2005

Privacy-preserving top-N recommendation on horizontally partitioned data

Huseyin Polat
Syracuse University, Department of Electrical Engineering and Computer Science, hpolat@ecs.syr.edu

Wenliang Du
Syracuse University, Department of Electrical Engineering and Computer Science, wedu@ecs.syr.edu

Privacy-Preserving Top-*N* Recommendation on Horizontally Partitioned Data

Huseyin Polat and Wenliang Du
Department of Electrical Engineering and Computer Science
Syracuse University, CST 3-114, Syracuse, NY 13244-1240, USA
hpolat,wedu@ecs.syr.edu

Abstract

Collaborative filtering techniques are widely used by many E-commerce sites for recommendation purposes. Such techniques help customers by suggesting products to purchase using other users' preferences. Today's top-N recommendation schemes are based on market basket data, which shows whether a customer bought an item or not. Data collected for recommendation purposes might be split between different parties. To provide better referrals and increase mutual advantages, such parties might want to share data. Due to privacy concerns, however, they do not want to disclose data.

This paper presents a scheme for binary ratings-based top-N recommendation on horizontally partitioned data, in which two parties own disjoint sets of users' ratings for the same items while preserving data owners' privacy. If data owners want to produce referrals using the combined data while preserving their privacy, we propose a scheme to provide accurate top-N recommendations without exposing data owners' privacy. We conducted various experiments to evaluate our scheme and analyzed how different factors affect the performance using the experiment results.

1. Introduction

Collaborative filtering (CF) is a recent technique for prediction and recommendation purposes that helps users cope with information overload using other users' preferences. The concept of CF originated in the early nineties with the Information Tapestry project [3]. CF techniques are widely used in E-commerce, direct recommendations, and search engines to suggest items to users [1, 2].

CF systems work by collecting ratings for items and matching together users sharing the same interest or styles. The goal of CF is to predict how well a user, referred to as the *active user* (a), will like an item that he/she did not buy before based on a community of users' preferences [5]. The key idea is that a will prefer those items that like-minded users prefer, or that dissimilar users do not. Filtering systems provide predictions for single items. They also perform top-*N* recommendation (TN), in which an ordered list of items that will be liked by a is provided.

Today's TN schemes [14, 9, 10] are based on market basket data where users' preferences are represented by 1 if they bought the items, or 0 otherwise. We present a TN scheme on binary ratings where customers rate products they bought as 1 if they liked them, or 0 otherwise. In our scheme, neighbors are selected among similar and dissimilar users because a will prefer those items that like-minded users prefer, or that dissimilar users do not.

To provide referrals, data collected from many users is used. Some online vendors, especially those newly created ones, might have problems with available data and own a limited number of users. It becomes difficult for them to form large enough reliable neighborhoods. Holding a low number of users might cause a cold start problem and restricts the CF systems to provide referrals for only a limited number of items. Recommendations then might be unreliable and sometimes are not computable at all.

Data collected for CF purposes might be horizontally or vertically partitioned between different parties. They hold disjoint sets of users' preferences for the same items in horizontal partition while in vertical partition, they own disjoint sets of items' ratings collected from the same users. Combining horizontally partitioned data (HPD) is helpful for CF systems when they own a low number of users. To provide more accurate recommendations, there should be large enough number of neighbors selected from available users; this might be achieved by integrating HPD.

Users buy products from different online vendors. Some users purchase books from Amazon.com while others buy from Barnes & Noble.com. Amazon.com's and Barnes & Noble.com's databases, which include ratings for the same books, recorded from disjoint sets of users, can be jointly used for better referrals. Joint data is beneficial for them because customers prefer returning to stores with better referrals. Combined data will also benefit customers by making it more likely to receive more accurate and reliable referrals.

