Privacy-Preserving Top-*N* Recommendation on Distributed Data

Huseyin Polat
Department of Computer Engineering, Anadolu University, Eskisehir, Turkey. E-mail: polath@anadolu.edu.tr

Wenliang Du
Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY. E-mail: wedu@ecs.syr.edu

Traditional collaborative filtering (CF) systems perform filtering tasks on existing databases; however, data collected for recommendation purposes may split between different online vendors. To generate better predictions, offer richer recommendation services, enhance mutual advantages, and overcome problems caused by inadequate data and/or sparseness, e-companies want to integrate their data. Due to privacy, legal, and financial reasons, however, they do not want to disclose their data to each other. Providing privacy measures is vital to accomplish distributed databased top-*N* recommendation (TN), while preserving data holders' privacy. In this article, the authors present schemes for binary ratings-based TN on distributed data (horizontally or vertically), and provide accurate referrals without greatly exposing data owners' privacy. Our schemes make it possible for online vendors, even competing companies, to collaborate and conduct TN with privacy, using the joint data while introducing reasonable overhead costs.

Introduction

Information overload has become a major problem for users with the amount of information available for individuals growing steadily (Billsus & Pazzani, 1998). Collaborative filtering (CF) is a recent technique for recommendation purposes and helps users cope with information overload using other users' preferences. The term *collaborative filtering* was first coined in the early 1990s (Goldberg, Nichols, & Oki, 1992). Collaborative filtering techniques are widely used in e-commerce, direct recommendations, and search engines to suggest items to users (Canny, 2002a, 2002b). With the help of CF, users can get recommendations about many of their daily activities.

Received May 9, 2007; revised December 27, 2007, accepted December 27; 2007

 DOI: 10.1002/asi.20831

The goal of CF is to predict how well a user (an active user, U) will like an item that he or she did not buy before, based on a group of users' preferences (Herlocker, Konstan, Borchers, & Riedl, 1999). The key idea is that U will prefer those items that like-minded users prefer, or that dissimilar users do not (Pennock, Horvitz, Lawrence, & Giles, 2000). The main assumption is that if user u_1 and user u_2 rate some items similarly, they share similar tastes, and hence will rate other items similarly (Goldberg, Roeder, Gupta, & Perkins, 2001). Collaborative filtering systems perform two tasks: They either provide predictions for single items or achieve top-N recommendation (TN), in which an ordered list of items that will be liked by U is provided.

To provide referrals, data collected from various customers is used. E-commerce companies collect customers' preferences about many products and create a database including ratings gathered from such users. They then employ CF algorithms to provide referrals based on the existing databases. However, some online vendors, especially the newly established ones, may have problems with available data and own a limited number of ratings. That is, it becomes difficult to generate reliable and truthful recommendations using a limited amount of data. Moreover, inadequate data causes cold start problem and restricts the recommender systems to offer referrals for only a limited number of products. Recommendations computed on such data might be unreliable and sometimes are not computable at all. Online vendors can produce better referrals if they share information about their customers with other vendors. Joint data is beneficial for e-commerce sites because customers prefer returning to stores with better referrals and they search for more products to purchase. Because customers receive more truthful and dependable recommendations, shared information is also advantageous to them.

Data collected for referral purposes might be horizontally or vertically distributed between different parties. In horizontal partitioning, different online vendors hold disjoint sets of

users' preferences for the same items, whereas in vertical partitioning, they own disjoint sets of items' ratings collected from the same users. Customers obtain products from different online vendors. For example, some people purchase books from Amazon.com, while others get them from Barnes & Noble .com. An individual's preferences for items might be split among different e-companies such as CDNow.com and Movie Finder.com. Combining horizontally distributed data (HDD) or vertically distributed data (VDD) is advantageous for CF systems and customers. Joint data makes it more likely to offer richer recommendation services because the amount of available data increases. Moreover, it is likely to provide more trustworthy and truthful predictions from joint data. A referral computed from the combined data is more likely to be precise and dependable than the calculated one from the disjoint data sets alone. Collaborative filtering systems help consumers find the products they want to buy at the e-commerce site (Sarwar, Karypis, Konstan, & Riedl, 2000). Online vendors should provide accurate referrals to their customers to avoid losing them because when products, which are not liked by customers, are recommended, this leads to angry customers who stop shopping at those sites. E-companies try to make more money by discovering ways to sell more products to customers (Sarwar et al., 2000). This can be partially achieved by providing more truthful and dependable referrals. Achieving such kinds of referrals depend on the available data collected for filtering services. Therefore, combining data augments available data and helps e-companies to increase their sales.

Mutual advantages due to collaboration between companies can arise from TN grounded on joint data. Data sharing might occur between online vendors, search engines, or even competing e-commerce companies, allowing data owners to provide richer referral services. TN qualities might be increased if data owners are able to combine their data. Therefore, TN on distributed data is vital. However, due to privacy, legal, and financial reasons, data holders do not want to collaborate and reveal their data to each other. Without privacy measures, e-companies are concerned about losing their data and profits. However, if privacy measures are provided, they can amalgamate their data. When privacy is protected, vendors feel more comfortable to collaborate. Providing privacy measures is vital to accomplish distributed databased TN. Therefore, we investigate the privacy-preserving top-*N* recommendation (PPTN) on distributed (horizontally or vertically) data problems.

In this article, we provide privacy-preserving schemes to solve the PPTN on distributed data problems, where data is distributed between various parties. However, for the sake of simplicity, we explain our approaches when data is distributed between two parties; and at the end of the article, we explore how to expand those two-party schemes to multiparty designs.

The organization of our article is as follows. We explain the related work in the next section; then we study PPTN on HDD or VDD in the Privacy-Preserving Top-*N* Recommendation (PPTN) on Distributed Data section. After explaining overhead costs analysis in the Overhead Costs Analysis section, we investigate security analysis in the succeeding section. In the Experiments section, we explore our experiments and discuss their results. We explore how to expand the proposed two-party methods to multiparty schemes in the Extension to Multiparty Schemes section. Finally, we summarize our results, conclusions, and further research issues in the last section.

Related Work

Top-N *Recommendation Algorithms*

Many top-*N* recommendation (TN) algorithms exist and examples include the algorithms proposed by Sarwar et al. (2000) and Mild and Reutterer (2001, 2003), to mention a few. Such schemes are based on market basket data, where users' preferences are represented by 1 if they bought the items or 0 otherwise. Market basket data is employed for mining association rules, which show correlation relationships among a set of data items.

Ratings are collected by e-companies for recommendation purposes. These ratings can be explicit ratings, where customers are explicitly asked to rate products they bought before; or implicit ratings, where a vote is obtained using different methods such as purchase records, repeated use, time to spend, etc. For a given customer, after forming the neighborhood with the best similar users, the purchase record of each neighbor is scanned through, and a frequency count is performed on the products they bought. The product list is then sorted and the most frequently purchased *N* items are returned as recommendations for *U* (Sarwar et al., 2000).

To form the neighborhood, similarities between the users in the database and *U* are computed; the best similar users are selected as neighbors. Mild and Reutterer (2001) and Mild and Reutterer (2003) use the Tanimoto coefficient, which can be defined, as follows, to find similarities:

$$w_{Uu} = \frac{t(y_U) \cap t(y_u)}{t(y_U) \cup t(y_u)}, \tag{1}$$

where $t(y)$ represents the number of elements in the basket y. Karypis (2001) discussed item-based TN, where he presented 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.

In information retrieval literature, there are other similarity measures that are applicable to binary vector inputs. In addition to the Tanimoto coefficient, similarity measures like cosine, overlap, dice, and so on, are defined over binary vector inputs (Bayardo, Ma, & Srikant, 2007).

Privacy-Preserving Data Mining

Privacy-preserving data mining (PPDM) is growing rapidly and the goal is to develop algorithms for modifying the original data so that the private data remain secret after the mining process (Verykios et al., 2004). The privacy-preserving measures should allow providing meaningful outcomes after

DOI: 10.1002/asi

conducting mining algorithms with privacy. For privacy, various techniques have been proposed and employed. Data involved in PPDM can be centralized or distributed (vertically or horizontally) among different parties.

Privacy-preserving data mining on distributed data. PPDM on distributed data has received increasing attention during the past years. Privacy-preserving naïve Bayes classifier for HDD is discussed in Kantarcioglu and Vaidya's (2003) work, where they assume that data is horizontally distributed. It is shown that using a secure summation and logarithm, distributed naïve Bayes classifier can be securely learned. Although their protocols are very efficient, they compromise a little on security. Privacy-preserving association rules on HDD are discussed in Kantarcioglu and Clifton (2004a), where they assume that data is horizontally distributed among three or more parties. Kantarcioglu and Clifton (2004a) address secure mining of association rules over HDD, while incorporating cryptographic techniques to minimize the shared data. Because cryptographic methods are used to achieve privacy, they introduce extra communication and computation costs. Kantarcioglu and Clifton (2004b) present a method for privately computing $k-nn$ classification from distributed sources without revealing any information about their data, other than that revealed by the final classification result.

Merugu and Ghosh (2003) present a framework for clustering horizontally distributed data in unsupervised and semisupervised scenarios, taking into account privacy requirements and communication costs. Vaidya and Clifton (2002, 2003, 2004) present privacy-preserving methods for different data mining tasks on VDD. They discuss privacy-preserving association rule mining (Vaidya & Clifton, 2002), where instead of using cryptographic solutions to scalar product, they give an algebraic solution that hides true values by placing them in equations masked with random values. Vaidya and Clifton explore *k*-means clustering (Vaidya & Clifton, 2003) and naïve Bayes classifier (Vaidya & Clifton, 2004) based on VDD, where cryptographic techniques are employed to accomplish privacy.

