Achieving Private Recommendations Using Randomized Response Techniques [⋆]

Huseyin Polat and Wenliang Du

Department of Electrical Engineering and Computer Science
Syracuse University, CST 3-114, Syracuse, NY 13244-1240, USA
Email: {hpolat,wedu}@ecs.syr.edu

Abstract. Collaborative filtering (CF) systems are receiving increasing attention. Data collected from users is needed for CF; however, many users do not feel comfortable to disclose data due to privacy risks. They sometimes refuse to provide information or might decide to give false data. By introducing privacy measures, it is more likely to increase users' confidence to contribute their data and to provide more truthful data. In this paper, we investigate achieving referrals using item-based algorithms on binary ratings without greatly exposing users' privacy. We propose to use randomized response techniques (RRT) to perturb users' data. We conduct experiments to evaluate the accuracy of our scheme and to show how different parameters affect our results using real data sets.

1 Introduction

Collaborative filtering (CF) is a recent technique for filtering and recommendation purposes. It has many important applications [1, 2] in E-commerce, direct recommendations, and search engines. With the help of CF, users can get recommendations about many of their daily activities. Using other users' data, CF systems try to predict how well an *active user* (a) will like an item that he/she did not buy before. The key idea is that a will prefer those items that like-minded users prefer, or that dissimilar users do not. Users might express their opinions about items they bought before or showed interest as *like (1)* or *dislike (0)*.

CF systems are advantageous; however, they fail to protect individual user's privacy and they have a number of disadvantages [1, 2]. The most important is that they are a serious threat to individual privacy. They pose various privacy risks [3] like unsolicited marketing, price discrimination, and being subject to government surveillance. Many users sometimes refuse to give data or might contribute false information. Customer data is a valuable asset and it has been sold when E-companies suffered bankruptcy [2]. By providing privacy measures, users feel comfortable to contribute more truthful information. The challenge is *how can people contribute their preferences about products for CF purposes without greatly compromising their privacy?*

[⋆] This work was supported by Grants ISS-0219560 and ISS-0312366 from the United States National Science Foundation.

In this paper, we propose a new scheme to achieve private recommendations using item-based algorithms on binary ratings. In our scheme, before sending data to the server, each user perturbs data in such a way that the server is not able to learn the true ratings. However, the perturbing scheme should still be able to allow the server to produce accurate referrals. We propose to use *randomized response techniques (RRT)* for data disguising. CF is based on aggregate values rather than individual data items. We hypothesize that it is possible to combine CF algorithms with the RRT to achieve users' privacy while still producing accurate referrals, because the aggregate data can be estimated with decent accuracy from disguised data if we have significantly large data. To verify this hypothesis, we implemented RRT for an item-based algorithm [11]. Using two existing data sets, we performed a series of experiments to evaluate the scheme's performance and to show how our results change with varying parameters.

2 Related Work

Canny [1, 2] proposes two schemes for privacy-preserving collaborative filtering (PPCF). A community of users can compute a public "aggregate" of their data without disclosing individual users' data. While his works focus on the peer-to-peer framework, in which users actively participate in the CF, our work focuses on another framework, in which users send their preferences to a server, which creates a model and provides referrals based on it. Polat and Du [8] apply randomized perturbation techniques (RPT) for PPCF. Although their schemes are based on numerical ratings and RPT, our work focuses on binary ratings and we employ RRT for data disguising. While they investigate how to provide private predictions using CF algorithms on user-user similarities, we focus on achieving private recommendations using item-based CF algorithms.

RRT were first introduced by Warner [12] as a technique to estimate the percentage of people in a population that has attribute A. The interviewer asks each respondent two related questions, the answers to which are opposite to each other. Using a randomizing device, respondents choose the first question with probability θ and the second question with probability 1-θ, to answer. The interviewer learns responses but does not know which question was answered. Sarwar et. al [11] propose item-based CF algorithms and present a model-based approach to pre-compute item-item similarity scores. After computing similarities, the best l similarities are retained (l is called the model size). The prediction is then computed by taking a weighted average of a's ratings on similar items.