Mutual advantages due to collaboration between parties can arise from TN grounded on joint data. Data sharing might occur between online vendors, search engines, or even competing E-commerce companies and allows data owners to provide richer recommendation services. TN qualities might be increased if data owners are able to combine their data. Recommendations computed from the combined data are likely more accurate than the ones calculated from one of the disjoint data sets alone because combined data allows the parties to find more reliable neighborhoods. Therefore, TN on HPD is essential. However, due to privacy, legal, and financial reasons, they do not want to share their data. If privacy measures are provided, they can share data. Providing privacy measures is a key to achieve HPD-based TN. Therefore, we investigate the privacy-preserving TN (PPTN) on HPD problem defined as follows:

To maximize the mutual profits, two online vendors, which own disjoint sets of users' preferences of the same items, want to provide TN to their future customers using the combined data while preserving their privacy. How can they produce recommendations on the integrated data without exposing their privacy?

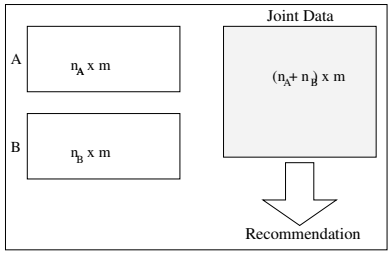

Figure 1. PPTN on HPD

Fig. 1 shows PPTN on HPD. Two vendors, A and B holding n_A and n_B number of users' ratings, respectively of the same m number of items. They perform TN using the joint data, which is an $(n_A + n_B) \times m$ matrix while preserving their privacy. Since privacy and accuracy are conflicting goals, the proposed protocol should achieve a good balance between them. We conducted experiments using two well-known real data sets to show the overall performance of our scheme and how accuracy changes with varying factors.

2. Related work

Canny proposes two schemes for PPCF [1, 2]. In these schemes, users control all of their own private data; a community of users can compute a public "aggregate" of their data, which allows personalized recommendations to be computed without disclosing individual users' data. Polat and Du use randomized perturbation techniques for PPCF [11, 13]. In their scheme, a server collects disguised ratings from users, creates a central database, and starts providing CF services based on the existing database. Although their schemes are based on numerical ratings, provide predictions for single items, and required data is available to the server, we investigate binary ratings-based TN on HPD while preserving data owners' privacy. PPCF on vertically partitioned data (VPD) problem is discussed in [12] while we investigate HPD-based TN with privacy.

Privacy-preserving naïve Bayes classifier for HPD is discussed in [7]. They show that using secure summation and logarithm, they can learn distributed naïve Bayes classifier securely. Privacy-preserving association rules on HPD are discussed in [6]. They address secure mining of association rules over HPD while incorporating cryptographic techniques to minimize the shared data. TN in reduced space is discussed by [14]. Customer preference data is considered as binary by treating each non-zero entry of the user-item matrix as 1. Item-based TN is discussed in [8] where Karypis presents item-based CF algorithms that first determine the similarities between various items and then used them to identify the set of items to be recommended.

3. HPD-based TN with privacy

After data collected for recommendations, a sends his/her known ratings and a query for which items he/she is looking for referrals to a server, which first selects neighbors. Then a frequency count is performed on the items neighbors bought. The item list is sorted and its most frequently purchased N items are returned as referrals. TN algorithms proposed by [14, 9, 10] are based on market basket data. However, purchasing and consuming items do not necessarily mean that consumers liked them. Customers buy products they might like; sometimes, however, they dislike what they bought. Referrals might not be accurate calculated from market basket data. Therefore, we hypothesize that it is likely to provide more accurate recommendations if data showing users' preferences as like or dislike is used.

It is imperative to select those users who have high positive and high negative correlations with a as neighbors because a will prefer those items that like-minded users prefer, or that dissimilar users do not. However, dissimilar users are not considered in TN process in [9, 10]. Accuracy might be increased if we select the best similar and dissimilar users as neighbors.

