Shared Collaborative Filtering

Yu Zhao
NEC Laboratories China
Beijing, China
zhao_yu@nec.cn

Xinping Feng *
CNNIC
Beijing, China
fxinping@gmail.com

Jianqiang Li, Bo Liu
NEC Laboratories China
Beijing, China
{li_jianqiang,liu_bo}@nec.cn

ABSTRACT

Traditional collaborative filtering (CF) methods suffer from sparse or even cold-start problems, especially for new established recommenders. However, since there are now quite a few recommender systems already existing in good working order, their data should be valuable to the new-start recommenders. This paper proposes shared collaborative filtering approach to leverage the data from other parties (contributor party) to improve own (beneficiary party's) CF performance, and at the same time the privacy of other parties cannot be compromised. Item neighborhood list is chosen as the shared data from the contributor party with considering differential privacy. And an innovative algorithm called neighborhood boosting is proposed to make the beneficiary party leverage the shared data. MovieLens and Netflix data sets are considered as two parties to simulate and evaluate the proposed shared CF approach. The experiment results validate the positive effects of shared CF for increasing the recommendation accuracy of the beneficiary party. Especially when the beneficiary party's data is quite sparse, the performance can be increased by around 10%. The experiments also show that shared CF even outperforms the methods that incorporate the detailed original rating scores of the contributor party without considering the privacy issues. The proposed shared CF approach obtains a win-win situation for both performance and privacy.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

*This author contributed to this paper during his internship at NEC Labs China.

RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.

Keywords

collaborative filtering, recommender, differential privacy, neighborhood boosting

1. INTRODUCTION

As the core technology adopted in current popular recommendation applications, collaborative filtering (CF) predicts the interestingness of items to a particular user based on the already known interestingness of items to other users, or called community data. The underlying assumption of CF approach is that personal tastes are correlative and the users who have the similar interests in the past tend to agree again in the future.

Since CF relies on pre-existing community data to start work, a pure CF approach has difficulty to handle sparse or even cold-start problems, i.e. the scenarios without enough initial community data. For tackling such shortcoming, some researchers endeavor to improve or propose new CF algorithms, such as using item-based approaches instead of use-based [8], or applying Latent Semantic Indexing (LSI) [7]. And the other methods turn to leverage other information besides collaborative information, such as hybrid methods which combine content and collaborative data [13].

In this paper, we propose shared CF approach to alleviate sparse problem. Shared CF means CF based not only on own community data, but also on data shared by other parties. Since there are now quite a few recommender systems already existing in good working order, leveraging the data of such existing systems should be valuable to the new-start recommenders. However, since shared CF is a type of multi-party data processing, it raises a new critical problem, privacy issue. Generally, for a party with community data, the privacy refers to the identities of the involved users. Whether the data privacy could be preserved is an essential concern for each party to participate in multi-party collaboration. At the same time, for shared CF, if some privacy preservation techniques adopted, the advantage of using other parties' data should be also kept, i.e., the CF accuracy should be improved comparing with using own data only. Shared CF is actually a trade-off problem to balance the performance and privacy preservation.

Research on privacy-preserving multi-party CF has been carried by several researchers from the community of privacy-preserving data mining, such as [6]. However, their target problem is different from ours. They assume that a whole bulk of community data is distributed or partitioned into multiple parties due to the privacy concerns. And their goal is to preserve the privacy of each party, and at the same time

to achieve decent accuracy. All parties are peer-to-peer during processing. Whereas, for shared CF, the scenario is that a party (beneficiary party) leverages the data shared from other independent parties (contributor parties) to improve its own CF accuracy. The two kinds of parties may be not peer during processing. Due to the difference of the goals, the existing multi-party CF work did not consider how to maximize the utilization of other parties' data, with the constraint of privacy preservation.

Comparing with existing multi-party CF, shared CF could have more potential to be applied in practical business. Here we give two possible business application scenarios. (1) The mature recommender systems with rich data could publish or sell their data to the public, and other systems could buy and utilize such data to improve their own performance. The privacy of the data provider needs to be preserved. (2) When a site joins in a commercial league, e.g., a new chain store launches, the league organizer can afford the data of other league members to this new site for speeding up its recommender system. Since each league member is autonomous, its data privacy should be preserved.

Obviously, for a specific beneficiary, not all data contributors could definitely make positive contribution. Intuitively, if item set (or user set) of a contributor overlaps with that of the beneficiary, the data from the contributor might be valuable for shared CF. Due to the privacy concerns, to judge whether the user sets of two parties overlap is quite impractical. On the other side, it is a common phenomenon that different parties have the similar item set, such as that movie items of the most movie sites are identical. Thus, in this paper, we concentrate on the case that the item sets of different parties overlap each other. Moreover, without loss of generality, we assume the item sets of different parties are the same, since for overlapping case, we could only leverage the overlapped part of the items.

In this paper, we propose an item-neighborhood based shared CF approach. The contributor shares the neighborhood information about the items, which is acquired based on its own community data, and at the same time to keep its data privacy. And an item neighborhood boosting algorithm is proposed for beneficiary party to leverage such shared information from the contributor.

