International Journal of Hybrid Information Technology
Vol.8, No.10 (2015), pp.27-40
http://dx.doi.org/10.14257/ijhit.2015.8.10.04

On Privacy-preserving Context-aware Recommender System

Yonglei Yao[1,2,*] and Jingfa Liu[1]

[1]*School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing 210044, China*
[2]*Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing 210044, China*
ylyao@nuist.edu.cn

Abstract

Privacy is an important issue in Context-aware recommender systems (CARSs). In this paper, we propose a privacy-preserving CARS in which a user can limit the contextual information submitted to the server without sacrificing a significant recommendation accuracy. Specifically, for users, we introduce a client-side algorithm that the user can employ to generalize its context to some extent, in order to protect her privacy. For the recommendation server, two server-side recommendation algorithms are proposed, which operate under the condition that only a generalized user context is given. The experimental results have shown that, using our approaches, the user context privacy can be achieved without a significant degradation of the recommendation accuracy.

Keywords: *context-aware recommendation, privacy, context, generalization*

1. Introduction

In recent years, Context-aware recommender systems (*CARS*s) [2] have drawn a lot of attention, because they can provide more compelling and useful recommendations, by taking contextual information into consideration. There are various approaches that incorporate contextual information into recommender systems and have offered improved performance in terms of mean absolute error [3], or prediction accuracy [4].

In a typical context-aware recommender application, to provide accurate recommendations, the server needs users' contextual information, such as location, time, companion, mood, activity, etc. However, this can have serious implications on user privacy [5, 6], as the contextual information might allow for identification of the user and her activities. Though context-aware recommendation is actively studied, it remains a challenging problem to limit the expose of user contextual information without significantly sacrificing accuracy of recommendation.

In the literature, secure multiparty computation (*SMC*) [10, 11] is a helpful technique that can compute over the encoded data without having knowledge of the original data. In recent years, *SMC* has been extensively studied and applied in many areas to address the privacy concerns, *e.g.,* Privacy-Preserving cooperative scientific computations [12], Privacy-Preserving Database Query [13], Privacy-Preserving Intrusion Detection [14], Privacy-Preserving Geometric Computation [15], and Privacy-Preserving data aggregation [16], to list a few. At first glance, it seems possible to directly apply *SMC* techniques to the area of context-aware recommendation. For example, we consider the following naïve solution: the user sends *N* contexts, which include her actual context and *N-1* random-generated contexts, to the server. For each context, the server computes a set of items. Finally, the user uses an efficient *1-out-of-N* oblivious transfer protocol to get the set of items computed according to her actual context. Because of the way that oblivious transfer protocol woks, the user can determine the set of preferred items, but the server could not learn which set of items the user chosen. However, *SMC* approaches

ISSN: 1738-9968 IJHIT

usually suffer from intensive computation and communication overhead, as shown in the above naïve approach, which includes *1-out-of-N* oblivious transfer as a building block that is communication and computation-intensive. This makes it unsuitable for scenarios where user devices are mobile phones and communicate with the server by a wireless link with limited bandwidth.

In this paper, we focus on enabling efficient yet privacy-preserving context-aware recommendation. Specifically, a context generalization-based privacy-preserving scheme is introduced. When querying the server, the user will send a generalized version of her contextual information, *i.e.,* a context segment that includes the actual context, not the actual context itself, to the server. Thus, the precision of user contextual information is deliberately reduced, and the server will recommend to the user with some uncertainty about user context. As a result, the context privacy of user is protected.

Our contribution can be summarized as follows:

1) We propose a context generation-based approach to protect user context privacy in context-aware recommender systems.

2) We formalize the context generation operation, and propose a context generalization algorithm, which can be used by a user to generalize her actual context when querying the server, in order to protect her privacy.

3) Given a generalized user context, two server side recommendation algorithms are proposed; one algorithm recommends compatible items while the other tackles items with conflicts.

4) We also conduct extensive experiments, and the results demonstrate the effectiveness and efficiency of the proposed solution.

The rest of the paper is organized as follows: Section 2 introduces related work, Section 3 describe the system model, the threat model, and the design goals. In section 4, the generalization-based privacy protection approach in context-aware recommender schemes is presented. Section 5 evaluates the performance of the proposed schemes, and section 6 concludes.