Similarities between a and other users are computed using different metrics. For market basket data, Tanimoto coefficient [9, 10] is used and can be defined as:

$$w_{au} = \frac{t(y_a \cap y_u)}{t(y_a \cup y_u)} \quad (1)$$

where $t(y)$ represents the number of elements in the basket y. We modify it as follows and use it as a similarity metric:

$$W_{au} = \frac{t(Y_s) - t(Y_d)}{t(Y)} \quad (2)$$

where $t(Y_s)$ and $t(Y_d)$ represent the number of similarly and dissimilarly rated items by users u and a, respectively and $t(Y)$ is the number of commonly rated items by them. For example, if a's ratings vector is $(1, 1, 1, 0, NR, 0, 1)$ and u's ratings vector is $(1, 1, 1, 1, 1, 0, NR)$, then $W_{au} = (4-1)/5 = 0.6$ where NR means not rated. Similarities range from -1 to 1. If $W_{au} > 0$, users a and u are similar; otherwise they are dissimilar. When $W_{au} = 0$, they are not correlated at all. After finding similarities, neighbors are selected using threshold or best-N_n methods to form the neighborhood. In the case of using threshold τ_n, those users whose similarities satisfy the condition $|W_{au}| > \tau_n$ are selected while in best-N_n method, N_n number of best users are selected as neighbors.

Unlike the scheme defined in [14], in our scheme, frequency count is not performed because users rate items as 1 or 0 and the neighbors composed of similar and dissimilar users. We find the number of 1s (l_j) and 0s (d_j) in each item's column after we reverse the ratings of dissimilar users because a will like the items that dissimilar users do not. We then compute $ld_j = l_j - d_j$. If $ld_j > 0$, then the item will be liked by a, otherwise not. After finding all items that will be liked by a, they are sorted according to ld_j values and first N items are returned as top-N recommendation. During TN, some computations can be done off-line while others online. Since online computation cost is critical to the performance, instead of finding referrals for all unrated items, a sends a query stating he/she is looking for recommendations for N_a items where $N < N_a < m - M$ where M is the number rated items by a.

Without privacy as a concern, two companies, A and B can exchange their own data, create a central database, and provide filtering services using the combined data. To get referrals, a sends his/her known ratings and a query to one of the parties, which finds referrals. However, with privacy as a concern, the companies should not be able to learn each other's data. They want to conduct TN using the joint data without disclosing data. Since either party can act as an active user in multiple scenarios to derive information about other party's data, the proposed protocol should be secure against such attacks coming from both parties. They communicate through a during online recommendation computation. The challenge is how they can provide TN services using HPD without exposing their privacy.

3.1. PPTN on HPD

To find neighbors in TN, threshold and best-N_n methods are used. Since different neighbor selection methods follow different steps, we divided our proposed scheme into threshold-based and best-N_n-based schemes and explained them in the followings. Data owners exchange data to find referrals. One party should get all required data for recommendation computations. Either party can act as a server to get required data and find the final referrals. They can switch their roles. We assume that B acts as a server.

3.1.1. Threshold-based PPTN on HPD. In threshold-based TN, users are selected as neighbors based on a a predefined threshold (τ_n) value. Both parties can find similarities between users they hold and a and select neighbors using τ_n. A sends required data to B, which finds recommendations. The details of the scheme are as follows:

Step 1. a sends his/her ratings and a query (for which N_a total number of unrated items he/she is looking for top-N recommendation) to both parties.

Step 2. A computes similarities between users it holds and a and selects neighbors based on τ_n. To prevent B from learning τ_n, A can use a random threshold rather than a fixed one. Therefore, it creates a uniform random number ($r_{A\tau}$) from a range $[-\alpha_A, \alpha_A]$ and adds that number to τ_n, finds $\tau_n + r_{A\tau}$, and uses it as a random threshold. B will not be able to learn the threshold due to the random number.

Step 3. It then finds $ld_{Aj} = l_{Aj} - d_{Aj}$ values for all $j = 1, \ldots, N_a$ from the neighbors' data and sends them to B through a. Since B does not know the threshold value, the neighbors, which neighbors rated which items, and the values of l_{Aj} and d_{Aj}, it will not be able to learn true ratings.