3 Providing Private Predictions Using RRT

The algorithm proposed by [11] might be extended to provide referrals on binary ratings. In[7], referrals are provided on market basket data, where Tanimoto coefficient is used to find similarities between users. Polat and Du [9] extend Tanimoto coefficient to find user-user similarities on binary ratings. We propose the following scheme to provide predictions on binary ratings using item-based algorithms. The server finds item-item similarities using $W_{j_1 j_2} = \big[t(V_s) - t(V_d)\big]/t(V)$, where $t(V_s)$ and $t(V_d)$ are the number of similar and dissimilar votes for items j_1 and j_2, respectively and $t(V)$ is the number of co-ratings. Similarities range from -1 to 1. If $W_{j_1 j_2} > 0$, items are similar; otherwise they are dissimilar. When $W_{j_1 j_2} = 0$, they are not correlated. The server selects the best l similarities for each item and stores them. It receives a's known ratings and the target item q and finds a's ratings for those items that are the best similar items to q. It calculates the number of 1s (l_j) and 0s (d_j) among those ratings after it reverses the dissimilar items' ratings. It then computes $ld_j = l_j - d_j$. If $ld_j > 0$, then the item will be recommended as *like*, otherwise it will be predicted as *dislike*.

To further improve accuracy, significance weighting (SW) can be applied [6]. Herlocker et. al [6] add a correlation factor that can devalue similarity weights that are based on a small number of co-rated items. Since similarities are found between items, we apply SW based on number of users who rated such items. We applied $2c/n$ as a correlation factor, where c is the number of co-ratings and n is the number of users. If two items have less than $n/2$ co-ratings, we applied a significance weight of $2c/n$, otherwise a significance weight of 1 was applied.

A typical ratings vector includes the votes and empty cells for unrated items. An example of a ratings vector for user u is $V_u = (11 \perp 00 \perp 101)$, where $\perp$ means not rated. To perturb V_u, u generates a random number (r_u) using uniform distribution over the range [0,1]. If $r_u \leq \theta$, then u sends the true data, V_u. Otherwise, he/she sends the false data (exact opposite of the ratings vector), which is $\overline{V}_u = (00 \perp 11 \perp 010)$, where $\overline{V}_u$ is the vector that reverses the 1's in V_u to 0's and 0's to 1's; we call $\overline{V}_u$ the opposite of V_u. With probability θ, true data is sent while false data is sent with probability 1-θ. Although the server has the ratings vectors, it does not know whether they are true or false data.

3.1 Finding Private Predictions Using RRT

Without privacy as a concern, users send true ratings to the server, which can provide referrals as explained before. However, with privacy as a concern, the server should not be able to learn the true ratings of users including a. Users might send false data to accomplish perfect privacy but producing accurate predictions is impossible from this data. If they send actual data, finding accurate recommendations is possible but privacy is not preserved. CF systems should provide referrals efficiently. To achieve a good balance between accuracy, privacy, and efficiency, we propose to use both one-group and multi-group schemes.

One-Group Scheme. There are different RRT for data disguising. The one-group scheme [4], in which all ratings are put into the same group and all of them are either reversed together or keep the same values. The server cannot know whether users tell the truth or lie because the random numbers are only known by the users. In this scheme, we achieve the same accuracy on perturbed data with original scheme. The model, which includes item-item similarities, created from perturbed data is the same with the one created from original data because all ratings are either reversed together or keep the same values.

Although we achieve decent accuracy in this scheme, the privacy level is very low. If the server somehow learns the true rating for only one item, it can obtain true votes for all items. To further improve privacy, we propose to use multi-group schemes, in which the set of the items is partitioned into a number of groups; then the RRT are used to perturb each group *independently.* We can partition m items into m groups (m-group scheme), with each group containing only one item. For each group, users randomly decide whether to disclose its true rating (with probability θ) or to disclose the false rating (with probability $1 - \theta$). The users repeat this process for all groups; the random decisions are independent for each group. The m-group scheme is very secure. If the server can figure out one rating for any item, other ratings are still hidden. However, accuracy might become very low. A compromise between the one-group scheme and the m-group scheme is to partition the items into M groups, where $1 < M < m$. The decision is the same for all items in the same group, but the decisions for different groups are independent.

Multi-Group Schemes. Each user first groups the items in the same way. They then disguise their ratings in each group *independently.* Since the ratings in different groups are perturbed *independently*, even if the server knows information about one group, it will not be able to derive information about other groups. We improve privacy level by introducing multi-group schemes, while with increasing M, accuracy decreases because we add more randomness. Users disguise their ratings in each group as they do for one-group scheme based on random numbers and θ. They then send perturbed data to the server that needs to create a model by estimating item-item similarities from disguised data and storing the best l of them for each item. After model estimation, based on a's query, the server sends the estimated similarities for q to a who can compute predictions for q using them as explained previously. Since active users only need to send a query rather than their ratings, the server will not learn true preferences. To form the model on disguised data, the server should find a way to estimate item-item similarities from perturbed data.