2. Background and Related Work

2.1. Context-aware Recommender System

Recommender systems (*RS*) are efficient tools designed to tackle the information overload problem by providing users with the most relevant content [1]. Starting with the collection of the initial set of ratings that is either explicitly provided by the users or is implicitly inferred by the system, a recommender system tries to learn a rating function *R*

$$R : User \times Item \rightarrow Rating \tag{1}$$

This rating function is used to predict the rating r_{ui} of a user *u* for a new item *i*, and recommend the highest-rated item (or *k* highest-rated items) for each user.

Recommendation Approaches are normally divided into two broad categories: content-based and collaborative filtering approaches [23]. Content-based approaches identify the common characteristics of items that have received a favorable rating from a user *u*, and recommend to *u* new items that share these characteristics. The key idea of the Collaborative filtering approaches is that, the rating of *u* for a new item *i* is likely to be similar to that of another user *v*, if *u* and *v* have rated other items in a similar way. Collaborative approaches outperform content-based ones in terms of most recommendation metrics, such as accuracy, diversity, and so on, and gain more popularity [1].

Collaborative filtering methods can be grouped in two general classes of neighborhood and model-based methods [23]. In neighborhood-based collaborative filtering, the user-item ratings stored in the system are directly used to predict ratings for new items, while model-based approaches use these ratings to learn a predictive model. Because of its

simplicity, justifiability, efficiency, and stability, neighborhood-based Collaborative filtering approaches are widely adopted and deployed [23]. We follow this approach in this paper.

Context-aware recommender systems (*CARS*s) [2] extend *RS*s by incorporating contextual information into the recommendation process as explicit additional categories of data. In a *CARS*, ratings are defined as

$$R : User \times Item \times context \rightarrow Rating \quad (2)$$

where context specifies the contextual information, which is usually multidimensional and includes location, time, mood, activity, and so on. We user r_{uic} to denotes the rating of user *u* on item *i* in context *c*.

The rating function *R* introduced in (2) is usually defined as a partial function, where only the initial set of ratings is known. Then, as usual in recommender systems, the goal of the recommendation is to predict the rating r_{uic} of a user *u* for a new item *i* in the context *c*, making the rating function *R* complete.

2.2 Privacy in Context-aware System

People have long realized the privacy issue in context-aware systems, and various privacy-preserving approaches have been proposed. Roughly speaking, these schemes can be categorized into four main types: Privacy Policies, Data perturbation, Anonymization and Lookup Notification [7]. However, these privacy management schemes are proposed at a general level, that is, they are introduced for all kinds of context-aware systems, but don't consider the special characteristic of context-aware recommendation problem. Thus, it is not possible to directly apply these privacy-preserving techniques in the area of context-aware recommendation.

A lot of research has been conducted on the specific issues of location privacy in Location-based Service [8, 24-26], aiming at preventing other parties from learning one's current or past location. But these works only focus on location, and omit other contextual information, such as time, activity, mood, etc. In contrast, context-aware recommender systems need to consider multi-dimensional contextual information.

There are also some works that address the privacy issue in recommender system [9, 19-21]. However, these work focus on user profile privacy, *i.e.*, protecting ratings of users. In contrast, our work will address context privacy and develops two schemes for user context protection in a context-aware recommender system.

3. Problem Formulation

3.1 System Model

Assume that there are *n* users and *m* items in a recommender system.

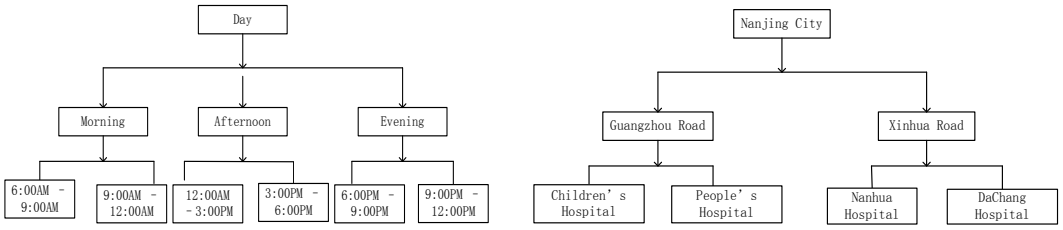
Assume contextual information is *l*-dimensional, and let $D_1, D_2, ..., D_l$ be contextual dimensions, such as location, speed, pollution level, *etc*. Thus, a specific user context is denoted as a *l*-dimensional vector $c = (c_1, c_2, ...c_l)$.