Step 4. B finds similarities for users it holds, selects neighbors based on τ_n, and finds $ld_{Bj} = l_{Bj} - d_{Bj}$ for all $j = 1, \ldots, N_a$. It computes ld_j values and finds referrals as explained before and sends the sorted item list to a.

Both parties can send the ld_{Aj} and ld_{Bj} values for all N_a to a without sending it to each other and a can find recommendations. However, since a gets N_a number of referrals rather than N recommendations, they exchange data and one of them provides recommendations to a.

3.1.2. Best-N_n-based PPTN on HPD. After similarities between all users and a are found, best N_n number of users are selected as neighbors. A finds similarities between those users it holds (u_A) and a and sends $|W_{au_A}|$ values to B, which first finds similarities between users it holds and a and selects best N_n users as neighbors. Since B can act as an active user in multiple scenarios to derive data, the scheme should not allow B to derive data from similarities found by A. The scheme's details are as follows:

Step 1. a sends his/her ratings and a query to A and B.

Step 2. A estimates similarities between users it holds and a using private similarity computation protocol, which is described in the following section to prevent B from deriving data. Then it permutes them using a permutation

function Π_A, which is only known by it, and sends permuted $|W_{au_A}|$ values to B through a. B will not be able to learn true ratings due to Π_A and private similarity computation protocol. It also does not know the types of correlations between users A holds and a.

Step 3. B finds similarities for users it owns and selects best N_n users among all n users as neighbors.

Step 4. It then finds ld_{Bj} values for those N_a items and sends them and the neighbors selected among users A holds to A through a. Since A does not know the neighbors that B selected among users it holds, which neighbors rated which items, and the values of l_{Bj} and d_{Bj}, it will not be able to learn true ratings.

Step 5. A finds ld_{Aj} and computes ld_j values for all N_a items. It then finds top-N recommendation and sends to a.

3.2. Private similarity computation

We propose to use private similarity computation protocol to find the similarities without exposing privacy. Since customers only buy and rate a few, active users' ratings vectors are usually sparse. However, since either party can act as an active user, they might use dense ratings vectors. We only explain the protocol for A because B also follows the same steps to find similarities for users it holds.

After A gets a's data, it finds M. If M is less than $\lfloor m/2 \rfloor$, then A finds the items that a did not rate. A then creates a uniform random integer R_{Aa} from the range $(1, m - M)$ and randomly selects R_{Aa} number of unrated items. It then fills those randomly selected R_{Aa} number of unrated items' cells in a's ratings vector with the corresponding default votes (v_ds) calculated using private default votes computation protocol, which is explained in the following section. If M is bigger than $\lfloor m/2 \rfloor$, A finds the items that a rated and creates a uniform random integer R_{Ar} from the range $(1, M)$. It then randomly selects R_{Ar} number of rated items and removes their ratings from a's ratings vector. A then forms a's new ratings vector and can estimate similarities using it. Since B does not know R_{Aa}, R_{Ar}, and randomly selected rated and unrated items, it will not be able to figure out W_{au_A} values from W'_{au_A} values calculated using new ratings vector even if it acts as an active user in multiple scenarios where u_A represents users that A holds. For each user held by A, A independently creates R_{Aa} or R_{Ar}, finds new ratings vectors, and estimates similarities based on them. The ranges for R_{Aa} and R_{Ar} can be adjusted based on how much privacy and accuracy wanted. Removing some of the ratings and adding v_ds might make accuracy worse because number of available ratings decreases and non-personalized ratings might not represent a's true preferences. However, when there are enough ratings, we can still estimate reliable similarities after removing some of them. Since v_ds are non-personalized ratings for a, it is likely to estimate similarities with decent accuracy using private similarity computation protocol after inserting v_ds.

3.3. Private default votes computation

Our scheme follows online and off-line computation components: Finding v_ds is done off-line while other computations are conducted online. Since default votes (v_ds) are used for finding referrals, before providing predictions to their new customers, data owners find v_ds off-line using private default votes computation protocol as follows:

Step 1. Each party finds l_j and d_j values for all $j = 1, \ldots, m$.