2. PROBLEM DEFINITION

Shared CF problem includes two participant parties: beneficiary party (or called party A in the following), and contributor party (or called party B). Their user sets are annotated by $U_A = \{u_{A1}, \cdots, u_{Ama}\}$ and $U_B = \{u_{B1}, \cdots, u_{Bmb}\}$. As mentioned above, we only consider the case that Party A and B have the same item set, annotated by $I = \{i_1, \cdots, i_n\}$. Within each party, the opinion of a specific user u on a specific item i is represented by a rating score r^A_{ui} or r^B_{ui}. If user u has not rated item i, r^A_{ui} (or r^B_{ui}) $= \Phi$. All the rating scores in earch party form a user-item rating matrix, with each row representing a user and each column representing an item. The two parties' rating matrixes are denoted as M_A and M_B. Shared CF task is defined as: to predict the missing rating score of M_A with the existing scores of both M_A and M_B, while preserving the data privacy of party B. Here, we set the privacy of Party B as the content of U_B and M_B, i.e., the user information. Additionally, besides to design a privacy-preserving prediction algorithm, another goal is to assure that the prediction accuracy of shared CF is better than the accuracy of using the existing scores of M_A only.

3. SHARED CF ALGORITHMS

To decouple the two parties to make shared CF more practical and flexible, we divide the shared CF task into a two-step process: (1) Party B generates a privacy-preserving shared data set based on its own community data M_B. (2) Party A leverages the shared data from Party B together with Party A's own community data M_A to make prediction of the missing scores of M_A.

3.1 Step 1: Item Neighborhood Sharing

Since user related information is sensitive for Party B, to publish Party B's user-only information or user-item relation information is obviously risky. While, item-only information seems to be safe for publishing. On the other hand, memory-based CF can leverage item similarity for recommendation. Therefore, either from Party B's perspective of privacy, or from Party A's perspective of data utility, to publish item similarity relevant information from Party B should be a good choice.

In the proposed solution, Party B generates and publishes each item's neighborhood list as its shared data set. For each item pair $i, j \in I$, based on the similarity analysis with the existing scores of M_B, the similarity between i and j is denoted as $s_B(i, j)$, and larger $s_B(i, j)$ value represents that i and j are more similar. The $k(k < n)$ nearest neighborhood (most similar) items of i in the context of M_B is denoted as B_i. Then, Party B publishes a set of B_i for each i in I, and such shared item neighborhood data set is denoted by $B = \{B_1, \cdots, B_n\}$. The detailed privacy issue of publishing item neighborhood list will be discussed in Section 5.

The item similarity analysis can be conducted by algorithms used in traditional CF. We adopt two kinds of similarity analysis, which come from traditional memory-based and model-based CF approaches, respectively. The performance for shared CF would be compared with experiments.

- Pearson correlation based similarity

Pearson correlation is the most popular similarity measure for memory-based CF [9]. The set of users who have already rated item i is denoted by $C_i = \{u | u \in U, r_{ui} \neq \Phi\}$. The average rating score of i by users of C_i is denoted as $\overline{r_i}$. Then, for a pair of item (i, j), their Pearson correlation is:

$$c(i,j) = \frac{\sum_{u \in C_{ij}} (r_{ui} - \overline{r_i})(r_{uj} - \overline{r_j})}{\sqrt{\sum_{u \in C_{ij}} (r_{ui} - \overline{r_i})^2 \sum_{u \in C_{ij}} (r_{uj} - \overline{r_j})^2}} \quad (1)$$

where C_{ij} stands for the set of users who have already rated both item i and j, i.e., $C_{ij} = C_i \cap C_j$.

- Latent semantic model based similarity

As a powerful model-based CF approach, Latent Semantic Analysis (LSA) could transfer both users and items into a more compact and meaningful latent semantic space [3]. Here, we leverage LSA to calculate the item similarity within such latent semantic space. We adopt Singular Value Decomposition (SVD) based LSA. A rating matrix $M_{m \times n}$ is firstly preprocessed with the method mentioned by [11]: Compute the average of each row (user), and the average of

each column (item); replace all missing values with the corresponding column (item) average, then get a new filled-in matrix; subtract the corresponding row (user) average from the new filled-in matrix, then obtain the row centered matrix M'. SVD factorizes M' into three matrices: $M'_{m\times n} = T_{m\times r} S_{r\times r} V_{r\times n}$, where r is the rank of M', and S is a diagonal matrix with all singular values of M', and T and V are both orthogonal matrices. To reduce the dimensionality, only $t(t < r)$ largest singular values are remained and the others are omitted. Then T, S, V are accordingly reduced to $T^*_{m\times t}$, $S^*_{t\times t}$, $V^*_{t\times n}$. Let $M^* = T^* S^* V^* = [v^*_1, \cdots, v^*_n]$. Then each column vector v^*_i in M^* is the feature vector of item i in the latent semantic space, and the similarity between two items could be measured by the normalized dot product of their corresponding feature vectors.

$$s(i,j) = \frac{v^*_i \cdot v^*_j}{\|v^*_i\| \times \|v^*_j\|} \tag{2}$$

Additionally, it is known that the data density of Party B impacts on the performance of its item similarity analysis [13]. In order to make Party A more flexible to leverage the shared data from Party B, we let Party B also publish its data density ρ_B, which is the faction of existing rating scores in M_B:

$$\rho_B = \frac{\left|\left\{(u,i)|u \in U_B, i \in I, r^B_{ui} \neq \Phi\right\}\right|}{m_b \cdot n} \tag{3}$$

3.2 Step 2: Item Neighborhood Boosting

After acquiring the shared data set from Party B - the item neighborhood data set B and data density ρ_B, Party A leverages it to improve the CF performance.