Furthermore, we use Ω_l to denotes the sample space of D_l, which may be in one of the following three forms:

1) A continuous range, for example, [*10, 200*] for speed;
2) A set with discrete numbers, for example, the sample space for Number of companion can be {*1, 2, 3, 4*};
3) Categorical data with non-numerical elements, for example, {*North, South, East, West*} for direction.

We assume that there is a hierarchical structure associated with each context dimension, which can be represented as a tree, where leaf nodes denote the most specific

values for the context dimension and intermediate nodes are more general values. This value generalization hierarchy of contextual information is popular in most of the context-aware recommender and profiling systems [2, 27, 28]. Figure 1 illustrates an example of value generalization hierarchies for the time and location domains.



(a) Time Hierarchy (b) Location Hierarchy

Figure 1. Examples of Value Generalization Hierarchies

Conceptually, ratings on the recommendation space can be stored in a *3*-dimensional cube. We divide this cube into a set of matrix. For each known context *c*, there is a matrix r_c which records the known ratings of all users for each item in that context. Formally, we have

$$r_c = (r_c^1, r_c^2, \ldots, r_c^n) \tag{3}$$

where each element is an *m*-dimensional vector recording the known ratings of a user for all items in this context.

We assume that, the server has recognized most related context factors and collected sample ratings under different contextual conditions, using some technique, such as a web-based approach in [17]. For each context, the ratings are stored in a matrix in the form of Equation (3).

For each item, the server has certain utility associated with it. The utility function is defined over the item space as the following

$$U : I \rightarrow Utilities \tag{4}$$

We use u_i to denotes the server's utility of item *i*.

Note that, there may exist conflicts between items. For example, company *A* and company *B* are competitors in the market as they produce the same type of products. They both post ads on the server, and each signs a contact with the sever which states that their product and the competitor's product can't be recommended to the user simultaneously. We will consider the two cases, items without or with conflict, in the next section.

3.2. Design Goals

For users, there are three goals: high-rating items, privacy, and efficiency.

Firstly, a user expects items that have high ratings for her in some specific context from the server. Secondly, the user would like to limit the amount of information about her context that is sent to the server and thus protect her privacy to some extent. Finally, since the user wants the recommendations fast and without draining much resource of his personal device, such as a Smartphone, the recommender system should be efficient in terms of both communication and computation cost. For communication efficiency, the user will restrict the number of items recommended to her by *k*, which determines the communication cost.

For the server, given a user *u* and her actual context *c*, the objective is to select a set of *k* items which will have high ratings for *u* in *c*, while at the same time, will potentially maximize its utility. That is,

$$A^* = \arg \max_{A \subset I, |A| = k} \sum_{i \in A} r_{uic} u_i \tag{5}$$

We define the product of user rating and server utility as the revenue of the item which is recommended to user *u* in context *c*. Thus, Equation (5) will select a set of *k* items, in such a way that the sum of revenues of these *k* items will be maximized.

When *k>1*, the recommendation is a problem of top-*k* recommendation problem. When *k=1*, the problem is transformed into a best item problem.

3.3 Threat Model

This paper focuses on thwarting attacks on breaching users' context privacy.

We assume that the server is *honest-but-curious*, which is consistent with most previous privacy protection schemes. That is, we assume that the server acts in an "*honest*" fashion and correctly follows the designated protocol specification to recommend high-rating items to users, but is "*curious*" to infer and analyze the message flow received during the protocol so as to learn additional information of users, that is, user contexts and other sensitive information that can be inferred.

4. Solution

4.1 Client-side Context Generalization

In this section, we introduce a context generalization approach. The intuition is that, by deliberately reducing the precision of the raw user context through extending it from a point context to a generalized context segment, user context privacy is protected to some extent.

