systems: Exact computation and simulations". In: *Proceedings of the 23rd International Conference on World Wide Web*. ACM. 2014, pp. 811–816.

[2] J. Golbeck and J. Hendler. "FilmTrust: movie recommendations using trust in web-based social networks". In: *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.* Vol. 1. 2006, pp. 282–286. DOI: 10.1109/CCNC.2006.1593032.

[3] Marco Gori and Augusto Pucci. "ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines." In:

[4] F Maxwell Harper and Joseph A Konstan. "The movielens datasets: History and context". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5.4 (2016), p. 19.

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. *MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS*. 2009.

[6] Ramon Lopes, Renato Assunção, and Rodrygo L.T. Santos. "Efficient Bayesian Methods for Graph-based Recommendation". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys '16. Boston, Massachusetts, USA: ACM, 2016, pp. 333–340. ISBN: 978-1-4503-4035-9. DOI: 10.1145/2959100.2959132. URL: http://doi.acm.org/10.1145/2959100.2959132.

[7] Rong Pan et al. "One-Class Collaborative Filtering". In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 502–511. ISBN: 978-0-7695-3502-9. DOI: 10.1109/ICDM.2008.16. URL: http://dx.doi.org/10.1109/ICDM.2008.16.