The server does not know whether the collected data is true or not because users disguise their ratings based on the relation between random numbers, which are only known by them, and θ. However, it is possible to estimate the item-item similarities because the server is able to estimate the probabilities of having true or false data given disguised data. If we call the perturbed data Y_k, the true data X_k, and $\overline{X}_k$ represents the exact opposite of X_k or false data, where $k = 1, 2, \ldots, M$ and k shows the group name, the server needs to find $p(X_k|Y_k = X_k)$ and $p(\overline{X}_k|Y_k = X_k)$ for each group, where $p(X_k|Y_k = X_k) + p(\overline{X}_k|Y_k = X_k) = 1$. $p(X_k|Y_k = X_k)$ can be calculated using the Bayes' rule as follows:

$$p(X_k|Y_k = X_k) = \frac{p(Y_k = X_k|X_k)p(X_k)}{p(Y_k = X_k)} \tag{1}$$

where $p(Y_k = X_k|X_k)$ is θ. The value of $p(Y_k = X_k)$ can be calculated from disguised data while the value of $p(X_k)$ can be computed as follows using the facts that $p(Y_k = X_k|X_k) = \theta$ and $p(Y_k = \overline{X}_k|X_k) = 1 - \theta$:

$$\begin{aligned} p(Y_k = X_k) = p(Y_k = X_k|X_k)p(X_k) + p(Y_k = \overline{X}_k|X_k)p(\overline{X}_k) \\ p(Y_k = X_k) = \theta p(X_k) + (1-\theta)p(\overline{X}_k) \end{aligned} \tag{2}$$

Eq. (2) can be solved for $p(X_k)$ as follows using the fact that $p(X_k)+p(\overline{X}_k) = 1$:

$$p(X_k) = \frac{p(Y_k = X_k) + \theta - 1}{2\theta - 1} \tag{3}$$

We get the following after replacing $p(X_k)$ with its equivalent in Eq. (1):

$$p(X_k|Y_k = X_k) = \frac{\theta^2 + \theta p(Y_k = X_k) - \theta}{2\theta p(Y_k = X_k) - p(Y_k = X_k)} \tag{4}$$

Note that X_k and Y_k are ratings vectors of original and disguised data, respectively. Therefore, to find $p(Y_k = X_k)$, the server finds posterior probabilities for all items in each group k, selects the best one, uses it as $p(Y_k = X_k)$, and computes $p(X_k|Y_k = X_k)$ and $p(\overline{X}_k|Y_k = X_k)$ values for each group. It then can estimate item-item similarities. Since all items in the same group are either reversed or keep the same values, similarities for them will be same with the ones computed from true data, while similarities computed from perturbed data for those items in different groups will be different from those calculated from true data due to RRT. However, the server can estimate similarities for them. Note that there are four possible situations because the received data might be true or false in each group. The server first computes four similarity values by considering all four situations. Then, it finds the probabilities of having such situations. Finally, it multiplies those weights with the corresponding probability values, sums the results, and finds the similarity values between items in different groups. After estimating similarities, the server forms the model by selecting the best l similarities for each item and starts providing filtering services.

It is still possible to achieve decent accuracy using multi-group schemes in addition to one-group scheme. Although the number of true similarities decrease with increasing M, there are still $(m^2 - Mm)/(2M)$ true similarities because similarity weights for items in the same groups will be identical to the ones computed from original data. In addition, since such weights between items are computed over all users who commonly rated them, the aggregate values can be estimated from disguised data when there are enough users' data.

Our scheme can also be extended to provide private top-N recommendations. To find top-N recommendations, the server computes ld_j values for all a's unrated items, sorts them, and provides first N items to a as top-N recommendations. Since online computation cost is critical, instead of finding referrals for all unrated items, a sends a query stating he/she is looking for recommendations for N_a items, where $N < N_a < m - m_{at}$ and m_{at} is the number of rated items by a. The server then sends item similarities for those N_a items to a who can compute ld_j values for them and find top-N recommendations.

3.2 Providing Private Recommendations With Full Privacy

In addition to preventing the server from learning the ratings, it should not be able to learn items rated by the users. We can extend our scheme to achieve this goal. Before they disguise their data, users conduct the followings to prevent the server from learning the rated items. First, each user u finds the number of rated items (m_{ut}) and randomly creates a uniform integer m_{ur} from the range $(1, m_{ut})$. They then randomly select m_{ur} unrated items and fill randomly selected $\lfloor m_{ur}/2 \rfloor$ items' cells with 1 and the remaining unrated items' cells with 0.