4.1.1 The Concept of Generalization: For a user, to limit the information disclosure about context *c*, she generalizes her context according to the value hierarchies for all the context dimension and only sending the generalized version $\hat{c} = (c_1, c_2, \ldots c_l)$ to the server. For example, instead of sending the exact context value{*14:00, Car, 120km/h*} for *3*-dimension context {*time, transportation, speed*}, a user may send a generalized version {*afternoon, vehicle, 80-140km/h*} to the server. We denote the generation as $c \to \hat{c}$.

We first introduce the notion of *generalization distance* for a single context dimension, which is defined as the length of path from the actual contextual value to the generalized value in the corresponding value hierarchy. For example, for the time contextual value "*10:00AM*", the generalization distance is *1* if it is generalized to "*Morning*", and *2* if to "*Day*", according to the time hierarchy in Figure1(a).

Next, we introduce the notion of *generalization distance vector* for $c \to C$, $GDV = < gd_1, gd_2, \ldots gd_l >$, which is defined as a vector with each element as the generalization distance for the corresponding context dimension. For example, if we generalize the context (*10:00AM, Nanhua Hospital*) to (*Day, Xinhua Road*), the *GDV* is *<2, 1>*. In fact, the generalization distance vector quantifies the operation of generalization along each context dimension.

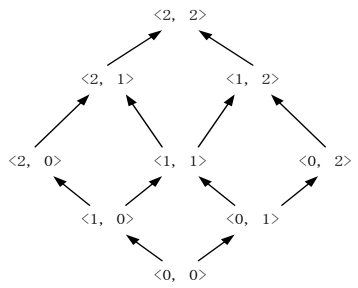


Figure 2. Multi-dimensional Generalization Lattice on Distance Vectors

Then, we define the *generalization hierarchy* for multi-dimensional context as a lattice on the corresponding generalization distance vectors, which defines the different ways in which a specific context can be generalized: Each path defines a possible alternative path that can be followed when generalizing contexts. Figure 2 illustrates the lattice representing the generalization hierarchy of 2-dimentioanl context (*Location, Time*), assuming the existence of value generalization hierarchies in Figure 1.

4.1.2 Measuring the Privacy: Obviously, for the user, there is a trade-off between privacy and quality of recommended items, as reduction of the context precision will degrade the quality of the recommended items.

To this end, we need to quantify privacy. In this paper, we will use entropy [22] to measure the level of privacy in the context generalization based scheme, as it is a proper tool for measuring uncertainty.

Considering a specific generalization $c \to \hat{c}$, where c= $(c_1, c_2, ...c_l)$ and $\hat{c} = (c_1, c_2, ...c_i)$. For context dimension $D_j (1 \le j \le l)$, c_j is the actual value for this dimension in the user context, and c_j is the generalized value reported to the server, which is actually the root node of a subtree which contains c_j in the value hierarchy for D_j . Let N_j denotes the number of leaf nodes of this subtree, and $p_t^j (1 \le t \le N_j)$ denotes the probability of each leaf node be the actual value for D_j , given c_j . The entropy related to D_j is

$$H(c_j) = -\sum_{t=1}^{N_j} p_t^j \log(p_t^j) \tag{6}$$

The number of potential specific contexts contained in $\hat{c}$, i.e., the possible set of contexts $c \to \hat{c}$, is:

$$N = \prod_{j=1}^{l} N_j \tag{7}$$

Let $P_t (1 \le t \le N)$ denote the probability that a specific context c is the user's actual context, given the generalized one, $\hat{c}$. The entropy regarding the generalized context is:

$$H(\hat{c}) = -\sum_{t=1}^{N} p_t \log(p_t) \tag{8}$$

We use this entropy to measure the privacy obtained by generalizing c to $\hat{c}$.

4.1.3 The Generalization Algorithm: Given a specific context tuple, and a privacy constraint, different generalization paths exist. Not all generalizations, however, can be considered equally satisfactory. For instance, according to Figure 1, generalizing (*10:00AM, Nanhua Hospital*) to (*Morning, Xinhua Road*) and (*10:00AM, Nanjing City*) achieves the same privacy, assuming a same probability of each potential context, because of the same entropy. But in the latter case, the time context gets no protection.