Step 2. A randomly selects $m_A = \lfloor m/2 \rfloor$ number of items. It creates large enough random values r_{Aj} for $j = 1, \ldots, m_A$, adds them to l_{Aj} and d_{Aj} values, and finds $l'_{Aj} = l_{Aj} + r_{Aj}$ and $d'_{Aj} = d_{Aj} + r_{Aj}$. Since $ld_{Aj} = l_{Aj} - d_{Aj} = l'_{Aj} - d'_{Aj}$, A finds ld_{Aj} values without using random numbers. Since B does not know how many users owned by A rated item j and how many of them rated as 1 or 0, it will not be able to learn true ratings for item j. Even if it learns ratings for m_A items, it does not know ratings for remaining $k = m - m_A$ items.

Step 3. A then sends ld_{Aj} values to B, which calculates $ld_j = ld_{Aj} + ld_{Bj}$ values, compares them with 0, and finds v_ds for those m_A items. If $ld_j > 0$, v_d is 1, or 0 otherwise.

Step 4. B finds $ld_{Bj} = l_{Bj} - d_{Bj}$ values for k items where $j = 1, \ldots, k$ and sends them and v_ds for m_A items to A that will not be able to learn ld_{Bj} values for m_A items.

Step 5. A finds v_d values for k items and tells B. They then store them into $m \times 1$ matrices.

4. Analysis

We analyzed our scheme in terms of online overhead costs because off-line costs are not critical to the performance. We show how much additional costs are introduced due to privacy. The number of communications is 2 without privacy as a concern. The overhead communication costs due to privacy are only 3 and 5 for threshold- and best-N_n-based schemes, respectively. The storage overhead due to privacy is relatively small $\big(O(m)\big)$ because A and B store default votes in two $m \times 1$ matrices.

The overhead computation cost is negligible in threshold-based scheme because one party creates a random number r_τ for random threshold and conducts one more addition. In best-N_n-based scheme, one party uses private similarity computation protocol, which increases or decreases the number of comparisons by $R_a n_j$ or $R_r n_j$, on average, respectively depending on M where j is A or B. The same party also uses a permutation function to permute similarities and creates n_j random integers.

We claim that our threshold-based scheme is secure. Since $r_{A\tau}$ is only known by A, B does not know random threshold. It does not know how many and which users were selected as neighbors because A only sends ld_{Aj} values after it selected neighbors using random threshold. B also does not know the types of correlations between users held by A and a. Even if B finds out neighbors and the types of correlations, it will not be able to derive true ratings for N_a items because after it gets ld_{Aj} values for N_a items for $j = 1, \ldots, N_a$, the probability of guessing the correct l_{Aj} and d_{Aj} values for it is 1 out of $(n_A - ld_{Aj} + 1)$. The probabilities of guessing the like and dislike ratings of item j are 1 out of $\mathrm{C}_{l_{Aj}}^{n_A}$ and $\mathrm{C}_{d_{Aj}}^{n_A - l_{Aj}}$, respectively. Therefore, the probability of guessing A's data for N_a number of items for B is 1 out of $\left((n_A - ld_{Aj} + 1)(\mathrm{C}_{l_{Aj}}^{n_A})(\mathrm{C}_{d_{Aj}}^{n_A - l_{Aj}})\right)^{N_a}$ where C_o^p is the number of ways of picking o unordered outcomes from p possibilities. The probability for A can be found similarly when it acts as a server.