Privacy-preserving collaborative filtering (PPCF) and PPCF on distributed data. Canny (Canny, 2002a, 2002b) proposes two schemes for privacy-preserving collaborative filtering (PPCF). 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. His methods reduce the CF task to an iterative calculation of the aggregate, requiring only addition for the vectors of user data. He then uses homomorphic encryption to allow sums of encrypted vectors to be computed and decrypted without exposing individual data. His schemes are based on distributed computation of a certain aggregate of all users' data.

Polat and Du (2005a) employ randomized perturbation techniques (RPT) for PPCF. Before sending the private data to the server, users perturb their private data using RPT. The server collects disguised ratings from users, creates a central database, and starts providing CF services on the existing database. Their solutions make it possible for servers to collect private data from users without greatly compromising users' privacy requirements. Their experiments have shown that their schemes can accomplish truthful predictions while preserving individual user's privacy. In Polat and Du's (2006) work, providing private predictions using randomized response techniques (RRT) are discussed.

Zhang, Ford, and Makedon (2006) propose a two-way communication privacy-preserving scheme. They observe that perturbation is item-invariant in existing randomization approaches. They suggest that users should perturb their ratings for each item based on the server's guidance. They can help users disclose much less private information at the same accuracy level. Parameswaran (2006) presents a data obfuscation technique that provides robust privacy protection with minimal loss in usability of the data. Data obfuscation is used to modify the value of the data items without distorting the usefulness of the data. Parameswaran (2006) designs and implements a privacy-preserving shared CF framework using the data obfuscation algorithm.

Polat and Du (2005b) investigate how to achieve private TN on HDD. They assume that data is horizontally distributed between two parties; and show their scheme makes it possible to provide TN efficiently with decent accuracy. Achieving predictions using numerical ratings on VDD without greatly exposing data owners' privacy is discussed in Polat and Du (2005c). A solution to the PPCF on VDD problem has been provided. The solution makes it possible for two parties to conduct filtering services using their joint data without revealing their data to each other. The results have shown that the proposed scheme produces accurate referrals compared with the true ratings. In Kaleli and Polat (2007), how to provide predictions on distributed data using naïve Bayesian classifier-based CF schemes is discussed. They assume that data is distributed between two parties and offer predictions for single items. With increasing data, performance of naïve Bayesian classifier-based CF scheme degrades.

Privacy-Preserving Top-*N* Recommendation (PPTN) on Distributed Data

Binary Ratings-Based TN Algorithms

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 and recommendations are offered based on the number of liked and disliked items by the neighbors, rather than the frequency count. Today's TN schemes employ market basket data; however, purchasing and/or consuming products do not necessarily mean that consumers liked them. Customers purchase products they might like; sometimes, however, they dislike what they bought. As a

DOI: 10.1002/asi

result, recommendations may not be correct and dependable when calculated from market basket data. It is better to use binary ratings data, as binary ratings data are super set of the market basket data and ratings data contain more information than the transaction data.

We hypothesize that it is likely to provide more truthful and trustworthy referrals if binary data, showing users' tastes as like or dislike, is used. Therefore, we propose a scheme for binary ratings-based TN. Although market basket data is used to derive association rules between various items, deriving such rules is not available in binary ratings. However, our goal here is to show how to provide accurate TN, rather than association rules, with privacy.

It is essential to select the users who have high positive and high negative correlations with U as neighbors because U will prefer the products that like-minded users prefer, or that divergent users do not (Pennock et al., 2000). However, dissimilar users are not considered in TN processes proposed by Mild and Reutterer (2001, 2003). Accuracy might be increased if we select neighbors among the best similar and dissimilar users. Therefore, in our scheme, neighbors are selected among similar and dissimilar users. For this reason, we modify the Tanimoto coefficient, as follows, and employ it as similarity metric:

$$W_{Uu} = \frac{t(Y_s) - t(Y_d)}{t(Y)} \tag{2}$$

where $t(Y_s)$ and $t(Y_d)$ represents the number of similarly and dissimilarly rated items by users U and u, respectively; and $t(Y)$ is the number of commonly rated items by them. For example, if U'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 $W_{Uu} = (4-1)/5 = 0.6$, where NR means not rated. Similarities range from -1 to 1. If $W_{Uu} > 0$, users U and u are similar; otherwise they are dissimilar. When it is 0, they are not correlated at all.

As mentioned previously, there are other similarity measures for binary vector inputs. Although they can work for our proposed schemes as well because our focus is not the similarity measures and the Tanimoto coefficient works well for sparse binary datasets (Mild & Reutterer, 2003), we only consider the Tanimoto coefficient. Such measures mentioned in a previous section are basically variations of the Tanimoto coefficient and they work for binary inputs. Therefore, besides the Tanimoto coefficient, other measures can be employed in our proposed schemes.

After finding the similarities, we select neighbors using the threshold (τ_n) or the best-N_n methods to form the neighborhood. In the threshold, those users whose similarities satisfy the condition $|W_{Uu}| > \tau_n$, whereas in the best-N_n method, the best N_n users are selected as neighbors. Unlike the scheme defined by Sarwar at al. (2000), in our scheme, frequency count is not performed because users vote items as 1 or 0, and the neighbors are composed of similar and dissimilar users. We find the number of ones (l_j) and zeroes (d_j) in each item's column after we reverse the ratings of dissimilar users, where $j = 1, 2, \ldots, m$, and m is the number of items. If two users are similar or have a high positive correlation between them, they share similar tastes and vote the items similarly. If two users are dissimilar or have a high negative correlation between them, when one of them likes an item, the other one dislikes the same item. Resnick, Iacovou, Suchak, Bergstrom, and Riedl (1994) compute referrals using the best similar and dissimilar users' data by taking the weighted average of such users' ratings. We then compute $ld_j = l_j - d_j$. If $ld_j > 0$, then the item will be liked by U, otherwise it will not. After finding all items that will be liked by U, they are sorted according to ld_j values, and the first N items are returned as referrals. During the TN process, some computations can be done offline while others can be done online. Instead of finding referrals for all unrated items, U sends a query stating he or she is looking for referrals for N_U items, where $N < N_U < m-M$ and M is the number of rated items by U because online computation and communication costs are critical to the performance.

Achieving Private Recommendations on Distributed Data

Several PPDM on horizontally or vertically distributed data problems have been studied and investigated. Although PPCF on VDD and PPTN on HDD problems have been examined before, our work explores how to achieve TN primarily and to predict for a single item when data is distributed horizontally or vertically between various parties without violating their privacy. We also propose new protocols to achieve such tasks while finding a balance between accuracy, privacy, and efficiency. Although most of the schemes, proposed before to solve distributed databased problems, employ cryptographic techniques, which introduce significant overhead costs, our proposed schemes do not use such techniques and introduce reasonable additional costs.

Polat and Du (2005b) provide private predictions for single items using numerical ratings on VDD only, whereas our schemes achieve TN using binary ratings on horizontally or vertically distributed data with privacy. In the scheme defined in Polat and Du's (2005c) work, after finding a prediction for a single item using numerical ratings on VDD, a referral (like or dislike) is provided by comparing the prediction with a predefined threshold. To provide TN for several items, the procedure conducted for a single item should be done for those items involved in the TN process, which introduces additional costs. Our scheme allows data owners to provide private TN for several items using binary ratings on HDD or VDD. The scheme proposed by Polat and Du (2005c) requires some cryptographic techniques to achieve data exchange, whereas in our work here, to provide private referrals, we do not employ such methods; rather, we propose some protocols, which introduce reasonable overhead costs. Although Kaleli and Polat (2007) show how to provide predictions for single items on distributed data using naïve Bayesian classifier-based CF algorithms, providing TN is costly using their schemes. Our study here achieves both private predictions for single products and private TN efficiently without jeopardizing data owners' privacy.

DOI: 10.1002/asi

Although our work is the extension of the one presented by Polat and Du (2005b), the differences can be explained, as follows: First, besides presenting a solution to PPTN on the HDD problem, we also propose schemes to achieve PPTN on VDD. Second, unlike their work, where they assume that data is distributed between two parties, we present solutions when data is distributed by more than two parties and explain our solutions for two-party schemes. Third, data holders are able to provide CF services for more products while employing our VDD-based schemes. Finally, our work here allows data owners to find more reliable matching between users when they do not have enough commonly rated items.

Our schemes proposed in this article, generally speaking, are different from the ones mentioned in an above section. To offer referrals, we apply binary ratings, rather than market basket data or numerical ratings. Although they discuss providing private predictions on existing databases, our goal is to present accurate referrals on distributed data with privacy. We present solutions to achieve both TN and predictions for single items. We investigate both HDD- and VDD-based schemes with privacy.

If data owners desire to generate referrals using the combined data while preserving their privacy, we recommend schemes to perform accurate TN on binary ratings without divulging data owners' data. To present referrals, data collected from many users is used. Some online vendors, especially the newly created ones, might have problems with available data. They may own a limited number of users' and/or items' ratings. Holding a low number of users and items might cause a cold start problem and restrict the CF systems to providing referrals for only a limited number of products. That is, recommendations for some merchandise might not be computable at all. It becomes difficult to form large enough reliable neighborhoods due to not-enough users. Given an entire set of items, customers generally rate a few of them. That may cause low overlap between users' ratings. Then, it turns out to be hard to find reliable similarities. Predictions computed based on such neighborhoods and/or similarities may be inaccurate and unreliable. With increasing available ratings, it is more likely for recommender systems to produce more trustworthy and accurate referrals.