Input: Privacy constraint *pc*; a context <0, 0, ..., 0>, value generalization hierarchies

Output: a generalized context $\hat{c} = < gd_1, gd_2, \ldots gd_l >$

Method:

1. Initialize gd_i to 1, $i=1,2, \ldots, l$
2 Computing and storing entropy of each context dimension;
3. While entropy($\hat{c}$) < *pc*
  3.1 For each i∈[*1, l*]
    $gd_i=gd_i+1$;
    Computing the new entropy of *i*th context dimension;
    Computing the entropy variance of all dimensions, v_i;
    $gd_i=gd_i-1$;
  3.2 Finding *q* such that v_q is the minimal variance;
  3.3 $gd_q=gd_q+1$;
  3.4 Updating the entropy for the *q*th context dimension;
3. Return $\hat{c}$

Figure 3. MinVar: The Generalization Algorithm

Thus, intuitively, we expect the obtained privacy "*smoothly*" distributed across all the context dimensions. This can be achieved by making the entropy for each context dimension "*as close as possible*". We use variance to capture the notion of "*as close as possible*".

Figure 3 presents an algorithm, called *MinVar*. Given a specific multi-dimensional context *c* represented as a generalization distance vector with all elements to be *0*, a privacy constraint *pc*, and value generalization hierarchies for all contexts, the algorithm produces a generalized context $\hat{c}$ that is also denoted as a generalization distance vector.

Step1 of the algorithm initializes each generalization distance to be *1*, *i.e.*, minimally generalizes each context.

In Step 2, the algorithm computes entropy of each context dimension and stores all the entropies in a vector.

Step 3 begins a loop, if the privacy constraint cannot be satisfied. In each iteration, the algorithm tries all the minimal generalizations, each generalizing only one context value by substituting it by its direct parent in the value generalization hierarchy, and finds the generalization that minimizes the entropy variance of the generalization distance vector. Then the algorithm updates the corresponding generalization distance and entropy.

Finally, the algorithm returns the generalization distance vector that satisfies the privacy constraints. Note that, if a context needs not to be protected, its generalization distance keeps constant to be *0*.

Clearly, MinVar is an approach based on exhaustive search and makes no claim to be efficient with respect to complexity. However, the search space is not large. In each iteration, the algorithm only needs to enumerate *l* possible generalizations. As a result, the algorithm will be efficient in practice.

4.2. Server-side Recommendation

For the server, it needs to determine the best *k* items to be sent to the user, given only the generalized context $\hat{c}$. Suppose the server also knows the frequencies of the contexts, that is, the probability of each of the possible context $c \to \hat{c}$. With this uncertainty, what the server can do is to maximize the expected revenue. Thus, equation (5) is reformed as:

$$A^* = \arg \max_{A \subset I, |A|=k} \sum_{c \to C} pr(c) \sum_{i \in A} r_{uic} u_i$$
$$= \arg \max_{A \subset I, |A|=k} \sum_{i \in A} \sum_{c \to C} pr(c) r_{uic} u_i \quad (9)$$

4.2.1 Recommending Items without Conflicts: As noted earlier, there may or may not be conflict between items. We first consider the case that no conflict exists between items.

A greedy algorithm is presented in Figure 4, which simply computes the expected revenues in the generalized context $\hat{c}$, for all new items and sorts them, then chooses the top-*k* items. As in each step, the algorithm always selects the item that maximizes the expected revenue, from the set of all the items that have not been selected yet, the final item set of *k* items will maximize the sum of revenues. That is to say, the returned *k* items are optimal, from the point of view of the server, given the generalized context $\hat{c}$.

The algorithm involves computing *m* item revenues, and sorting *m* products. Thus, the complexity of the algorithm is $O(m \log m)$, assuming that quick sort or heap sort is used in sorting *m* products.

Input: a generalized context $\hat{c}$; all rating matrix
Output: a set *A* of *k* items
Method:
1. Initiate $A = \Phi$
2. For each $e \in I$, computes
$$ru_e = \sum_{c \to \hat{c}} pr(c) r_{uec} u_e$$
3. While $|A| < k$
$$A = A \cup \arg \max_{e \in I/A} ru_e$$
4. Return *A*

Figure 4. Algorithm for Recommending K Items without Conflicts

4.2.2. Recommending Items with Conflicts: When conflicts exist between items, the recommendation problem becomes more complex.