Without shared data from Party B, for Party A, a typical single-party memory-based CF algorithm is: if k-nearest neighborhoods of an item i are acquired as A_i, the predicted rating of user $u \in U_A$ on item i is calculated as:

$$r^{A\,\prime}_{ui} = \overline{r^A_i} + \frac{\sum_{j\in A_i} c_A(i,j) \cdot (r^A_{uj} - \overline{r^A_j})}{\sum_{j\in A_i} c_A(i,j)}, \tag{4}$$

where $c_A(i,j)$ is the Pearson correlation of item pair (i,j) in the context of Party A.

In order to leverage the shared data from Party B, we propose to fuse the derived neighborhood information from the two parties. An algorithm, named as neighborhood boosting, is designed to combine A_i and B_i for every item i, and get a boosted neighborhood set, notated by H_i, to be used in equation (4) instead of A_i.

The user set in Party A and B is not the same, thus the two parties may have different user patterns. B_i reflects Party B's user pattern only. Arbitrary usage of B_i in Party A's CF might incur huge noise. On the other hand, item similarity analysis, either in Party A or in Party B, might be biased due to insufficient data, noise, or the inherent limitation of similarity measures, etc., in the context of its own party. But if A_i and B_i has an intersection $N_i = A_i \cap B_i$, the neighborhood information in N_i should be more convincing than others, in the context of both parties. Thus, the neighborhoods in N_i should be got higher priority for utilization. It is assumed that the amount of the neighborhoods in resultant boosted neighborhood H_i is to be kept as k, the same as A_i and B_i. Besides N_i, another $(k - |N_i|)$ neighborhoods are selected from $(A_i - N_i) \cup (B_i - N_i)$ leveraging the data density information of each party. If Party A's data is denser than Party B's, it shows that CF in Party A should rely more on its own data, i.e., choose more information from $(A_i - N_i)$ than from $(B_i - N_i)$. Contrarily, if Party A's data is sparser than Party B's, $(A_i - N_i)$ might be not sufficiently valuable due to the data sparseness. Then, although $(B_i - N_i)$ reflects only Party B's user pattern, we choose more information from $(B_i - N_i)$ than from $(A_i - N_i)$. If Party A's data density is ρ_A, we have:

$$k^A_i = \left[(k - |N_i|)\frac{\rho_A}{\rho_A + \rho_B}\right], k^B_i = k - |N_i| - k^A_i \tag{5}$$

where $[\cdot]$ is integral operator to get the integral part of its parameter. If $(A_i - N_i)$ and $(B_i - N_i)$ are both descending sorted by similarity score, let A^*_i is set of the first k^A_i neighborhood items in $(A_i - N_i)$, and B^*_i is set of the first $k^B)_i$ neighborhood items in $(B_i - N_i)$. Finally, the boosted neighborhood set H_i is gotten as $H_i = N_i \cup A^*_i \cup B^*_i$. Since this algorithm is to combine two relatively weak neighborhood set A_i and B_i into a stronger set H_i for Party A, we name it neighborhood boosting.

After the boosted neighborhood set is derived, the predicted rating of user $u \in U_A$ for item i is calculated as:

$$r^{A\,\prime}_{ui} = \overline{r^A_i} + \frac{\sum_{j\in H_i} c_A(i,j) \cdot (r^A_{uj} - \overline{r^A_j})}{\sum_{j\in H_i} c_A(i,j)} \tag{6}$$

4. EXPERIMENTS AND EVALUATION

4.1 Data Set Preparation

Two popular movie recommendation data sets, MovieLens [1] and Netflix [2], are used to simulate a shared CF process. The MovieLens data set version with 10 million ratings for 10,681 movies by 71,567 users is picked. Netflix data set has 100,480,507 ratings for 17,770 movies by 480,189 users. The rating scores in the two data sets are both graded from 1 to 5. We let MovieLens act as Party A (beneficiary party), and Netflix as Party B (contributor Party).

To let the two parties have the same items, the movie items of these two data sets are aligned by comparing the movie title plus publication year, and 4,064 identical items are found. Meanwhile, to limit the data size for speeding up the experiments, the data sets are cut down: for movie items, 1,000 items are randomly sampled from the 4,064 identical movie items; for users, within each party, 10,000 users are randomly sampled; correspondingly, only the rating scores related to the sampled items and users are picked in both parties. Finally, the sampled data set of MovieLens, denoted by X, includes 183,934 ratings for 1,000 movies by 10,000 users, and the sampled data set of Netflix, denoted by Y, includes 305,401 ratings for the 1,000 movies by 10,000 users. Within these two sampled sets, their movie sets are identical, while the user sets are different.

Since the rating prediction accuracy for Party A is to be evaluated, a set of test data needs to be picked from MovieLens data set. We randomly divide the rating scores of MovieLens data set X into 5 subsets with equal size. Note that these subsets of ratings are still in the matrix of the

[1] http://www.grouplens.org/node/73
[2] http://www.netflixprize.com

1,000 movies and 10,000 users. 4 subsets are picked as training data (i.e., existing ratings), and 1 another subset is for test. With permutation, totally 5 combinations of training data (denoted by $X^1_{\#1} - X^5_{\#1}$) and test data are generated.