Combining distributed data is valuable for recommender systems and customers, especially when e-companies do not own enough ratings for dependable referrals. Recommendations computed from the combined data are more likely to be accurate than calculated ones from the disjoint data sets alone because joint data allows the parties to form more consistent neighborhoods and find reliable similarities. Joint data is valuable for e-companies because customers prefer returning to stores with superior recommendations. It is also advantageous to customers because they receive more truthful and trustworthy referrals. To offer richer recommendation services while providing more truthful recommendations to customers, more data and large enough number of neighbors and overlap between users are needed; this might be achieved by integrating distributed data.

With the increasing popularity of e-commerce, the number of online vendors is growing rapidly. To keep the current customers, e-companies should provide dependable referrals; this can be achieved when there is enough data. The recently established companies are more likely not to have enough ratings. It turns out to be not easy for them to compete with other vendors. Therefore, combining distributed data is useful. It is essential for e-commerce companies even competing online vendors to collaborate due to mutual advantages. Common advantages due to collaboration between parties can arise from TN, grounded on joint data. Combining distributed data may help e-companies to overcome difficulties caused by inadequate data and/or sparseness and improve referral qualities. Thus, TN grounded on distributed data is essential. However, due to privacy, legal, and financial reasons, online vendors do not want to combine their private data without privacy measures are in place. We investigate the PPTN on distributed data problem, which can be defined as follows: To maximize the mutual profits, online vendors want to provide referrals for their future customers using the combined data while preserving their privacy. The challenge is how to produce accurate referrals on joint data, without greatly exposing data owners' privacy?

Privacy in this context can be defined as follows: Those companies that want to combine their data should not be able to learn the true rating values of products and the rated items in each other's databases. However, they can guess the true ratings and the rated items with some probabilities. Such probabilities should not be bigger than some predefined threshold probabilities. We focus on corporate privacy rather than individual user's privacy. Because our concern is not protecting users' data, we assume that users directly send their ratings to the companies. Each company, due to financial, privacy, and legal reasons, does not want the other companies able to learn the ratings and rated items in its database.

Without privacy as a concern, online vendors can exchange their own data, create a central database, and provide filtering services using the combined data. *U* then sends his or her known ratings and a query to one of them, which computes referrals. With privacy as a concern, the companies should not be able to learn actual values of the ratings and the rated items held by each other. Because each party can act as an active user in multiple scenarios deriving information about other parties' data, the proposed protocol should be secure against such attacks, possible from all parties. They communicate through *U* during online recommendation computation. The challenge is how to provide TN services using distributed data without divulging data to each other. Because privacy, accuracy, and efficiency are conflicting goals, the proposed schemes to solve distributed databased TN with privacy problems should achieve a good balance among them. The attacks, "acting as an active user in multiple scenarios," can be simply described, as follows: One of the companies, wishing to derive data about the other parties' databases, can act as an active user in several times. That party employs the same ratings vector during all TN processes manipulating only one rating value each time.

DOI: 10.1002/asi

Because it gets some similarity values computed using its ratings and the users' ratings in the other parties' databases, the party acting as an active user can easily figure out the differences between similarities computed successively. Based on such differences, it is able to find out the ratings of the item for which the rating was manipulated. We study the PPTN on HDD and the PPTN on VDD problems in the following subsections.

PPTN on HDD. To make data sharing possible when data is horizontally distributed, the identity of the products should be established across the data holders' databases. This data exchange between vendors can be achieved offline.

Figure 1 shows PPTN on HDD, where A and B hold n_A and n_B number of users' ratings, respectively, of the same m number of items. They perform TN with privacy using the joint data, which is an $(nA + n_B) \times m$ matrix.

Because different neighbor selection methods follow different steps, we divide our proposed scheme into the threshold-based and the best-N_n-based schemes, and explain them in the following. Data owners need to exchange data to find referrals. One party should get all data required for recommendation computations; either party can act as a server or a master site to get required data and find the final referrals. They can also switch their roles. We assume that B acts as a server.

Threshold-based PPTN on HDD. In the threshold-based TN, users are selected as neighbors based on a predefined threshold (τ_n) value. Each party can compute similarities between users it owns and U. The parties then can select neighbors based on τ_n. B requests data from A and finds recommendations. The details are as follows:

1. U sends his or her ratings and a query (for which N_U unrated items he or she is looking for TN) to both parties.
2. A computes similarities between users it holds and U, 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. For this purpose, it generates a uniform random number ($r_{A\tau}$) from a range $[-\alpha, \alpha]$, 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.
3. It then calculates $ld_{Aj} = l_{Aj} - d_{Aj}$ values for all $j = 1, \ldots, N_U$ from the neighbors' data, and sends them to B through U. Because B does not know the τ_n value, the neighbors, which neighbors rated which items, and the values of l_{Aj} and d_{Aj}, it will not learn true ratings.
4. B finds similarities for users it holds, selects neighbors based on τ_n, and finds $ld_{Bj} = l_{Bj} - d_{Bj}$ for all $j = 1, \ldots, N_U$. It can now compute $\mathrm{l}d_j$ values by summing up ld_{Aj} and ld_{Bj} values. Finally, it finds referrals as explained before and sends the sorted item list to U.

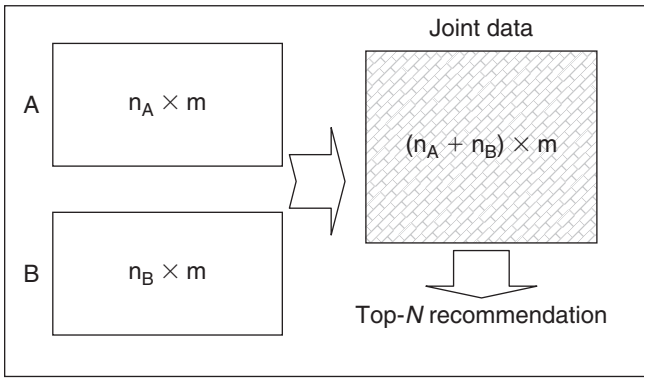

FIG. 1. Privacy-preserving top-N recommendation on horizontally distributed data.

Both parties can send the $\mathrm{l}d_{Aj}$ and $\mathrm{l}d_{Bj}$ values for all N_U items to U without sending it to each other; and U can find referrals. However, because U gets N_U recommendations rather than N, they exchange data and one of them provides recommendations to U.

Best-N_n-based PPTN on HDD. After similarities between all users and U are found, the best N_n users are selected as neighbors. A finds similarities between the users it holds and U, and sends $|W_{Uu_A}|$ values to B, where u_A represent the user A holds. B first finds similarities between the users it holds and U, and selects the best N_n users as neighbors. Because B can act as an active user in multiple scenarios to derive data, the scheme should not allow B to obtain data from similarities found by A. The scheme's details are as follows:

1. U sends his or her ratings and a query to A and B.
2. A estimates similarities between the users it holds and U using the private similarity computation protocol (PSCP), 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 $|W_{Uu_A}|$ values to B through U. B will not be able to learn true ratings due to Π_A and the PSCP. Moreover, it does not know the types of correlations between the users A holds and U. Due to permutation, B will not be able to learn which users' data is employed for recommendations.
3. B finds similarities for users it owns and selects the best N_n users among all n users as neighbors.
4. It then computes $\mathrm{l}d_{Bj}$ values for N_U items; and sends them along with the neighbors, selected among users A holds, to A through U. Because 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 find out true ratings owned by B.
5. A finds $1d_{Aj}$ and computes 1_{dj} values for all N_U items. It then calculates referrals and sends them to U. Although B acts as a server, to reduce communication costs, A directly sends recommendations to U.

PPTN on VDD. It would be difficult to find out whether two users from the different vendors refer to the same person or not when data is vertically distributed. This can be solved by using some unique identities provided by e-companies to

DOI: 10.1002/asi

customers for online shopping. The identities of users can be exchanged offline.

Figure 2 shows VDD-based TN with privacy, where two parties hold m_A and m_B number of items' ratings, respectively; collected from the same n users. They then perform TN using the joint data, which is an $n \times (m_A + m_B)$ matrix without unveiling their data to each other.

Our VDD-based scheme consists of offline and online computation phases. During offline computation, both parties compute default votes (v_ds) for the items they hold. v_ds or nonpersonalized predictions are used during the online computation step to accomplish the PSCP. Finding v_ds is relatively easy and can be found as follows because each party owns the entire data required for finding them:

1. Each party finds l_j and d_j values for all items $j = 1, \ldots, m_z$ they hold, where z is A or B.
2. Each party then finds $ld_j = l_j - d_j$ values for all items they hold. If $ld_j > 0$, v_d is 1, or 0 otherwise.

Because the corresponding data required to find v_ds is known by each party, the parties are not able to derive any information about each other's data. Moreover, they will not be able to find out the v_ds for the items held by each other because they do not exchange them. We investigate our scheme further, based on the neighbor selection methods. We also divide the scheme into two cases because the target items for the referrals are sought, might be held by different parties. In one case, all N_U items are held only by one party. In another case, they are split between such parties. Thus, we study two cases separately.