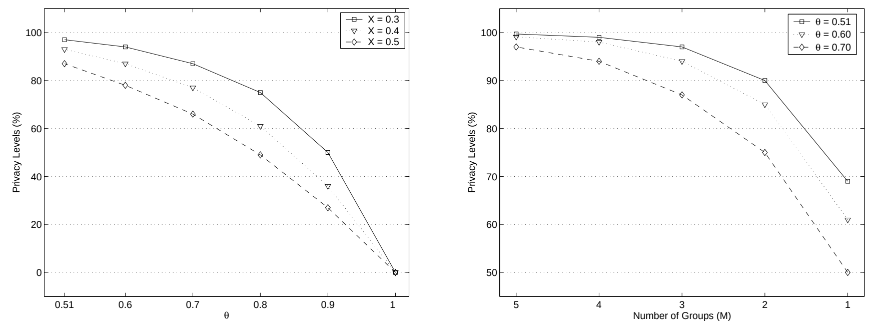
We can still provide accurate referrals while achieving full privacy because users fill empty cells with equal numbers of 1s and 0s. When there are enough users, the contributions of appended ratings to similarities will be close to zero. The changes in the number of similarly or dissimilarly rated items will be close to each other. The server does not know the rated items due to appended ratings. However, it can guess the randomly selected unrated items. The probability of guessing the number of randomly selected unrated items (m_{ur}) for the server is 1 out of $m'/2$, where m' represents the number of 1s or 0s, depending on which one is less, including the fake ratings for randomly selected unrated items. After guessing m_{ur}, the probability of guessing the $m_{ur}/2$ randomly selected unrated items filled with 1s is 1 out of $\mathrm{C}^{m'_1}_{m_{ur}/2}$ and the probability of guessing the $m_{ur}/2$ randomly selected unrated items filled with 0s is 1 out of $\mathrm{C}^{m'_0}_{m_{ur}/2}$, where m'_1 and m'_0 represent the number of 1s and 0s, respectively. Therefore, the probability of guessing the fake ratings is 1 out of $\big((m'/2)(\mathrm{C}^{m'_1}_{m_{ur}/2})(\mathrm{C}^{m'_0}_{m_{ur}/2})\big)$.

4 Overhead Costs and Privacy Analysis

Our scheme does not introduce additional communication and storage costs due to privacy concerns. Model creation is done off-line while referrals are computed and provided online. Our scheme's online component does not introduce overhead computation costs while off-line computation costs increase. However, off-line computation costs are not critical to the overall performance. Privacy can be measured with respect to the reconstruction probability (p) with which the server can obtain the true ratings vector of a user given his/her disguised data. Privacy level (PL) can be defined in terms of p as follows [10]: $PL = (1-p) \times 100$. With increasing p, privacy level decreases. To decrease p, the randomness should be increased, which makes accuracy worse because privacy and accuracy conflict each other. We can define p in terms of $p(X_k|Y_k = X_k)$ and M as follows:

$$p = \Big[p(X_k|Y_k = X_k)\Big]^M = \left[\frac{\theta^2 + \theta Y - \theta}{2\theta Y - Y}\right]^M \qquad (5)$$

where Y represents $p(Y_k = X_k)$. With increasing M, p decreases while PL increases. The value of p depends on θ, M, and the value of Y or X, where X represents $p(X_k)$. Since randomization process is conducted independently for different groups, PL increases with increasing M. When $\theta = 1.0$ or $\theta = 0.0$, we disclose everything about the original data. However, when θ is away from 1.0 or 0.0 and approaches to 0.50, PL increases because we add more randomness. We calculated PLs and showed them in Fig. 1. We varied θ from 0.51 to 1.0 because the complementary θ values achieve the same PL and found PLs for X being 0.3, 0.4, and 0.5, where we fixed M at 3. We then varied M from 1 to 5, fixed X at 0.3, and found PLs for θ being 0.51, 0.6, and 0.7. As expected, PLs increase with decreasing θ from 1 to 0.51 and increasing M values.