Furthermore, in order to investigate the performance variation with the data density, we also generate several groups of data sets with different density from MovieLens data set. We randomly divide $X^1_{\#1}$ into 5 subsets with equal size. Similarly, 4 subsets are picked as training data, and 1 another subset is for test. By permutation, totally 5 combinations of training data (denoted by $X^1_{\#2} - X^5_{\#2}$) and test data are generated for cross-validation. Apparently, $X^1_{\#2}$ is sparser than $X^1_{\#1}$. Repeating such dividing process, we totally generated 15 groups of data sets, denoted by Group #1-#15. Within each group $\#i$ (i=1-15), there are 5 combinations of training data (denoted by $X^1_{\#i} - X^5_{\#i}$) and test data for cross-validation. Different group has different data density.

We also generate 5 different Party B data sets from Netflix data set for multiple tests. We randomly divide the ratings of Netflix Data set Y into 5 subsets with equal size. Each two subsets are united to a data set. Then 5 data sets are constructed, denoted by $Y^1 - Y^5$, each of which includes 122,160 ratings, and the data density is 1.22%.

Finally, there are 15 groups $\times$5 sets of Party A data and 5 sets of Party B data. Table 1 shows the information of these generated data sets.

4.2 Methods and Metrics

Our approach is derived from memory based (item neighborhood based) CF approach. So, we select single-party item neighborhood based CF, i.e., equation (4), as a baseline to validate the performance improvement brought by shared CF. Moreover, if without privacy concern, Party B can share M_B directly. Then in order to investigate the influence by considering the privacy in shared CF, we set up another baseline: assuming an imaginary site C includes a user set $U_A \cup U_B$, and the rating matrix is the concatenation of M_A and M_B, C makes single-party CF, while the evaluation is still based on the test data of Party A.

Since we have two kinds of item similarity measures, finally we have 8 different methods for experiments. Among them, four methods are for shared CF:

- **P(A)+P(B)**: for both Party A and B, the item similarity is calculated with Pearson correlation;
- **P(A)+L(B)**: for Party A, the item similarity is calculated with Pearson correlation based similarity analysis; while for Party B, the item similarity is calculated with latent semantic model;
- **L(A)+P(B)**: for Party A, the item similarity is calculated with latent semantic model based similarity analysis; while for Party B, the item similarity is calculated with Pearson correlation;
- **L(A)+L(B)**: for both Party A and B, the item similarity is calculated with latent semantic model.

And the other four methods are for baseline:

- **P(A)**: single-party CF in Party A, the item similarity is calculated with Pearson correlation;
- **L(A)**: single-party CF in Party A, the item similarity is calculated with latent semantic model;
- **P(A+B)**: single-party CF in the imaginary site C, the item similarity is calculated with Pearson correlation;
- **L(A+B)**: single-party CF in the imaginary site C, the item similarity is calculated with latent semantic model.

For all latent semantic model based similarity analysis, the dimension reduction parameter t is set as 2.

We adopt Mean Absolute Error (MAE) for evaluating the accuracy of rating predictions. For each group $\#i$ (i=1-15) of Party A's data sets: (1) For method P(A), L(A), there are totally 5 experiments, each of which is done with one data set of group $\#i$. The average MAE value of these 5 experiments is acquired. (2) For other 6 methods which incorporate Party B's data, experiments are done with different Party B's data set ($Y^1 - Y^5$) one by one. Firstly, with each $Y^j (j = 1 - 5)$, 5 experiments are done using group $\#i$'s 5 sets, and a first-step average MAE value is acquired from such 5 experiments for the specific Y^j. Secondly, the final average MAE value is calculated from the 5 first-step average MAE values.

4.3 Selection of k

All the methods for experiments are based on item neighborhood. Thus the parameter k for k-nearest neighborhood needs to be decided. We adopted 5 different k values (20, 40, 60, 80, and 100) for experiments one by one. For each method and each data set, we record which k value can bring the best performance (the smallest average MAE value). And then for each method and each k value, we count how much percentage this k value can bring the best performance in all the data sets. The results (Table 2) show that, for most methods (6 of 8), the best performance can be reached when $k = 20$. The results also show that for each method, the variance of the average MAE vales with different k values is all quite small ($\leq 1.7 \times 10^{-7}$), which indicates that, for all the methods, the performance is stable with different k values. Thus, for final evaluation, we let $k = 20$.

4.4 Experiment Results

The average MAE results of each method with each Party A's data set group are shown in Table 3 and Figure 1, where the horizontal axis stands for the density ratio of Party A's training data to Party B's data, and the vertical axis stands for the resultant average MAE value. The results show that, in the whole, all shared CF methods outperform the single-party CF with Party A's own data, and even outperforms P(A+B) and L(A+B), which incorporate Party B's original rating data without privacy preservation. And among the four shared CF methods, P(A)+L(B) has the best performance. Such comparison demonstrates that our proposed shared CF methods can obtain win-win of both Party A's recommendation accuracy and Party B's data privacy.