Case 1: All N_U items held by one party. In this case, all items for which U is looking for recommendations are held by one data owner. Such party receives all the required data from the other party, and performs final computations to find TN for N_U items. The details are, as follows, assuming that B holds all N_U items:

1. U sends his or her corresponding ratings to A and B; and a query to B because it owns those items.

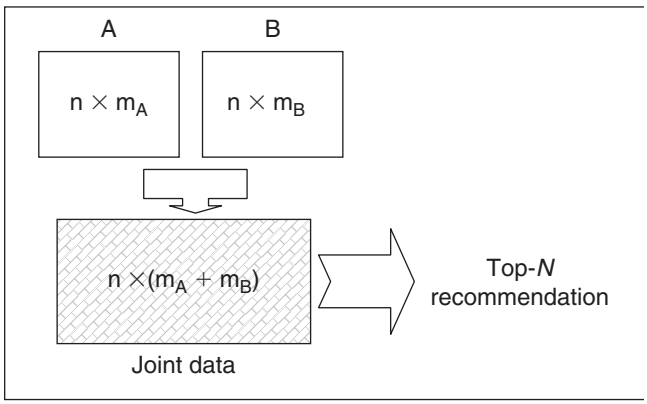

FIG. 2. Privacy-preserving top-N recommendation on vertically distributed data.

2. A computes the values required to find similarities between the users it holds and U using the corresponding data, because it owns ratings for m_A items. Because B can act as an active user in multiple scenarios to derive data, A uses the PSCP to prevent B from learning data about the ratings it holds.
3. After finding such values, they are sent to B through U. Due to the PSCP, B will not be able to obtain information about A's data. Like A, B finds the values required for similarities, between the users it holds and U, using the ratings it owns. It then computes the W_{Uu} values for $u = 1, \ldots, n$ by adding the values received from A to the ones it found.
4. After finding W_{Uu} values, B forms the neighborhood by selecting the best neighbors. To prevent A from learning τ_n and N_n values, B can use random τ_n and N_n values, rather than fixed ones. Therefore, it generates a uniform random number $r_{B\tau}$ from a range $[-\alpha_B, \alpha_B]$, adds that number to τ_n, finds $\tau_n + r_{B\tau}$, and uses it as a random threshold. A will not be able to learn the threshold due to the random number. It also creates a uniform random integer r_{BN_n} from a range $[-\gamma_B, \gamma_B]$, adds that number to N_n finds $N_n + r_{BN_n}$, and selects the best $N_n + r_{BN_n}$ neighbors to form the neighborhood. A will not be able to learn how many neighbors were selected to form the neighborhood due to random values, which are only known by B.
5. B finally computes ld_j values for N_U items and finds referrals as explained before; and sends the sorted item list to U. When A holds all N_U items, both parties follow the same steps, where A received data from B and finds recommendations.

Case 2: N_U items split between parties. In this case, the items for which U is looking for referrals are split between the data owners. After computing similarities, one company forms the neighborhood, calculates ld_j values for the target items it holds, and sends them along with the neighborhood information to the other party. Such party then finds ld_j values for the target items it holds, finds the sorted item list, and then sends it to U. Data exchange without greatly compromising privacy is challenging. The details are, as follows, where B acts as master site:

1. U sends his or her corresponding ratings and a query to A and B.
2. B computes the values required for similarities between the users it holds and U using the corresponding data, because it owns ratings for m_B items. To calculate such values, B uses the PSCP to prevent A from deriving data. Through U, B then sends such values to A that will not be able to learn actual ratings it holds due to the PSCP.
3. A finds the values to compute similarities between the users it holds and U using the corresponding data. It computes the W_{Uu} values for $u = 1, \ldots, n$ by adding values received from B to the values it found.
4. After finding W_{Uu} values, A forms the neighborhood by selecting the best neighbors. It employs the random τ_n values to select neighbors, while it applies the fixed N_n values. Since N_U items are split between A and B, B also needs the neighborhood information. Therefore, A tells which neighbors form the neighborhood and signs of

DOI: 10.1002/asi

similarities to B. Because A does not send the similarity values, B will not learn true ratings values, even if it acts as an active user in multiple scenarios. B will only know which users were selected as neighbors and the types of correlation they have with U.

5. Because the target items are split and all ld_j values are required to find the sorted item list for $j = 1, \ldots, N_U$, A should send ld_j values for the target items it holds to B, which can find sorted item list and provide TN to U. However, because B knows the neighbors and the types of correlations, it can obtain information about A's data in multiple scenarios. Therefore, A should calculate ld_{Aj} values in such a way to prevent B from learning the true ratings of the items held by A, accomplishing accurate referrals. After forming the neighborhood and sending the information to B, A can select those neighbors whose data will be used to compute ld_{Aj} values for the target items it holds, using random threshold and N_n values, rather than fixed ones. Because B does not know these random values, it will not learn how many neighbors were selected.

After receiving required data from A, B computes ld_{Bj} values for the target items it holds, computes $ld_j = ld_{Bj} + ld_{Aj}$, finds referrals as explained before based on ld_j values, and sends the sorted item list to U. If A acts as master party, both parties follow the same steps, where A finds referrals and tells U.

Private Similarity Computation Protocol (PSCP)

We propose to use the PSCP to find the similarities without exposing data holders' privacy. Because customers only buy and rate a few items, active users' ratings vectors are generally sparse. However, Becuse either party can act as an active user, they may use dense ratings vectors to obtain true information. We only explain the protocol for A because B also follows the same steps.

After A acquires U's data, it finds the number of rated items by $U(M)$. If M is less than, $[m/2]$, then A finds the items that U did not rate. A creates a uniform random integer R_{Aa} from the range $(1, m - M)$ and randomly selects R_{Aa} unrated items. It then fills those randomly selected unrated items' cells in U's ratings vector with the corresponding v_ds, calculated using the private default vote computation protocol (PDVCP), which is explained in the following section. If M is bigger than $[m/2]$, A finds the items that U rated and creates a uniform random integer R_{Ar} from the range $(1, M)$. It randomly selects R_{Ar} rated items and removes their ratings from U's ratings vector. A then forms U's new ratings vector and can estimate similarities using it. Because B does not know R_{Aa} or R_{Ar}, and randomly selected rated and unrated items, it will not be able to figure out W_{Uu_A} values from W'_{Uu_A} values, calculated based on a new ratings vector, even if it acts as an active user in multiple scenarios. 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 is wanted. Removing some of the ratings might cause a loss in accuracy because the number of available ratings decreases. When there are adequate ratings, we hypothesize that we can still estimate dependable similarities after removing some of them. Although the amount of data increases by appending v_ds, which are nonpersonalized predictions for U, some of them might not represent U's true preferences about the unrated products. However, it is still possible to estimate similarities using the PSCP after inserting v_ds into U's ratings vector.

Private Default Vote Computation Protocol (PDVCP)

In our scheme, finding v_ds is done offline, whereas other computations are conducted online. Because v_ds are used to find referrals, before providing referrals, data owners find v_ds offline, using the PDVCP, as follows:

1. Each party finds l_j and d_j values for all $j = 1, \ldots, m$.
2. A randomly selects $m_A = [m/2]$ 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 $d'_{Aj} = d_{Aj} + r_{Aj}$. Because $ld_{Aj} = l_{Aj} - d_{Aj} = l'_{Aj} - d'_{Aj}$, A finds ld_{Aj} values without using random numbers. Because 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 discover true ratings for item j. Even if it learns ratings for m_A items, it does not know ratings for the remaining $k = m - m_A$ items.
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 the m_A items. If $ld_j > 0$, v_d is 1, or 0 otherwise.
4. B finds $ld_{Bj} = l_{Bj} - d_{Bj}$ values for k items where $j = 1, \ldots, k$. It sends them along with v_ds for m_A items to A. Because A receives ld_{Bj} values for k items, it will not be able to learn ld_{Bj} values for m_A items.
5. A finds v_d values for k items and tells B. They then store them into $m \times 1$ matrices.

Overview of the Proposed Schemes

The overview picture of our proposed schemes is given in Figure 3. Generally speaking, data owners communicate through U during online TN computation. The overall picture varies when data horizontally or vertically distributed. Note also that due to different neighbor selection methods and various distributions of target items, the number of communications and operations and/or transformations might be different. We assume that A acts as a master site. The major operations and/or transformations conducted by data owners to achieve privacy are PSCP, PDVCP, permutation, random selection of the threshold value and the number of the best neighbors. As seen from Figure 3, U sends V_U and Q to both parties, where Q represents the query and V_U represents ratings vector or the vector containing corresponding ratings only to be sent to A or B. V_A represents the vector, which contains the required data to find TN recommendations by B. For different data distributions and cases, V_A might contain similarities, values needed to find similarities, neighbor

DOI: 10.1002/asi

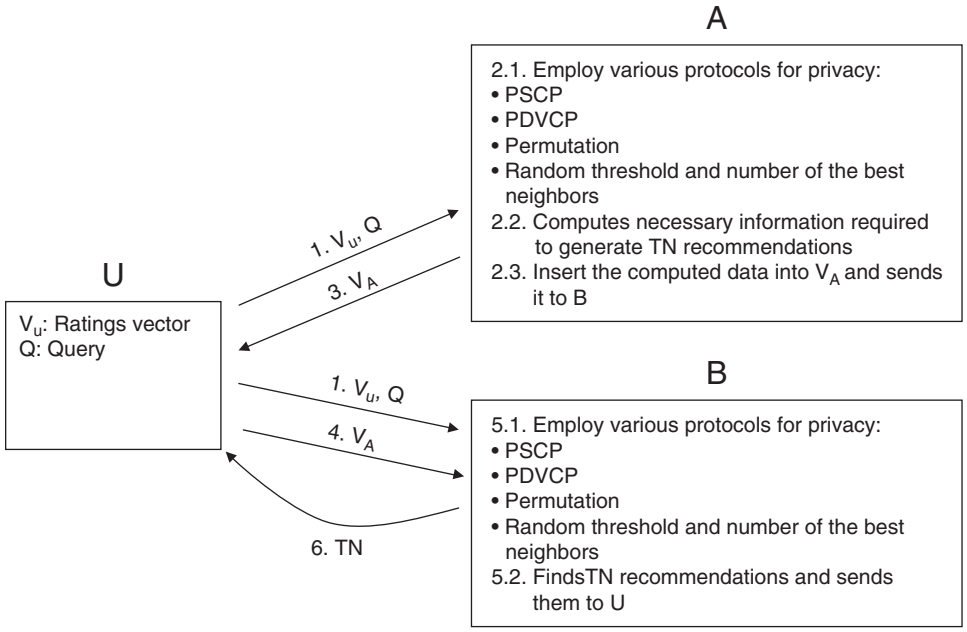

FIG. 3. Overview of the proposed schemes.

information, and so on. After collecting required data, the master site finally provides TN recommendations to U.