Our best-N_n-based scheme is secure due to permutation and private similarity computation protocol. For one user, the probability of guessing R_a is 1 out of $(m - M)$ and the probability of guessing the correct R_a number of items is 1 out of $\mathrm{C}_{R_a}^{m-M}$. The probability of guessing the correct type of correlation is 1 out of 2 and the probability of guessing the correct $t(Y)$ value is 1 out of $(M + R_a)$. For B, the probability of guessing the correct $t(Y_s)$ or $t(Y_d)$ value is 1 out of $\left((M + R_a)(1 - |W_{au_A}|)\right)$. The probabilities of guessing similarly or dissimilarly rated items are 1 out of $\mathrm{C}_{t(Y_s)}^{t(Y)}$ and $\mathrm{C}_{t(Y_d)}^{t(Y)-t(Y_s)}$, respectively. Finally, the probability of guessing the correct users for B is 1 out of $n_A!$. Therefore, the probability of guessing the A's data for B is 1 out of $\Big((n_A!)\big[2(m - M)(\mathrm{C}_{R_a}^{m-M})(M + R_a)(1 - |W_{au_A}|)(\mathrm{C}_{t(Y_s)}^{t(Y)})(\mathrm{C}_{t(Y_d)}^{t(Y)-t(Y_s)})\big]^{n_A}\Big)$ when default votes are appended. The probability can be found similarly when ratings are removed.

We claim that our proposed protocol for finding default votes is secure due to the following reasons. Each party sends ld_j values for corresponding items to each other. Since they do not know how many users held by each other rated item j and how many of them rated as 1 or 0, they will not learn true ratings. Since they exchange data for half of the items, even if a party derives data about them, it will not be able to learn data about others. The probability for B to guess A's data can be found similarly as explained above for threshold-based scheme.

5. Experimental results

5.1. Data sets and evaluation criteria

We used two well-known real data sets in our experiments. Jester has 100 jokes and records of 17,988 users where the ratings range from -10 to +10 and they are continuous [4]. MovieLens (ML) consists of ratings made on a 5-star scale for 3,591 movies made by 7,463 users. It was collected by the GroupLens Research Project (www.cs.umn.edu/research/Grouplens).

We measured the accuracy of our scheme using classification accuracy (CA), coverage, and F-Measure (FM). CA is the ratio of number of correct classifications to number of classifications. Coverage is the percentage of items for which a CF algorithm can provide referrals. FM [14] is a weighted combination of precision and recall, which are used for information retrieval tasks where:

$$FM = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

5.2. Methodology

We first transformed numerical ratings into binary ratings. We labelled items as 1 if the numerical rating for the item was bigger than 3, or 0 otherwise in ML. We labelled them as 1 if the numerical rating for the item was above 2.0, or 0 otherwise in Jester. We randomly selected 9,000 and 6,000 users from Jester for training and testing sets, respectively. ML was randomly divided into training and testing sets with 4,000 and 3,000 users, respectively. We then randomly selected 2,000 users for training among those 9,000 and 4,000 users. 500 users were randomly selected among those 6,000 and 3,000 users as test users.

5.3. Experimental results

To evaluate the overall performance of our scheme, we conducted several experiments. First, we ran experiments to find the optimum τ_n value for neighbor selection. We used 2,000 and 500 users for training and testing, respectively. We held 5 rated items from each test user's data and tried to find predictions for them using our scheme while varying τ_n. We then compared predictions with the true ratings. We only showed CAs in Fig. 2 for both data sets. As seen from the figure, the results are best when τ_n is 0.1 and 0.2 for Jester and ML, respectively. Therefore, we selected them as optimum τ_n values. The results are slightly becoming worse when τ_n is away from its optimum value.

To show how accuracy changes with different numbers of best neighbors (N_n), we conducted experiments using the same 2,000 and 500 users for training and testing, respectively. Since results for both data sets are similar, we only showed FM values for Jester in Fig. 3. We again held 5 rated items' ratings, tried to find recommendations for them, and compared them with true ratings. As seen from Fig. 3, the results are becoming better with increasing N_n up to 1,000 best neighbors and they become steady after that.