Proposition 4.1 For a generalized context $\hat{c}$, it is *NP-Hard* to select a set of *k* (*k>1*) items A^* to maximize the expected revenue, from a set of *m* (*m>k*) items with conflicts existing between some items.

Proof: We prove this by a reduction from the Independent Set problem. In the decision version of the Independent Set problem, we are given a graph *G* and a number *k*. Our goal is to determine ***whether*** G contains an independent set of size *k*.

Consider an arbitrary *G=(V, E)*,where *V* is the vertex set and *E* is the edge set, and a number *k*. For each vertex v_t, we construct an item it, and set $r_t=1$ for all users in all contexts and $u_t=1$. Thus, ***each*** item will have the same revenue to the server. For each edge (v_s, v_t), we construct a conflict between it and is. Thus, if we can find *k* items without conflicts, we also find an independent set containing *k* nodes in *G*. Since the *Independent Set* problem is *NP*-Complete, the item selection problem is *NP-hard*. This ends the proof.

Input: a generalized context $\hat{c}$; all rating matrix; item subset I_s $(1 \le s \le k)$
Output: a set A of k items
Method:
1. Initiate $A = \Phi$
2. For each I_s

$$A = A \cup \arg\max_{e \in I_s} \sum_{c \to \hat{c}} pr(c) r_{uec} u_e$$

3. Return A

Figure 5. Recommending *k* Items with Conflicts Existing only in each Subset

This problem cannot be approximated efficiently for the worst case. Thus, we only consider some special ones.

For example, consider the special case that, all the items are divided into *k* subsets, and conflicts only exist between items in the same subset; for items from different subsets, there is no conflict. When recommending items to a user, the server simply choose a single item with the expected highest revenue from each subset, and then returns k highest ones to the user.

Many real-world recommendation problems fall into this category, *e.g.*, recommending a suit of equipments for a user who wants to camp, in which the server needs to select one single item from every incompatible set to form an equipment suit.

The algorithm for this special case is shown in Figure5.

Of course, there are other special cases that can be solved efficiently. We don't enumerate them in this paper.

4.2.3. Predicting Item Ratings: When recommending items for a user *u* in a generalized context $\hat{c}$, the server needs to estimate the ratings for a new item *i* in each possible context $c \to \hat{c}$, that is, to compute r_{uic}.

In this paper, we use user-centric neighborhood-based collaborative filtering [19] to predict the rating of a user for a new item, taking context into consideration. User-centric schemes evaluate the interest of a user *u* for an item *i* using the ratings for this item by neighbors, which have similar rating patterns. A neighbor of user *u* is typically the user *v* whose ratings on the items rated by both *u* and *v* are most correlated to those of *u*. Because of its simplicity and efficiency, this scheme is extensively deployed [1, 2].

Formally, we have

$$r_{uic} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vic}}{\sum_{v \in N_i(u)} w_{uv}} \tag{10}$$

Where $N_i(u)$ denotes the set of *u*'s neighbors who have ratings for item *i*, and w_{uv} denotes level of similarity between *u* and *v*.

We use Pearson *Correlation* (PC) similarity to calculate w_{uv}:

$$w_{uv} = \frac{\sum_{c} w_{uvc}}{|c|}$$

$$w_{uvc} = \frac{\sum_{i \in I_{uv}} (r_{uic} - \overline{r_{uc}})(r_{vic} - \overline{r_{vc}})}{\sqrt{\sum_{i \in I_{uv}} (r_{uic} - \overline{r_{uc}})^2 \sum_{i \in I_{uv}} (r_{vic} - \overline{r_{vc}})^2}} \quad (10)$$

Where I_{uv} denotes the set of items that both *u* and *v* have rated, and $\overline{r_{uc}}$ denotes the average rating of user *u* for all the items in I_{uv} in context *c*. The similarity of *u* and *v* is calculated by averaging their similarities in all the contexts, and for each context, we compute their similarity by *Pearson Correlation* (*PC*) similarity. Note that, similarities between users can be computed, updated and saved by the server in an offline manner.