Overhead Costs Analysis

As explained previously, our proposed schemes for TN on distributed data consist of online and offline computation phases. Unlike the online computation, the offline stage is not critical to the overall performance. Therefore, in this section, we focus on the online computation costs, rather than offline costs. We also call attention to our proposed schemes' overhead costs due to privacy concerns.

Storage Cost Analysis

First, we elucidate the additional costs for HDD-based schemes. The only extra storage requirement is saving v_ds. The parties, A and B, save nonpersonalized ratings in two $m \times 1$ matrices. Therefore, the additional storage cost due to privacy concerns is relatively small, and it is O(m). Second, in VDD-based schemes, A and B save v_ds for items they hold into $m_A \times 1$ and $m_B \times 1$ matrices, respectively, where $m = m_A + m_B$. Therefore, the overhead storage cost is O(m) for VDD-based schemes.

Communication Cost Analysis

The communication costs during the online computation should be small because recommender systems should be able to offer recommendations to many customers within a short amount of time. Unlike off-line communication costs, overhead online communication costs due to privacy concerns should be minimized.

For HDD-based schemes, the number of online communications is two without privacy as a concern, while additional communication costs due to privacy are only 3 and 5 for threshold- and best-N_n-based schemes, respectively. Therefore, extra online communication costs introduced by our schemes are small. Such costs for VDD-based schemes are also very small. Because active users send their data along with a query and receive recommendations without privacy as a concern, the communication cost for the online phase is only 2. However, with privacy as a concern, communication costs increase because both parties communicate with each other through U.

For our VDD-based schemes, when the target items are held by one party, the number of communications during the online phase is 5, while it is 7 when they are split between parties. Although our schemes cause from 50 to 350% increase in the communication, since the baseline communication time is negligible, this is not a significant increase. During the communication between parties through U, the amount of information communicated is also imperative due to overall performance and practical reasons. In order to make it practical and reduce the cost, as mentioned before, we propose to find referrals for N_U items, rather than for all unrated items, where $N < N_U < m - M$ and M is the number of rated items by U. After finding similarities, the similarity vector, rather than the similarity matrix, is sent through U.

Computation Cost Analysis

We first investigate the additional computation costs for HDD-based schemes due to privacy concerns. In the threshold-based scheme, during the online step, the extra computation cost is negligible, because the overhead costs due to privacy concerns consist of creating a random number (r_τ) and conducting one more addition to find the random threshold. In the best-N_n-based scheme, one party uses the PSCP, which increases or decreases the number of comparisons by

DOI: 10.1002/asi

$R_a \times n_c$ or $R_r \times n_c$, on average, respectively; depending on M, where c is A or B. The same party also generates n_j random integers and utilizes a permutation function to permute similarities.

The overhead online computation costs for VDD-based schemes due to privacy concerns are also negligible. When one party holds the target items, it employs the PSCP, which increases or decreases the number of comparisons by $R_a \times n$ or $R_r \times n$, on average, respectively; depending on M, where n is the number of users. The same party also generates n random integers while employing the PSCP. The second party creates a random number or integer based on which neighbor selection method is used, and conducts one more addition. When the target items are split, one party applies the PSCP, which introduces extra costs, as explained above. The other party generates a random value for random τ_n. It also needs to randomly select neighbors whose data will be used to compute $\mathrm{l}d_j$ values, and generates N_c random integers, where N_c represents the number of target items held by that party.

Security Analysis

During the course of this work, we only consider the "acting as an active user in multiple scenarios" attack, which can be conducted by both parties. Other than this attack, competing companies or data holders can offer some incentives (discounts or coupons) or bribery to the users who provided data for CF services to them. Because both parties can bribe the same users to derive data from each other's database or to manipulate each other's data, the required data through such bribed users may not be true or trusted. These users can employ such offers against the other party to get more discounts or coupons. This kind of attack becomes expensive and the derived data becomes questionable and doubtful. Therefore, we assume that the companies do not try such attacks and we only consider the "acting as an active user in multiple scenarios" attack. Our proposed schemes should not allow the data owners to learn the true rating values and rated items in each other databases.

First, we investigate the security of the PSCP. We study the probability of guessing A's data for B, where B is acting as the master site. For one user, the probability of guessing R_a is 1 out of $(m - M)$, and the probability of guessing the correct R_a items is 1 out of $C_{R_a}^{m-M}$, where C_g^f represents the number of ways of picking g unordered outcomes from f possibilities. The probability of guessing the correct correlation type 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 $((M + R_a) \times (1 - |W_{Uu_A}|))$. The probabilities of guessing similarly or dissimilarly rated items are 1 out of $C_{t(Y_s)}^{t(Y)}$ and $C_{t(Y_d)}^{t(Y)-t(Y_s)}$, respectively. Thus, the probability of guessing the A's data is 1 out of $[2 \times (m - M) \times C_{R_a}^{m-M} \times (M + R_a) \times (1 - |W_{Uu_A}|) \times C_{t(Y_s)}^{t(Y)} \times C_{t(Y_d)}^{t(Y)-t(Y_s)}]^{n_A}$ due to the PSCP when v_ds are appended. The probability can be found similarly when ratings are removed.

Afterward, we explore the PDVCP, which is used in HDD-based schemes, in terms of security. We argue that our proposed protocol for finding v_ds is secure due to the following reasons. The parties send $\mathrm{l}d_j$ values for corresponding items to each other. Because 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. Furthermore, because they exchange data for half of the items, even if a party derives data about them, it will not be able to find out information about the remaining items. However, for B, the probability of guessing the correct l_{Aj} and d_{Aj} values after it gets $\mathrm{l}d_{Aj}$ for $j = 1, \ldots, m_A$ is 1 out of $(n_A - \mathrm{l}d_{Aj} + 1)$. The probabilities of guessing the like and dislike ratings of item j are 1 out of $C_{l_{Aj}}^{n_A}$ and $C_{d_{Aj}}^{n_A - l_{Aj}}$, respectively. Thus, the probability of guessing A's data for B is 1 out of $[(n_A - ld_{Aj} + 1) \times (C_{l_{Aj}}^{n_A}) \times (C_{d_{Aj}}^{n_A - l_{Aj}})]^{m_A}$ for m_A items. The probability for A can be found similarly.

Data collected for TN purposes might not be evenly distributed between various parties. One of the parties might own very few ratings. Although this may be the case, due to financial reasons, nobody may want to collaborate with that party. If it owns very little data compared to others' data, the gain due to its data will be very small. However, if this is the case, it is possible to derive data about that party's ratings by active user attacks in multiple scenarios. One solution to this problem can be explained as follows: The party that has limited number of ratings first finds item and/or user mean votes (column and row v_ds). It then creates imaginary items and/or users; and uniformly randomly selects some of their entries. It finally fills those cells with v_ds. In the same way, when an active user asks for a recommendation, it employs the same method to make the sizes of the active user ratings vector and its database consistent to achieve CF services. Because v_ds are used and they are computed based on few ratings, this might make accuracy worse.

Security Analysis of HDD-Based Schemes

Because our proposed schemes differ according to neighbor selection methods, we can study the threshold- and the best-N_n-based schemes separately. Our threshold-based scheme is secure due to the following reasons: Because $r_{A\tau}$ is only known by A, B will not be able to find out the random threshold. Moreover, B does not know how many users, or which users, were selected as neighbors, because A finds $\mathrm{l}d_{Aj}$ values after it selects the neighbors using the random threshold. Even if B finds out the neighbors, it would not be able to derive true ratings for items due to the same reasons for the PDVCP, as explained previously. Without random threshold, B can act as an active user in multiple scenarios to derive information about A's data from $\mathrm{l}d_{Aj}$ values. Note that A finds such values using those users' data whose similarity values bigger than the random threshold value. When A uses different random threshold values each time, B will not be able to learn the threshold, the number of users selected to compute $\mathrm{l}d_{Aj}$ values; and it cannot derive data from $\mathrm{l}d_{Aj}$ values even if it acts as an active user in multiple scenarios.

DOI: 10.1002/asi

Our best-N_n-based scheme is secure due to permutation and the PSCP, which is applied during the best-N_n-based scheme. We analyzed the PSCP in terms of security in the previous section. Because A permutes the best selected users, the probability of guessing the correct users for B is 1 out of n_A! Therefore, when v_ds are appended, the probability of guessing the A's data for B due to permutation and the PSCP is 1 out of $((n_A!) \times [2 \times (m - M) \times (C_{R_a}^{m-M}) \times (m + R_a) \times (1 - |W_{Uu_A}|) \times (C_{t(Y_s)}^{t(Y)}) \times (C_{t(Y_d)}^{t(Y)-t(Y_s)})]^{n_A})$. The probability can be found similarly when ratings are removed.