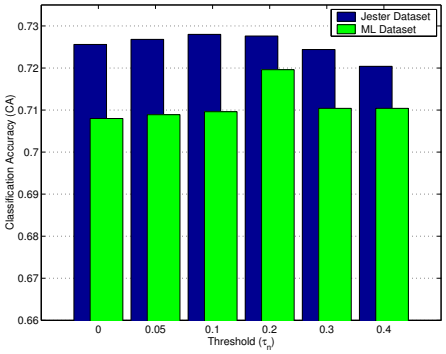

Figure 2. CA vs. threshold (τ_n)

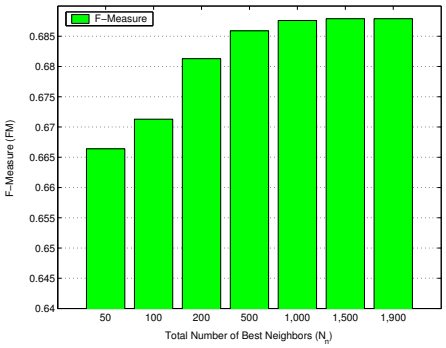

Figure 3. FM vs. N_n

We then ran experiments to show how different numbers of users (n) affect the results. We hypothesize that with increasing n, it is more likely to find large enough neighborhoods for accurate referrals. Since HPD-based recommendation scheme combines two disjoint sets, it is likely to increase accuracy. We used threshold selection scheme for neighbor selection using optimum τ_n values while varying n from 100 to 2,000. We randomly selected training users from training sets where we used the same 500 test users. Using our scheme, top-10 recommendations were found for randomly selected rated items from each test user's ratings vector. We then compared predictions with true ratings, calculated CA and FM values for both data sets, and showed results in Table 1. With increasing n, the results become better. Therefore, combining HPD helps CF systems to provide more accurate referrals.

Table 1. Accuracy vs. n

	n	100	200	500	1,000	2,000
Jester	CA	0.7010	0.7048	0.7078	0.7102	0.7116
	FM	0.6586	0.6619	0.6642	0.6662	0.6703
ML	CA	0.6530	0.6726	0.6874	0.6970	0.7062
	FM	0.7269	0.7480	0.7683	0.7769	0.7863

CF algorithms are not able to provide referrals for all items due to low available data. When there is a low number of users, it becomes difficult to find predictions for some items. Since HPD-based scheme integrates split data, it is more likely to find recommendations for more items. To show how different n values affect coverage, we conducted experiments using ML. We found coverage values while varying n from 50 to 2,000 and showed results for different τ_n values in Fig. 4. As seen from the figure, coverage increases with increasing n because it becomes more likely to have items rated by more users.

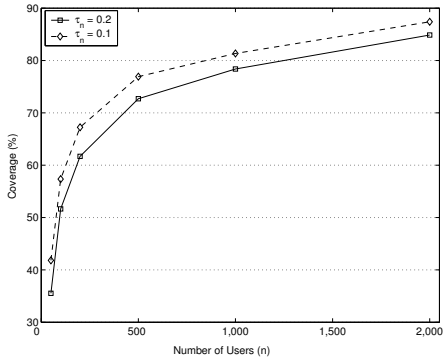

Figure 4. Coverage vs. n

In threshold-based scheme, we propose to use a random threshold. A selects τ_{Ar} values based on α_A. As seen from Fig. 2 where we showed CAs with varying τ_n values, if τ_n is changed from 0.2 to 0.1 or 0.3 for ML, 1% accuracy is lost. Since τ_{Ar} values are uniformly created, when α_A is 0.01, on average, we lost 0.5 % accuracy.

Finally, we ran experiments to show how different R_a and R_r values affect the overall performance. In private similarity computation protocol, we either remove ratings or add default votes for randomly selected items based on M. We hypothesize that inserting default votes might increase accuracy because a's available ratings increases and makes it possible to find more reliable matchings and accurate referrals. However, since default votes might not match a's true preferences for those items, inserting them might make accuracy worse. We conducted experiments while varying R_a values using both data sets and showed FMs for only ML in Fig. 5. We used the same 2,000 training users while 500 users who rated less than 60 items were randomly selected for testing. We then found top-10 recommendations for randomly selected 10 rated items from each test user's data. Predictions for those items were compared with true ratings. As seen from the figure, when R_a is 10, accuracy improves while it becomes worse when it is 20 or more. However, when R_a is 100, accuracy loss is only 1%.