(a) Privacy Levels vs. θ (b) Privacy Levels vs. M

Fig. 1. Privacy Levels With Varying Parameters

5 Experiments

We used Jester and MovieLens (MLP) real data sets in our experiments. Jester data set [5] has 100 jokes and records of 17,988 users. The ratings range from -10 to +10, and the scale is continuous. MLP data was collected by the GroupLens Research Project (www.cs.umn.edu/research/Grouplens). It consists of ratings for 1,682 movies made by 943 users. Ratings are made on a 5-star scale. As evaluation criteria, we employed classification accuracy (CA), which is the ratio of the number of correct classifications to the number of classifications and F-measure (FM), where FM = (2 $\times$ precision $\times$ recall) / (precision + recall).

We transformed numerical ratings into two labels. We labeled items as *like* if numerical ratings were bigger than 3, or *dislike* otherwise in MLP. We labeled them as *like* if numerical ratings were above 2.0, or *dislike* otherwise in Jester. We randomly selected 2,000 and 800 training and 500 and 143 test users from Jester and MLP, respectively. Although we used the same test users throughout our trials, we used different numbers of training users based on various experiment settings. For each test user, we withheld 5 rated items' ratings, tried to find referrals for them, and compared them with true ratings. We ran data disguising

500 times and compute predictions on scrambled data. We compared referrals on perturbed data using our scheme with true ratings, calculated CAs and FMs for all test users, and displayed final values.

We hypothesize that accuracy and privacy depend on several factors including the significance weighting (SW), the model size (l), the number of users (n) and groups (M), the value of θ, and the number of appended ratings. We first performed testings to show how SW affects our results. We conducted trials with varying n values. We discarded such similarities calculated on fewer than 3 co-ratings. We found out that recommendation qualities become better when SW is applied for all n values because by applying it, we devalued such similarity weights calculated on limited number of co-ratings. CA is increased by 0.0045 with SW when n is 800 for MLP. We applied SW in the following experiments.

To find the optimum l, we performed trials using MLP while varying l values. For each l value, we ran the algorithm, found referrals for test items, compared them with true ratings, and calculated CAs and FMs. We ran this procedure 50 times and computed overall averages. With increasing l up to 500, the results are becoming better. However, after 500 best similarities, the results slightly become worse with increasing l. Therefore, we selected 500 as l for MLP. In the following experiments, we set l being 100 and 500 for Jester and MLP, respectively.

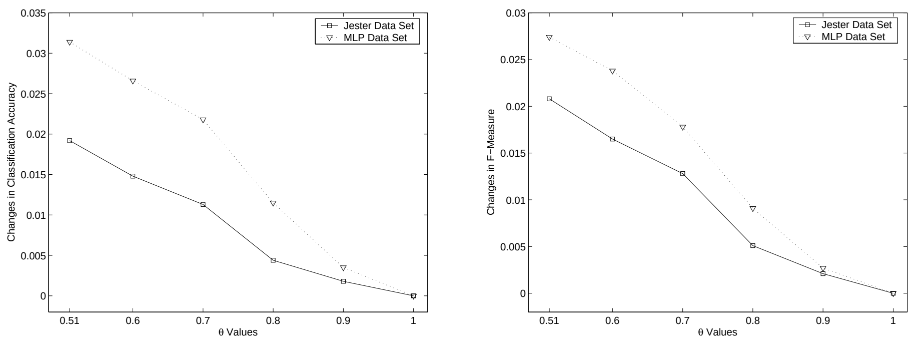
To show how data partition or various M values affect our results, we performed testings using both data sets with varying group schemes. We used randomly selected 500 and 200 training users from Jester and MLP, respectively, where we set θ at 0.65. We performed our experiments for up to five-group scheme, calculated CAs and FMs, and displayed the results in Table 1. As we expected, the results are becoming worse with increasing M because the bigger the M we use, the more randomness we add. For MLP, the loss in CA is 0.0338 when we changed M from 1 to 5 while it is 0.0188 for Jester.

Table 1. Recommendation Qualities With Varying M Values

	Jester				MLP			
M	1	2	3	5	1	2	3	5
CA	0.7095	0.6974	0.6962	0.6907	0.6813	0.6698	0.6580	0.6475
FM	0.6668	0.6550	0.6524	0.6455	0.7470	0.7391	0.7279	0.7158

Accuracy will be different for varying θ values because randomness differs with varying θ. We performed testings using both data sets, where we used 500 and 200 training users from Jester and MLP, respectively. We only showed results for three-group scheme, where we varied θ from 0.51 to 1.00, because complementary θ values give the same results. We compared the referrals estimated from disguised data with the ones computed from original data. We displayed CAs and FMs in Fig. 2. As expected, changes in accuracy are becoming worse while θ values are converging to 0.51. With decreasing θ values from 1 to 0.51, the randomness becomes bigger and causes losses in accuracy. Note that since all users send true data when $\theta = 1.0$, the changes in accuracy will be 0.