Security Analysis of VDD-Based Schemes

In VDD-based schemes, when the entire target items are held by one party, let's say B, the required data is sent to B by A. Because A employs the PSCP to determine the similarities, B will not be able to derive true ratings from estimated similarity values due to the previously mentioned reasons. A will not be able to find out the true ratings held by B even if it acts as an active user in multiple scenarios to get TN from B because B uses a random value for the threshold and N_n values. A does not know how many users and which users were selected as neighbors, which are only known by B. Even if A finds out neighbors, it will not be able to derive true ratings for items due to the same reasons for the PDVCP.

When the target items are split between parties, one party, let say B, estimates the similarities and sends them to A, which then estimates referrals for the target items and sends them to B, providing recommendations to U. A will not be able to derive information about B's data from estimated similarities due to the PSCP. On the other hand, B is not able to figure out target items' ratings held by A from ld_{Aj} values because A randomly selects the neighbors to compute such values. When A acts as an active user in multiple scenarios and gets recommendations from B, it will not learn true ratings of the target items held by B, as explained before, because the sorted list, rather than ld values, is sent to U by B.

Experiments

To evaluate the overall performance of our schemes, we carried out numerous experiments using real data sets while varying different parameters. Our goal here is to show how our privacy-preserving scheme based on distributed data behaves with various parameters, rather than evaluating it without privacy concerns. We performed experiments to show accurate our results are.

Data Sets

It is important to verify our proposed schemes empirically using different data sets. We have evaluated our protocols using two well-known real data sets, Jester and MovieLens Million (MLM), in our experiments. We have drawn empirical conclusions from these existing data sets. Jester is a Web-based joke recommendation system, developed at the University of California, Berkeley (Gupta, Digiovanni, Narita, & Goldberg, 1999). In Jester, users rate a core set of jokes, and then receive referrals about others. MLM data was collected by the GroupLens Research Project at the University of Minnesota (www.cs.umn.edu/research/Grouplens). MovieLens is a Web-based research recommender system that debuted in the fall of 1997 (Sarwar et al., 2000). Users visit MovieLens to rate and obtain referrals for movies. Jester includes continuous ratings, while MLM consists of discrete votes. The ratings in Jester range from −10 to 10; votes in MLM data sets range from 1 to 5. Jester has 100 jokes; MLM has 3,592 movies. Jester has 17,988 users, while MLM has 7,463 users. Jester has a data set that is much more dense than the MLM data sets because almost 50% of all possible ratings are present, whereas in MLM, each user has rated at least 20 movies. These data sets are publicly available for CF purposes. Although EachMovie data set (http://www.research.digital.com/SRC/EachMovie/) was among the data sets publicly available for referrals, it was shut down in October 2004 and it is no longer available. Our results based on these data sets can be generalized because they are representative of a set of systems, which are based on the ratings of items entered by users.

Evaluation Criteria

We measured accuracy using classification accuracy (CA), F-measure (F1), and coverage. CA is the ratio of the number of correct classifications to the number of classifications. F1 is a weighted combination of *precision* and *recall*, which are used for information retrieval tasks. F1 is defined as follows:

$$\text{F1} = \frac{2 \times precision \times recall}{precision + recall}, \tag{3}$$

where *precision* is the ratio of the number of liked items assigned to *like* class to the number of items assigned to *like* class, and *recall* is the ratio of the number of liked items assigned to *like* class to the number of liked items (Miyahara & Pazzani, 2002). Coverage is a measure of the percentage of items for which a recommendation system can provide predictions. A basic coverage metric is the percentage of items for which predictions are available (Herlocker et al., 1999). A low number of users and neighbors results in a low coverage.

Methodology

Because top-N recommendations are provided based on binary ratings, we first transformed numerical ratings of Jester and MLM data sets into binary ratings. Using the similar methodology conducted by Miyahara and Pazzani, (2002), we labeled items as 1, if the numerical rating for the item was bigger than 3, or 0 otherwise in MLM, while we labeled them as 1, if the numerical rating for the item was above 2.0, or 0 otherwise in Jester. We then randomly divided these data sets into training and testing. For this

DOI: 10.1002/asi

purpose, we randomly selected 9,000 and 6,000 users from Jester for training and testing, respectively. The MLM data set was also randomly divided into training and testing sets with 4,000 and 3,000 users, respectively. Two-thousand users were randomly selected for training among the 9,000 and 4,000 customers, while we randomly selected 500 users as test users among the 6,000 and 3,000. In the subsequent experiments, we used these 2,000 and 500 users for training and testing, respectively.

Experimental Results

We performed different sets of experiments to show how various factors affect our results.

Optimum threshold (τ_n) value. 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, in an attempt to get predictions for them, using our scheme while varying τ_n values from 0.05 to 0.4. We then compared predictions with the withheld ratings. We ran this procedure 100 times and computed the overall CA and F1 values for both data sets. We only showed CAs for both data sets in Figure 4, because F1s show asimilar trend with CAs.

As seen from Figure 4, the results are best when τ_n is 0.1 and 0.2 for Jester and MLM, respectively. Therefore, we selected them as the optimum τ_n values. For both data sets, the results are slightly worse when τ_n is away from its optimum value.

Optimum number of best neighbors (N_n). To show how accuracy changes with different numbers of best neighbors (N_n), we performed experiments using the same 2,000 and 500 users for training and testing, respectively. We again held 5 rated items from each test user's data, and tried to find referrals for them. To find the optimum value of N_n, we varied N_n from 50 to 1,900 for both data sets. After finding predictions using our scheme with varying N_n, we computed CAs and F1s by comparing them with the true ratings. We ran this procedure 100 times and found overall CA and F1 values. Because CA and F1 values show comparable trends, we only showed CAs for both data sets in Figure 5.

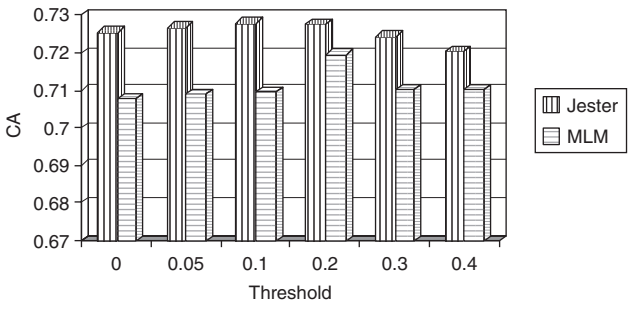

FIG. 4. Accuracy versus various threshold values.

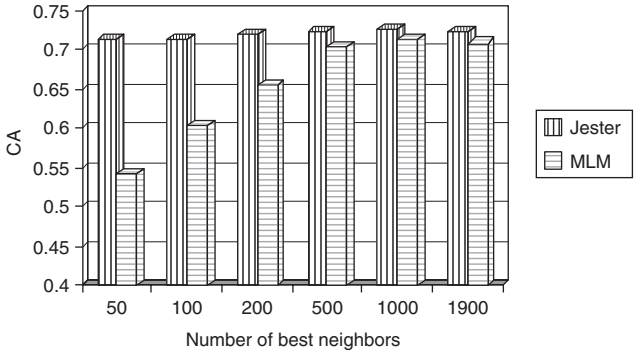

FIG. 5. Accuracy versus various number of best neighbors.

As seen from Figure 5, the results become better with increasing N_n, up to the best 1,000 neighbors, and they become worse beyond that. When N_n is small, the results are worse for MLM than the results for Jester. The reason for this can be explained by the sparseness of MLM. With decreasing N_n, available ratings become too limited for MLM, and then the accuracy of the recommendations computed based on them becomes worse. We set N_n to 1,000 in the subsequent experiments.

Coverage with varying amounts of combined data. CF algorithms are not able to provide referrals for all items due to the lower amount of available data. When there are a low number of users, it becomes difficult to find predictions for some items. Because HDD-based schemes integrate split data and makes it possible to have more users' data, it is more likely to find referrals for more items. To demonstrate how combining HDD affects coverage, we conducted experiments using MLM and Jester. We found coverage values for horizontally split data (n = 50, 100, 500, and 1,000) and combined data (n = 100, 200, 1,000, and 2,000). We illustrated the outcomes as percentages for only MLM for different τ_n values in Table 1.

As seen from Table 1, coverage increases when we integrate data because it becomes more likely to have items rated by more users. Therefore, integrating HDD makes it possible to find recommendations for more products. When we decreased the τ_n from 0.2 to 0.1, coverage increases because with decreasing threshold value, more neighbors are involved in recommendation computations. Because Jester is much denser than the MLM, for Jester, when n is bigger than 100, coverage reaches its maximum value, while it is 99.5% when n is 50.