To show how accuracy changes with R_r, we conducted experiments using both data sets and showed results for

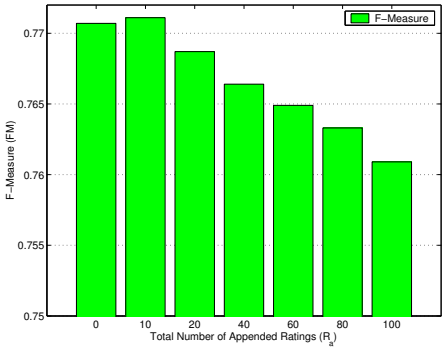

Figure 5. Accuracy vs. R_a

only Jester in Fig. 6. We used the same 2,000 training users while 500 users who rated more than 80 items were randomly selected for testing. We then found top-10 recommendations for randomly selected 10 rated items from each test user's data while varying R_r from 0 to 60. Predictions for those items were compared with true ratings. As seen from the figure, when half of the ratings are removed, accuracy loss is only 1% while it is 2% when R_r is 60. With increasing R_r, results are becoming worse as we expected.

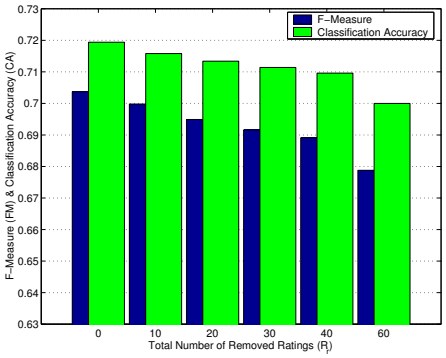

Figure 6. Accuracy vs. R_r

6. Conclusions and future work

We have presented a solution to PPTN on HPD. Our solution makes it possible for two parties to conduct TN using joint data with privacy. Our experiments have shown that our solution can achieve accurate referrals. Our proposed private similarity computation protocol can achieve a good balance between accuracy and privacy by adjusting parameters. Predictions for single items can be computed with privacy using our scheme. We will study how aggregate data disclosure affects the accuracy and the privacy of our scheme. We will also study VPD-based TN with privacy. We will investigate how our scheme works when there is an overlap between users held by data owners.

References

[1] J. Canny. Collaborative filtering with privacy. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 45–57, Oakland, CA, USA, May 2002.

[2] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245, Tampere, Finland, 2002.

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[4] D. Gupta, M. Digiovanni, H. Narita, and K. Goldberg. Jester 2.0: A new linear-time collaborative filtering algorithm applied to jokes. In *Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, Berkeley, CA, USA, August 1999.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. T. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999.

[6] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. *Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

[7] M. Kantarcioglu and J. Vaidya. Privacy preserving naive bayes classifier for horizontally partitioned data. In *Proceedings of the IEEE ICDM Workshop on Privacy Preserving Data Mining*, pages 3–9, Melbourne, FL, USA, 2003.

[8] G. Karypis. Evaluation of item-based top-N recommendation algorithms. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 247–254, Atlanta, GA, USA, 2001.

[9] A. Mild and T. Reutterer. Collaborative filtering methods for binary market basket data analysis. *Lecture Notes in Computer Science*, 2252:302–313, 2001.

[10] A. Mild and T. Reutterer. An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data. *Journal of Retailing and Consumer Services*, 10(3):123–133, 2003.

[11] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, FL, USA, November 2003.

[12] H. Polat and W. Du. Privacy-preserving collaborative filtering on vertically partitioned data. Submitted to the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), October 3–7 2005.

[13] H. Polat and W. Du. SVD-based collaborative filtering with privacy. In *Proceedings of the 20th ACM Symposium on Applied Computing Special Track on E-commerce Technologies*, Santa Fe, NM, USA, March 13–17 2005.

[14] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system-A case study. In *Proceedings of the ACM WebKDD 2000 Web Mining for E-commerce Workshop*, 2000.