In details, the chart shows that more sparse is Party A's data, more performance increase can be obtained by adopting shared CF methods, comparing to the baseline methods. When the density ratio is below 10%, with P(A)+L(B), the performance could be increased by around 10% comparing to L(A) and around 6% comparing to P(A). And other shared CF methods could also get around 6% performance increase to L(A). It demonstrates that when Party A's own data is too sparse to give sufficient information for CF, Party B's data could help a lot. But the interesting phenomenon

Table 1: Data sets generated from MovieLens (Party A) and Netflix (Party B)

MovieLens (Party A) (Each group has 5 sets) 1,000 items × 10,000 users				Netflix (Party B) (5 sets) 1,000 items × 10,000 users		Density ratio of Party A's training data to Party B's data
Data Set Group	Training Data		Test Data	Ratings	Density	
	Ratings	Density	Ratings			
#1	147,144	1.47%	36,786	122,160	1.22%	120.45%
#2	117,712	1.18%	29,428			96.36%
#3	941,68	0.94%	23,542			77.09%
#4	75,332	0.75%	18,833			61.67%
#5	60,264	0.60%	15,066			49.33%
#6	48,208	0.48%	12,052			39.46%
#7	38,564	0.39%	9,641			31.57%
#8	30,848	0.31%	7,712			25.25%
#9	24,676	0.25%	6,169			20.20%
#10	19,740	0.20%	4,935			16.16%
#11	15,792	0.16%	3,948			12.93%
#12	12,632	0.13%	3,158			10.34%
#13	10,104	0.10%	2,526			8.27%
#14	8,080	0.08%	2,020			6.61%
#15	6,464	0.06%	1,616			5.29%

Table 2: Experiment results for selection of k

	k=20	k=40	k=60	k=80	k=100	Variance
P(A)	16.54%	18.05%	19.92%	22.56%	**22.93%**	1.6×10^{-8}
L(A)	**56.10%**	15.85%	8.54%	9.76%	9.76%	8.9×10^{-9}
P(A+B)	**32.29%**	22.92%	12.50%	15.63%	16.67%	8.3×10^{-9}
L(A+B)	**30.32%**	22.87%	15.96%	18.62%	12.23%	1.6×10^{-8}
P(A)+P(B)	24.52%	14.42%	**30.77%**	14.90%	15.38%	1.7×10^{-7}
L(A)+P(B)	**38.83%**	15.96%	5.32%	23.94%	15.96%	6.8×10^{-9}
P(A)+L(B)	**44.57%**	27.43%	12.00%	8.57%	7.43%	1.5×10^{-8}
L(A)+L(B)	**28.13%**	21.88%	13.54%	17.71%	18.75%	1.3×10^{-8}

is that although P(A+B) and L(A+B) also leverage Party B's data, even with detailed original rating data of Party B, their performance is contrarily worse than the methods with only Party A's own data. That is to say, if leveraging Party B's data with simply concatenating M_A and M_B, the performance of CF for Party A would not get better, but contrarily become even worse. The reason is analyzed as below. Since the two parties have different user groups, the community feature should differ from each other. Then if the rating data is mixed up for item similarity analysis, each party's data might act as noise for the other, so that the performance decreases. However, our proposed shared CF methods do not directly incorporate the detailed rating data of Party B, but do leverage the item neighborhood information which is generated by Party B itself. This finding is similar to the basic idea of [1], where the author argues that the user-to-user similarity computation is not reliable when various domains of items are mixed together. It is the same for item similarity analysis with diverse communities of users.

Our proposed shared CF approach is based on so-called item neighborhood boosting, i.e. combining the resultant item neighborhood information from both parties. We argue that even when Party A's data is much sparser than Party B, Party A can still generate more or less item neighborhood information by using its own data that is unique and not existing in Party B's output. So, although the contribution ratio of Party A's neighborhood information is small when

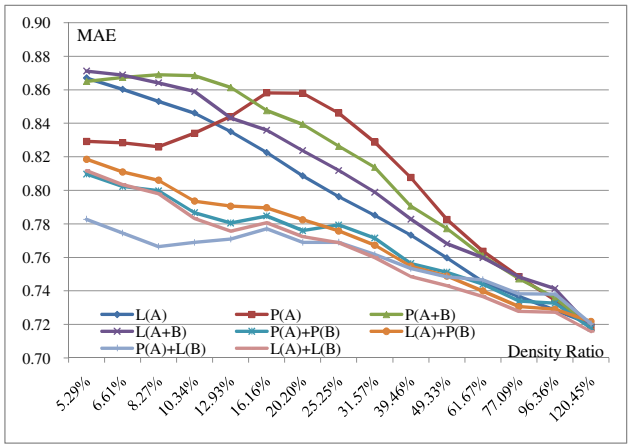

Figure 1: Average MAE of each method and data set group

Party A is much sparser than Party B (according to equation (5)), it could be still effective. In order to prove this, we complement another experiment where the final neighborhood information for prediction is only from the Party B's output. Since P(A)+L(B) has the best performance in the above experiments, in this complementing experiment, the Party B's item similarity analysis is based on latent se-

Table 3: Experiment results table - average MAE for each method and data set group