Recommendation qualities with varying amounts of combined data. We hypothesize that accuracy improves when our HDD-based schemes are employed because such schemes

TABLE 1. Coverage (%) with varying amounts of combined data.

n	50	100	200	500	1,000	2,000
$\tau_n = 0.1$	41.8	57.3	67.2	76.9	81.3	87.4
$\tau_n = 0.2$	35.5	51.6	61.6	72.7	78.3	84.8

DOI: 10.1002/asi

TABLE 2. Recommendation qualities with varying amounts of combined data.

n	100	200	500	1,000	2,000
CA	0.6530	0.6726	0.6874	0.6970	0.7062
F1	0.7269	0.7480	0.7683	0.7769	0.7863

combine two disjoint sets of users. We carried out experiments to verify this hypothesis. With increasing n, it is more likely to find large enough neighborhoods to provide more accurate and dependable referrals. We used the threshold selection scheme for neighbor selection using the optimum τ_n value. We ran our experiments for horizontally split data (n = 100, 500, and 1,000) and combined data (n = 200, 1,000, and 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 calculated recommendations using both split and integrated data. We then compared predictions with true ratings, calculated CA and F1 values for both data sets, and showed results for only MLM in Table 2 because the results are similar.

As expected, with increasing amounts of combined data, the results become better because more users' data are involved in recommendation computations. Increasing available users helps to form large enough reliable neighborhoods. Therefore, combining HDD facilitates recommender systems to offer more truthful and trustworthy referrals.

Accuracy with varying numbers of appended ratings (Ra). In the PSCP, we either remove ratings from U's ratings vector or add default votes (v_ds) into U's ratings vector for randomly selected items based on M. To show how inserting or removing ratings, and varying R_a or R_r affects the overall performance of our distributed data-based TN schemes, we performed a variety of experiments while varying those values. We hypothesize that inserting default votes might increase accuracy because U's available ratings increase and makes it possible to find more reliable matching and truthful referrals. However, because v_ds might not match U's true preferences for those items, inserting them might decrease accuracy. We carried out experiments while varying R_a values using both data sets, and showed F1s for only MLM in Figure 6 because the results based on both data sets show similar trends. We used 2,000 training users, while 500 users who rated less than 60 items were randomly selected for testing. We employed the threshold neighbor selection method with the optimum τ_n value. 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%. Therefore, appending ratings increases accuracy up to some point but

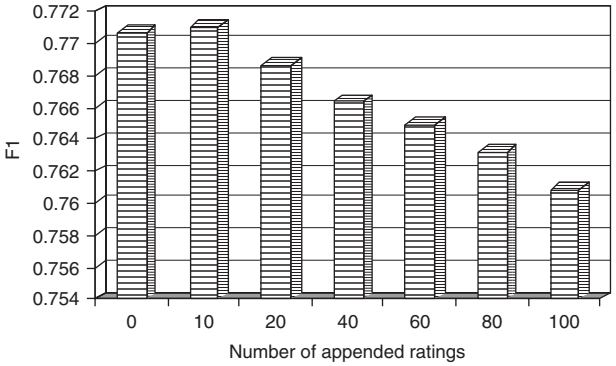

FIG. 6. F1 versus varying appended ratings.

makes it worse after that. Because accuracy loss due to appending ratings is very small, we can still achieve accurate referrals without greatly compromising data owners' privacy.

Accuracy versus varying numbers of removed ratings (R_r). To show how accuracy changes with varying R_r values, we conducted experiments using both data sets and showed CAs for Jester in Figure 7, while we demonstrated F1s for MLM in Figure 8. We used 2,000 training users, while 500 users who rated more than 80 and 200 items were randomly selected for testing from Jester and MLM, respectively. We then found top-10 recommendations for the randomly selected 10 rated items from each test user's data, while varying R_r values. For Jester, we varied it from 0 to 60, while we changed it from 0 to 160 for MLM. Recommendations for the withheld items were compared with the actual ratings.

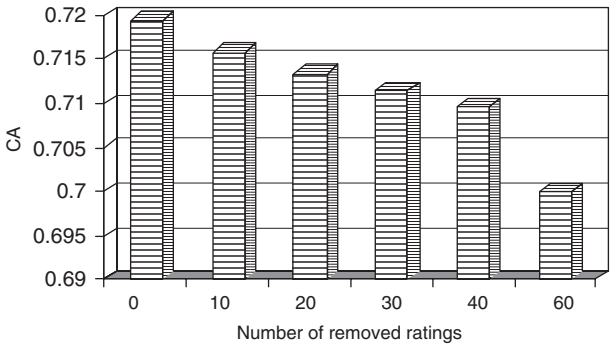

FIG. 7. CA versus varying numbers of removed ratings.

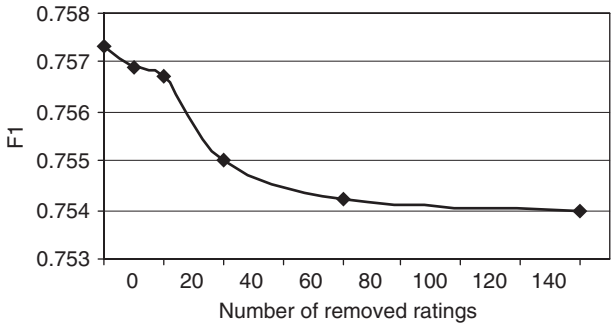

FIG. 8. FI versus varying numbers of removed ratings.

DOI: 10.1002/asi

As seen from Figure 7, when half of the ratings are removed, accuracy loss is only 1%, while it is 2% when R_r is 60. For MLM, with increasing R_r values, accuracy losses are even smaller compared to the losses we have in Jester. Because test users randomly selected from MLM own at least 200 ratings, while users selected from Jester have at least 80 votes, removing some of them does not affect the results with any significance. Although with increasing R_r results become worse as we expected, accuracy losses become smaller with increasing available ratings.

Accuracy versus varying random τ_n and N_n values. We proposed to utilize random threshold methods in our distributed databased TN schemes. When one party holds the entire N_U items, it employs random threshold and N_n values to form the neighborhood, and then provides recommendations based on it. In the case where the target items are split between parties, one party finds l*d* values for the items it owns, based on the neighborhood formed according to random threshold and N_n values. Therefore, we prevent parties from deriving information about each other's data. However, the results of our schemes become worse due to random methods for neighbor selection. For example, in the threshold-based schemes, we propose to use a random threshold. *A* selects τ_{Ar} values based on α_A. As seen from Figure 4, where we showed CAs with varying τ_n values, if τ_n is changed from 0.2 to 0.1 or 0.3 for MLM, 1% accuracy is lost. Because τ_{Ar} values are uniformly created, when α_A is 0.01, on average we lost 0.5% accuracy. For Jester, average accuracy losses due to random threshold are even smaller.

Threshold values determine the number of users whose data is used for TN generation. Accuracy changes with varying threshold values. For this reason, optimum threshold values are determined based on experiments. To achieve the best accuracy, optimum values should be used. However, parties are able to select threshold values different from the optimum ones due to privacy concerns by sacrificing little on accuracy.

As seen from Figure 5, because the changes in our results for both data sets are too small when we varied the optimum N_n to 500 or 2,000, the effects of using random N_n will be smaller. Our proposed methods for selection neighbors allow data owners to provide accurate referrals, while preventing them from deriving data about each other's data.

Recommendation qualities with varying amounts of combined items. It is more likely to improve accuracy by combining VDD. We hypothesize that the accuracy of our VDD-based schemes improves because the available amount of data (items) increases. To show how the results change with varying amounts of combined items, we performed experiments using MLM data because Jester has a limited number of items. For this purpose, we randomly selected 1,000 and 500 users for training and testing, respectively, and utilized the threshold-based method to select the neighbors setting τ_n to be 0.2. For test users, we selected the users who rated at least 20 items for each set of experiments. After finding referrals using our scheme with varying amounts of combined data, we compared them with true ratings, and calculated CAs and F1s. We found outcomes for vertically split data (m = 250 and 500) and combined data (m = 500 and 1,000). We show the results in Table 3. Using our scheme, we provided top-10 recommendations for randomly selected rated items from each test user's ratings vector.

TABLE 3. Recommendation qualities with varying numbers of combined items.

m	250	500	1,000
CA	0.5120	0.5390	0.6880
F1	0.4910	0.5430	0.7512

As seen from the table, the results are becoming better with increasing amounts of combined data. Although accuracy slightly improves when we combine data and increase *m* from 250 to 500, the results gain significant improvements when we increase *m* from 500 to 1,000. MLM data is very sparse and when we have a limited number of items, it becomes difficult to find the overlap between users and to form reliable neighborhoods.

As seen from our experiment results mentioned above, accuracy losses in our proposed schemes due to privacy protection measures are usually less than 1%. In Polat and Du (2005), accuracy loss due to randomized perturbation techniques is almost 9% for MLM on the correlation-based scheme. Randomized response techniques-based PPCF schemes presented by Polat and Du (2006) cause various losses up to 2% in accuracy due to varying data-disguising ways. The method proposed by Zhang et al. (2006) achieves the same accuracy level as with the one proposed by Polat & Du (2005). Therefore, accuracy losses due to our methods for achieving distributed data-based recommendations with privacy are less than the ones caused by the scheme proposed by Zhang et al. (2006). The experimental results show that the accuracy of CF engines remains nearly the same despite the data obfuscation process proposed by Parameswaran (2006). However, the utility of the ranking order is decreased due to data obfuscation and the error is about 5% on average. In conclusion, compared to the above-mentioned works, our schemes generally achieve higher accuracy. Moreover, our proposed schemes help companies provide more truthful and dependable recommendations.