We hypothesize that we can still provide accurate recommendations with full privacy if we have enough users' data. Since false ratings are inserted into ratings vectors, referral qualities will decrease. To show how accuracy changes with appended ratings and varying n values, we performed experiments using both data sets. We appended ratings as explained in Section 3.2. We disguised users' ratings after inserting false ratings using θ being 0.65. We calculated CAs and FMs, and showed them in Fig. 3. As expected, recommendation qualities increase with increasing n. When there are more than 200 users, improvements become steady. However, recommendation qualities rapidly increases with increasing n when there are limited number of users. When ratings are appended, the results become worse compared to results without full privacy issues. However, the decrease in accuracy due to full privacy is small. For Jester, with increasing n, FM values based on appended ratings are becoming closer to the ones computed without appended ratings. For MLP data, when one-group scheme is used with n being 800, the accuracy lost is only 0.0057 due to appended ratings. It is 0.0067 for three-group scheme. Therefore, our scheme can be easily extended to achieve full privacy while still providing accurate referrals.

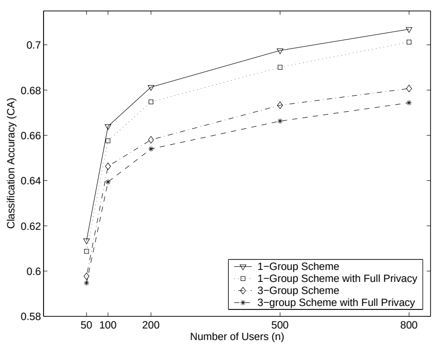


(a) Changes in CA (b) Changes in FM

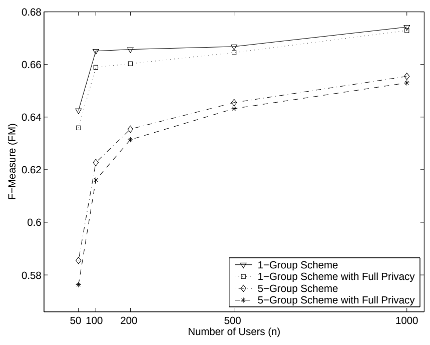
Fig. 2. Changes in Recommendation Qualities With Varying θ Values

6 Conclusions and Future Work

We presented a solution to achieve binary ratings-based private referrals on item-based algorithms using RRT. We showed that it is possible to provide accurate recommendations on perturbed data. Besides one-group scheme, we introduced multi-group schemes to achieve an equilibrium between accuracy, privacy, and efficiency. We will investigate whether we can achieve accurate referrals based on inconsistently disguised data. We will study how to extend our scheme to other recommendation algorithms.



(a) MLP Data Set



(b) Jester Data Set

Fig. 3. Recommendation Qualities With Varying n Values and Full Privacy

References

1. J. Canny. Collaborative filtering with privacy. In *Proceedings of the 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 ACM SIGIR Conference*, pages 238–245, Tampere, Finland, August 2002.
3. L. F. Cranor. 'I didn't buy it for myself' privacy and ecommerce personalization. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pages 111–117, Washington, DC, USA, 2003.
4. W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of the 9th International ACM SIGKDD Conference*, Washington, DC, USA, August 2003.
5. D. Gupta, M. Digiovanni, H. Narita, and K. Goldberg. Jester 2.0: A new linear-time collaborative filtering algorithm applied to jokes. In *Proceedings of the Workshop on Recommender Systems, ACM SIGIR'99*, Berkeley, CA, USA, August 1999.
6. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. T. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference*, Berkeley, CA, USA, August 1999.
7. A. Mild and T. Reutterer. Collaborative filtering methods for binary market basket data analysis. *Lecture Notes in Computer Science*, 2252:302–313, 2001.
8. H. Polat and W. Du. Privacy-preserving collaborative filtering. *International Journal of Electronic Commerce*, 9(4):9–36, 2005.
9. H. Polat and W. Du. Privacy-preserving top-N recommendation on horizontally partitioned data. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, Paris, France, September 19–22 2005.
10. S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.
11. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pages 285–295, Hong Kong, May 2001.
12. S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.