Party A's Data Set Group		Baseline Methods				Shared CF			
No.	Density ratio to Party B's data	P(A)	L(A)	P(A+B)	L(A+B)	P(A) +P(B)	L(A) +P(B)	P(A) +L(B)	L(A) +L(B)
#1	120.45%	0.7184	0.7203	0.7183	0.7182	0.7177	0.7217	0.7204	0.7157
#2	96.36%	0.7344	0.7286	0.7360	0.7414	0.7326	0.7290	0.7380	0.7272
#3	77.09%	0.7485	0.7366	0.7472	0.7484	0.7339	0.7306	0.7383	0.7278
#4	61.67%	0.7637	0.7454	0.7609	0.7599	0.7440	0.7399	0.7466	0.7366
#5	49.33%	0.7826	0.7597	0.7772	0.7682	0.7511	0.7489	0.7484	0.7432
#6	39.46%	0.8076	0.7733	0.7906	0.7828	0.7564	0.7547	0.7534	0.7485
#7	31.57%	0.8287	0.7851	0.8138	0.7989	0.7715	0.7673	0.7617	0.7599
#8	25.25%	0.8461	0.7962	0.8263	0.8119	0.7794	0.7758	0.7690	0.7687
#9	20.20%	0.8579	0.8087	0.8394	0.8237	0.7759	0.7824	0.7689	0.7724
#10	16.16%	0.8582	0.8225	0.8476	0.8359	0.7847	0.7896	0.7771	0.7806
#11	12.93%	0.8440	0.8350	0.8408	0.8314	0.7805	0.7906	0.7709	0.7756
#12	10.34%	0.8341	0.8461	0.8684	0.8642	0.8104	0.8212	0.8037	0.8116
#13	8.27%	0.8259	0.8530	0.8690	0.8641	0.7998	0.8060	0.7665	0.7980
#14	6.61%	0.8284	0.8602	0.8898	0.8954	0.8246	0.8321	0.7946	0.8238
#15	5.29%	0.8292	0.8670	0.8650	0.8712	0.8097	0.8185	0.7827	0.8118

mantic model. And this experiment is denoted by L(B). Figure 2 shows the results. It illustrates that when Party A's data is much sparser than Party B (the density ratio is below 15%), the performance of shared CF with neighborhood boosting is better than with either Party A's or Party B's neighborhood information solely.

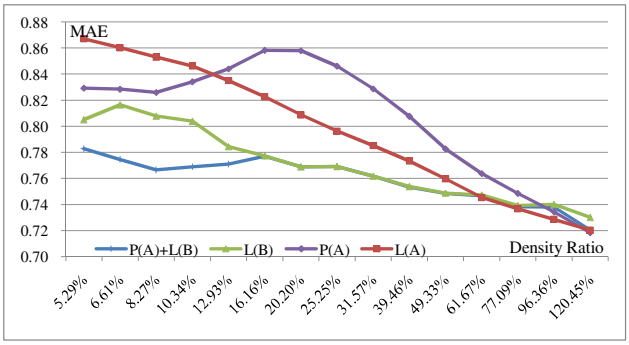

Figure 2: The experiments result showing the effects of item neighborhood boosting

However, Figure 2 also shows that when the density ratio is above around 15%, P(A)+L(B) has almost the same performance as L(B), and also close to L(A). It might be caused by the data sets we choose in the experiments. Since MovieLens and Netflix are both movie data sets with general user community, although their different user groups would lead different user interest patterns, such difference might be not big. Thus, when the density of these two data sets is close, the neighborhood information generated by either party is similar, and the effect of neighborhood boosting cannot stand out.

In a word, the experiments show that our proposed item neighborhood based shared CF method is effective to improve Party A's recommendation performance with leveraging Party B's data, while preserving the data privacy of Party B at the same time.

5. PRIVACY DISCUSSION

Besides data density, the contributor Party B only shares an ordered set of the k most similar items for each item, but not the detailed rating scores or even the similarity scores between items. Intuitively, such high-level ordered sets could mask the user privacy of party B. We designed an experiment to validate it, with the idea of differential privacy [2]. Differential privacy measures how probable an adversary can infer from the output of a computation about any record's presence or absence in the input, by investigating the difference of the computation outputs on adjacent inputs which differ on at most one record.

Within the above evaluation experiments, 5 sets from Netflix act as Party B. Each set includes 10,000 users and their 122,160 ratings on 1,000 items. We concern about per user privacy, that is to say, how probable an adversary can infer from the outputted ordered lists and data density about whether a specific user is in the Party B or not. Within a set, we exclude one user with his or her ratings to form an adjacent input data set of the original set, and investigate the difference between the new output and the original one. Then for each set, there are totally 10,000 adjacent data sets by choosing different user for excluding.

First we look at the shared item neighborhood information. For an item i, the original output of the k most similar items is an ordered set B_i, and the output from an adjacent input data set $B^{'}$ is denoted by $B_i^{'}$. Without concern of order, the intersection of the two sets are denoted by $U_i = B_i \cap B_i^{'}$, and the difference of $B_i^{'}$ from B_i is denoted by $D_i = B_i^{'} - B_i$. We define an edit distance measure between the two ordered sets as a 2-dimension vector: $\|B_i^{'} - B_i\|_o = [r, s]^T$, where $r = |D_i|$ represents the number of $B_i^{'}$'s missing items comparing to B_i, and s represents the minimum swap operations are needed for the common elements U_i in $B_i^{'}$ to reach the same order in B_i. For example, if $B_i = \{i_1, i_2, i_3, i_4\}$ and $B_i^{'} = \{i_4, i_3, i_1, i_5\}$, then $\|B_i^{'} - B_i\|_o = [1, 2]^T$.

Within each Party B data set, for each item, we output the most k similar item sets with all 10,000 adjacent data sets

using both Pearson correlation and latent semantic based methods, and calculate the distance from the original result. And k is again set as 20. For Pearson correlation based method, the maximum distance for similarity score (according to equation (1)) is $max\Delta_{c(i,j)} = 0.4103$. And for the edit distance of the resultant ordered k most similar item set, maximum r is 5 and s is always zero, and 97.91% of the results achieve zero distance, i.e., $B_i = B_i^{'}$. For latent semantic based method, the maximum distance for similarity score (according to equation (2)) is $max\Delta_{s(i,j)} = 0.1372$. For the edit distance of the resultant ordered set, maximum r is 3 and s is also always zero, and 99.32% of the results achieve zero distance. The slight distance obviously shows that the item neighborhood information is not sensitive about any user's presence or absence. And comparing to detailed similarity scores, the ordered set of the most k similar items are more suitable for sharing , since it is more hardly influenced by the presence or absence of any one user.