Extension to Multiparty Schemes

We explained our proposed schemes for TN based on vertically or horizontally distributed data between two parties. Such schemes can easily be extended to multiparty schemes. For the threshold-based TN on HDD scheme, *U* sends his or her data and a query to all parties, which first compute similarities and find neighbors based on the random threshold, as explained before. They then select a master site among them. They communicate through *U* and send l*d* values for the N_U items to the master site, which finds recommendations. For the

DOI: 10.1002/asi

best-N_n scheme, each party other than the master site finds similarities as in the two-party scheme. They then send the similarities to the master site, which selects the best N_n neighbors, and tells each party the selected neighbors among their users. Such parties then compute ld values, and send to the master site, which can now provide recommendations. For the threshold-based scheme, overhead communication cost due to privacy concerns is $2d - 1$, where d is the number of companies, while it is $4d - 2$ in the best-N_n scheme.

For VDD-based schemes, protocols for multiparty depend on the target items. If they are all held by a single company, the company acts as a master party and provides recommendations. Each party, other than the master one, computes similarities as in the two-party scheme, and sends them to the master company. Then, the master site forms the neighborhood using neighbor selection methods based on the similarities and produces recommendations. In this case, the number of communications is $2y + 1$. U sends his or her data and a query to each company when the target items are split between different parties. Such parties then compute similarities as in a two-party scheme, and send them to the master site, forming the neighborhood based on the similarities, and tell other sites through U. The companies that hold one or more of the target items calculate ld values like in the two-party scheme, and the master site collects such data, providing recommendations. Overhead communication cost due to privacy concerns is $4d - 2$.

Conclusion and Future Work

We have presented solutions to accomplish private TN based on distributed data. Several PPDM on distributed data problems have been studied in the literature. However, our work here is the first to investigate the provision of recommendations based on distributed data while protecting data owners' privacy. Our proposed schemes make it possible for online vendors, even competing companies, to collaborate and conduct TN, using the joint data, without greatly compromising their privacy. Our proposed protocols (PSCP and PDVCP) allow the online vendors to perform private filtering services on joint data. The most important advantage of these protocols is that they make it possible to find equilibrium among accuracy, privacy, and efficiency. The e-companies are able to adjust the parameters of the protocols such as the number of removed or appended ratings to achieve the required level of accuracy and privacy. As explained previously, our schemes do not introduce significant overhead costs like storage, communication, and computation costs. It is possible to apply significance weighting to our schemes for improving accuracy. We explained the schemes when data is distributed between two parties. Moreover, we showed that our schemes can be extended to multiparty schemes. Predictions for single items can also be computed with privacy using our schemes. In that case, instead of finding predictions for N_U items, the system finds only one prediction for a single target item. Using our real databased experiment results, we have shown that our solutions provide precise referrals. As seen from our results, our schemes achieve more than 70% CA and F1 when there are enough available ratings of users on items. As explained previously, our schemes are secure against possible attacks coming from the data owners and allow them to provide private referrals efficiently.

Our proposed schemes also work for market basket data because market basket data is often represented in binary format. Therefore, when companies own market basket data represented in binary format, they are able to employ our schemes to achieve distributed databased CF services with privacy. Our schemes cannot be directly applied to numeric ratings. However, it is possible to transform such ratings into binary ones. Therefore, if online vendors have numeric ratings and want to use the proposed schemes to achieve distributed dat-based TN services with privacy, they can transform their ratings into binary ones and employ our schemes.

In our future work, we will study how aggregate data disclosure affects the accuracy and the privacy of our schemes. We will deeply explore multiparty schemes and show how accuracy, privacy, and efficiency vary with different numbers of vendors, involving recommendation computations. In addition to horizontally and vertically distributed data, we will study hybrid data partitioning-based TN with privacy. Another issue that may be considered is how our results are affected when there is an overlap between data held by different companies. Although we only consider the "acting as an active user in multiple scenarios" attack, there may be other attacks such as bribery or using incentives to derive data from users who provided data for recommendation purposes to online vendors. We will explore such attacks in detail. It is possible to provide TN services based on multicategory ratings. It is likely to improve our approach in such a way that to achieve TN using multicategory ratings. We will study whether we can accomplish accurate recommendations based on multicategory ratings with privacy. Although our aim is to evaluate the proposed schemes with privacy concerns using experiments, we believe that there remains work to be done experimentally to evaluate the proposed algorithm without privacy concerns and to compare collaborative recommendations with recommendations using only one of the data owners' data without privacy concerns. Finally, we will study how skewed distribution of data affects the overall performance.

Acknowledgments

This work was supported in part by Grant IIS-0219560 and IIS-0312366 from the U.S. National Science Foundation.

References

Bayardo, R.J., Ma, Y., & Srikant, R. (2007). Scaling up all pairs similarity search. In Proceedings of the 16th International Conference on World Wide Web (pp. 131–140). New York: ACM.

Billsus, D., & Pazzani, M.J. (1998). Learning collaborative information filters. In Proceedings of the 15th International Conference on Machine Learning (pp. 46–54). San Francisco: Morgan Kaufmann.

DOI: 10.1002/asi

Canny, J. (2002a). Collaborative filtering with privacy. In Proceedings of the IEEE Symposium on Security and Privacy (pp. 45–57). Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Canny, J. (2002b). Collaborative filtering with privacy via factor analysis. In Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval (pp. 238–245). New York: ACM.

Goldberg, D., Nichols, D., & Oki, B.M. (1992). Using collaborative filtering to weave an information Tapestry. Communications of ACM, 35(12), 61–70.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval, 4(2), 133–151.

Gupta, D., Digiovanni, M., Narita, H., & Goldberg, K. (1999). Jester 2.0: A new linear-time collaborative filtering algorithm applied to jokes. In Proceedings of the Workshop on Recommender Systems: Algorithms and Evaluation, 22nd Annual International ACM Conference on Research and Development in Information Retrieval (pp. 291–292). New York: ACM.

Herlocker, J.L., Konstan, J.A., Borchers, A., & Riedl, J.T. (1999). An algorithmic framework for collaborative filtering. In Proceedings of the 22nd Annual International Conference on Research and Development in Information Retrieval (pp. 230–237). New York: ACM.

Kaleli, C., & Polat, H. (2007). Providing naïve Bayesian classifier-based private recommendations on partitioned data. Lecture Notes in Artificial Intelligence, 4702, 515–522.

Kantarcioglu, M., & Vaidya, J.S. (2003). Privacy-preserving naïve Bayes classifier for horizontally partitioned data. In Proceedings of the IEEE ICDM Workshop on Privacy-Preserving Data Mining (pp. 3–9). Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Kantarcioglu, M., & Clifton, C. (2004a). Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Transactions on Knowledge and Data Engineering, 16(9), 1026–1037.

Kantarcioglu, M., & Clifton, C. (2004b). Privately computing a distributed *k*-nn classifier. Lecture Notes in Computer Science, 3202, 279–290.

Karypis, G. (2001). Evaluation of item-based top-*N* recommendation algorithms. In Proceedings of the 10th International Conference on Information and Knowledge Management (pp. 247–254). New York: ACM.

Merugu, S., & Ghosh, J. (2003). Privacy-preserving distributed clustering using generative models. In Proceedings of the 3rd IEEE International Conference on Data Mining (pp. 211–218). Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Mild, A., & Reutterer, T. (2001). Collaborative filtering methods for binary market basket data analysis. Lecture Notes in Computer Science, 2252, 302–313.

Mild, A., & Reutterer, T. (2003). 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.

Miyahara, K., & Pazzani, M.J. (2002). Improvement of collaborative filtering with the simple Bayesian classifier. IPSJ Journal, 43(11), 3429–3437.

Parameswaran, R. (2006). A robust data obfuscation approach for privacy-preserving collaborative filtering. Unpublished doctoral dissertation, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA.

Pennock, D.M., Horvitz, E., Lawrence, S., & Giles, C.L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (pp. 473–480). San Francisco: Morgan Kaufmann.

Polat, H., & Du, W. (2005a). Privacy-preserving collaborative filtering. International Journal of Electronic Commerce, 9(4), 9–36.

Polat, H., & Du, W. (2005b) Privacy-preserving top-*N* recommendation on horizontally partitioned data. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (pp. 725–731). Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Polat, H., & Du, W. (2005c). Privacy-preserving collaborative filtering on vertically partitioned data. Lecture Notes in Computer Science, 3721, 651–658.

Polat, H., & Du, W. (2006). Achieving private recommendations using randomized response techniques. Lecture Notes in Computer Science, 3918, 637–646.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J.T. (1994). GroupLens: An open architecture for collaborative filtering of Netnews. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (pp. 175–186). New York: ACM.

Sarwar, B.M., Karypis, G., Konstan, J.A., & Riedl, J.T. (2000). Application of dimensionality reduction in recommender system: A case study. In Proceedings of the ACM WebKDD 2000 Web Mining for E-commerce Workshop (pp. 682–693). New York: ACM.

Vaidya, J.S., & Clifton, C. (2002). Privacy-preserving association rule mining in vertically partitioned data. In Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (pp. 639–644). New York: ACM.

Vaidya, J.S., & Clifton, C. (2003). Privacy-preserving *K*-means clustering over vertically partitioned data. In Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (pp. 206–215). New York: ACM.

Vaidya, J.S., & Clifton, C. (2004). Privacy-preserving naïve Bayes classifier for vertically partitioned data. In Proceedings of the 2004 SIAM Conference on Data Mining (pp. 522–526). Philadelphia: Society for Applied Mathematics.

Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., & Theodoridis, Y. (2004). State-of-the-art in privacy-preserving data mining. ACM SIGMOD Record, 3(1), 50–57.

Zhang, S., Ford, J., & Makedon, F. (2006). A privacy-preserving collaborative filtering scheme with two-way communication. In Proceedings of the 7th ACM Conference on Electronic Commerce (pp. 316–323). New York: ACM.

DOI: 10.1002/asi