Next, we investigate the other part of shared data: the data density of Party B. According to the definition in equation (3), the theoretical maximum difference of the density of an adjacent input data set $B^{'}$ from the original ρ_B is:

$$max\Delta_{\rho B} = max_{B'} \left|\rho_B - \rho_{B'}\right| = \frac{1-\rho_B}{m_b - 1}. \quad (7)$$

Since most of the data sets used for collaborative filtering are sparse, this theoretical maximum difference is usually quite small. For the data sets we use in the experiment, $\rho_B = 1.22\%$ and $m_b = 10,000$, then $max\Delta_{\rho B} = 9.88 \times 10^{-5}$. And based on the experiments of outputting with all 10,000 adjacent inputs for each Party B data set, the actual maximum difference is only 2.56×10^{-5}. The difference is so tiny that the data density is also not sensitive to whether a specific user exists in Party B or not.

Moreover, we look at the quantitative definition of differential privacy [2]: A computation K satisfies ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one record, and all $S \subseteq Range(K)$,

$$Pr[K(D_1) \in S] \leq e^{\epsilon} \times Pr[K(D_2) \in S]; \quad (8)$$

and a theorem for achieving differential privacy by random noise [2]: for a function $f(X)$ with d-term output, $f(X) + Laplace(0,\sigma)^d$ satisfies ϵ-differential privacy when

$$\sigma \geq max_{D_1,D_2} \left\|f(D_1) - f(D_2)\right\|_1 / \epsilon, \quad (9)$$

where D_1 and D_2 differ on at most one record.

Thus, in order to preserve the privacy more strictly and achieve ϵ-differential privacy, we can add appropriate Laplace random noise to perturb both the similarity scores and data density value. We select $\epsilon = 0.1$, then for each item pair (i,j), the Pearson correlation is calibrated to $\bar{c}(i,j) = c(i,j) + Noise_1$, where $Noise_1$ comes from $Laplace(0,\sigma_1)$ and $\sigma_1 = max\Delta_{c(i,j)}/\epsilon = 4.103$; the latent similarity based similarity score is calibrated to $\bar{s}(i,j) = s(i,j) + Noise_2$, where $Noise_2$ comes from $Laplace(0,\sigma_2)$ and $\sigma_2 = max\Delta_{s(i,j)}/\epsilon = 1.372$. Then we acquire the k most similar items for each item i - $\bar{B}_i$ with $\bar{c}(i,j)$ or $\bar{s}(i,j)$. Similarly, the data density of Party B is also calibrated with noise as $\bar{\rho}_B = \rho_B + Noise_3$, where $Noise_3$ comes from $Laplace(0,\sigma_3)$ and $\sigma_3 = max\Delta_{\rho B}/\epsilon = 2.56 \times 10^{-4}$. With such calibrated shared data, we repeated the evaluation experiments described in section 4. The difference of shared CF results from original results is shown in Table 4. The results shows that the added random noise slightly influence the accuracy of shared CF, and does not change the conclusion that shared CF outperforms than the baseline methods. And since $max\Delta_{s(i,j)} < max\Delta_{c(i,j)}$, we can notice that the noise-incurring difference of using latent semantic based item similarity measure in Party B is smaller than that of using Pearson correlation in Party B.

Table 4: Difference with calibrated shared data

No.	P(A) +P(B)	L(A) +P(B)	P(A) +L(B)	L(A) +L(B)
#1	0.74%	1.20%	-0.19%	0.34%
#2	1.11%	1.81%	0.48%	0.47%
#3	0.76%	0.89%	0.17%	0.28%
#4	1.41%	1.22%	-0.42%	0.31%
#5	1.26%	1.83%	0.39%	0.57%
#6	0.06%	0.49%	0.14%	0.33%
#7	1.84%	1.20%	0.21%	0.57%
#8	-0.54%	0.98%	-0.09%	-0.46%
#9	-0.03%	0.52%	0.47%	0.32%
#10	1.71%	1.47%	0.52%	0.32%
#11	1.52%	-1.92%	0.24%	0.89%
#12	1.59%	0.03%	0.39%	0.48%
#13	-0.30%	1.11%	-0.41%	-0.29%
#14	0.83%	0.45%	-0.31%	0.29%
#15	0.79%	1.22%	0.57%	0.42%

[5] also leverages differential privacy to propose a private recommender system within a single party, and the added random noise is based on the theoretical maximum distance of each computation on any two adjacent inputs. However, we generate noise with the actual maximum distance, for example, for data density of Party B, we use actual maximum distance 2.56×10^{-5} instead of theoretical distance 9.88×10^{-5}. Since the data publishing in shared CF scenario is usually not real-time or even just one-time process, the input data set is already fixed when being preprocessed for sharing. Thus the actual maximum distance can be used for generating random noise for achieving differential privacy in our case. Apparently, the advantage of using actual maximum distance is to decrease the noise and to publish more accurate output while preserving privacy. Moreover, Party B can be flexible to choose which parts of ratings to participate in generating shared data to avoid outliers and control the noise level.

6. RELATED WORK

The similar work to shared CF is distributed or multi-party CF, which handles CF over distributed community data. The motivation for data distribution can be separated into three types: to improve the accuracy of prediction; to alleviate the scalability problem of traditional centralized CF; or to tackle the privacy risk of recommender keeping all the detailed community data. Such three kinds of motivation lead different research focus.

The distributed CF method proposed in [1] belongs to the first type. It argues that user similarity computation is not reliable when the rating data is mixed with diverse domains of items. Then an item-content-dependent partitioning method for rating data is designed, and the rating data is distributed into several specialized repositories. CF is then based on aggregating information coming from these

distributed repositories. The recommendation accuracy can be improved by this method. Such scenario is totally different from ours, but its finding can be used to be an evidence for our result that shared CF with aggregating (boosting) item neighborhood information from multiple parties outperforms the method which concatenates different parties' rating data simply. Item similarity analysis in the latter method is not reliable since the different parties have different user groups.

The second type of distributed CF research is mostly leveraging P2P ideas and technologies to solve the scalability problem of centralized CF. Most work is on the neighbor locating in the distributed environment, such as the routing algorithms proposed by [12]. This scenario is also far from our focus. And since the routing algorithms do not consider the privacy issues, and moreover their goal is to make the performance of distributed CF is comparable to centralized CF, the methods are difficult to be applied for shared CF.

The third type aims at the privacy issues. The task of such type of distributed CF is to make the CF performance as good as or close to the performance when the data is centralized, and to preserve the privacy of each distributed party at the same time. Some work sanitizes or pertubates the original data of each party in a designed fashion and carries out CF based on sanitized data while guaranteeing the decent accuracy, such as [10]. Secure multi-party computing (SMC) based approaches are also popular, such as [6] proposes secure communication protocols among different parties for memory-based CF. Moreover, some other work specifically aims at the scenario that each user keeps its own profile information, and privacy-preserving user similarity analysis methods are proposed, such as [4]. Privacy-preserving distributed or multi-party CF seems to be similar to our work. However, in fact, the scenario and target are different. They consider the data distributed in different parties as a whole. Their goal is to guarantee the CF performance and the data privacy for all parties. While, shared CF considers only the recommendation accuracy of Party A (beneficiary party), and Party B (contributor party) only shares its information. Our goal is to improve only Party A's CF performance, while preserving Party B's privacy. And most methods proposed for privacy-preserving multi-party CF needs mutual communication between different parties. However, in shared CF scenario, Party B just publishes its shared data, and Party A collects it. There is no communication needed between Party A and Party B, which is much more practical.

7. CONCLUSION AND FUTURE WORK

Considering leveraging the data from other parties to improve the collaborative filtering performance, shared collaborative filtering is proposed. Item neighborhood information is proposed to be shared by the contributor party to keep the data privacy of the contributor party. In order to leverage such shared information and at the same time considering the difference between the user interest patterns of these two parties, an algorithm called item neighborhood boosting is designed for fusing the respective item neighborhood information generated by each party. MovieLens and Netflix datasets are used as two data parties for simulating shared CF. The experiment results show that shared CF has positive effects on the prediction accuracy, and our methods even outperforms the methods that incorporate the detailed original rating scores of the contributor party without considering privacy. Thus, we obtain a win-win situation for both performance and privacy.

There are still several open issues worthy of future investigation. Based on the experiment results, shared CF can increase the performance in different degree for different data density. So it might be useful to preprocess, such as partitioning, the data of beneficiary party to maximize the positive effects of shared CF. Additionally, as discussed in experiment section, to seek other data sets which have quite different user interest patterns for experiments would be interesting and meaningful for validating our methods. And in current experiments, only basic-type single-party CF algorithms are used for both baseline and proposed methods, state-of-the-art advanced algorithms will be tried. Moreover, when more than one contributor parties share their information, how to leverage it is also our future work.

8. REFERENCES

[1] S. Berkvosky, T. Kuflik, and F. Ricci. Distributed collaborative filtering with domain specialization. In *RecSys'07 Proceedings*, pages 33–40, 2007.

[2] C. Dwork. Differential privacy: A survey of results. In *TAMC'08 Proceedings*, pages 1–19, 2008.

[3] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[4] N. Lathia, S. Hailes, and L. Capra. Private distributed collaborative filtering using estimated concordance measures. In *RecSys'07 Proceedings*, pages 1–8, 2007.

[5] F. McSherry and I. Mironov. Differentially private recommender systems. In *KDD'09 Proceedings*, pages 627–636, 2009.

[6] H. Polat and W. Du. Privacy-preserving collaborative filtering on vertically partitioned data. In *PKDD'05 Proceedings*, pages 651–658, 2005.

[7] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *WEBKDD'00 Proceedings*, 2000.

[8] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Item-based collaborative filtering recommendation algorithm. In *WWW'01 Proceedings*, pages 285–295, 2001.

[9] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *CHI'95 Proceedings*, pages 210–217, 1995.

[10] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys'09 Proceedings*, pages 157–164, 2009.

[11] M. Vozalis, A. Markos, and K. Margaritis. Evaluation of standard svd-based techniques for collaborative filtering. In *HERCMA'09 Proceedings*, 2009.

[12] B. Xie, P. Han, F. Yang, R.-M. Shen, H.-J. Zeng, and Z. Chen. Dcfla: A distributed collaborative-filtering neighbor-locating algorithm. *Information Science*, 177(6):1349–1363, 2007.

[13] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR'05 Proceedings*, pages 114–121, 2005.