5. Experiments and Analysis

5.1. Dataset

Currently, as rating data which considering context was not available in any of the existing systems, we were not able to use any publicly available data. Instead, we follow the approach in [17]. We set up a web site and asked users to rate some *Places of Interest (POIs)* under different contextual conditions. We considered a set of 28 *POI*s in Nanjing, and collected user ratings, which range from *1* to *5*. We consider two contextual factors, namely, *location* and *time*.

The web site was made accessible to a group of students who were asked to navigate and choose POIs during a period of four weeks. In an interview, a single POI is considered and four ratings are requested: how likely is the user to visit the POI, assuming that four alternative contextual conditions are given. In each interview, the web page provided a short introduction of the POI, the information about the distance between the POI and the user's current location, and the time available which is the difference between the closing time of the POI and current time.

We collected such ratings from over a hundred of students in Nanjing University of Information Science and Technology, and acquired a total of over 3000 ratings for the 28 POIs.

The utility of each item for the server is drawn uniformly at random from the range [1, 10]

5.2. Metrics

We established a value generalization hierarchy for location and time, respectively. The time hierarchy has the same structure as shown in Figure 1(a), while the location hierarchy has a complex but similar structure as shown in Figure 1(b).

In our scheme, to protect user privacy, user context is deliberately generalized, and the precision of the context information is reduced. As a result, the quality of the recommendation will be degraded. To evaluate this scheme, we employed two metrics.

Firstly, the predictive accuracy metric *MAE* (*Mean Absolute Error*) [22], the most widely used metric, is used as an evaluation metric. We extend the computation of *MAE* by taking context into consideration.

$$MAE_c = \frac{1}{|R_{test}|} \sum_{r^a_{uic} \in R_{test}} \left| r^p_{uic} - r^a_{uic} \right| \quad (12)$$

Where R_{test} is a test set which contains part of known ratings and is used to evaluate the prediction accuracy, and r^p_{uic} and r^a_{uic} denote the predicted and the actual rating, respectively. Note that, *MAE* only consider user ratings, but not server utilities

Secondly, taking server utility into consideration and focusing on the item set recommended to the user, we introduce a metric *MRE* (*Mean Recommendation Error*) as following:

$$MRE_c = \frac{1}{k}\left|\sum_{i \in A^*} r_{uic} u_i - \sum_{j \in A^o} r_{ujc} u_j\right| \quad (13)$$

A^* denotes the set of k items that the server recommends to the user, under the condition that only a generalized context $\hat{c}$ is given, which generalizes the actual context c. A^o denotes the optimal set of k items that maximizes server revenue, which is computed with the actual context c. Thus, *MRE* captures the difference between the expected revenue given a generalized context and the true revenue if the actual context c is given.

5.3 Results

The dataset is divided into two sets, namely, the training set and the test set. The training set containing 4/5 and the test set 1/5 of the whole dataset.

Firstly, we evaluate the generation-based privacy protection approach by the *MAE* metric. Using ratings in the training set, we predict ratings in the test set, and compute the average *MAE* over all the contexts.

The *MAE* for different privacy levels achieved is shown in Figure 6. The privacy level is measured by entropy, as noted in section III. As shown in Figure 6, when we increase the privacy from 0, *i.e.*, without context generalization, to $\log 24$, the *MAE* also increases. This is consistent with our intuition that, with the increased privacy achieved, the service quality will drop. Overall, as Figure 6 shows, the *MAE* is not great. This can be explained by the fact that our rating data is not so sparse.

Secondly, we evaluate the privacy protection approach by the *MRE* metric, under varying constraints. As stated above, the *MRE* metric captures the difference between the expected revenue given a generalized context and the true revenue if the actual context *c* is given.

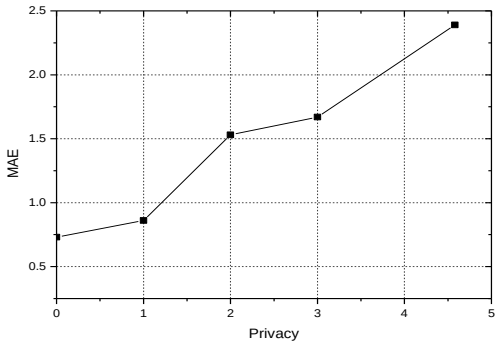


Figure 6. *MAE* with different Privacy Level

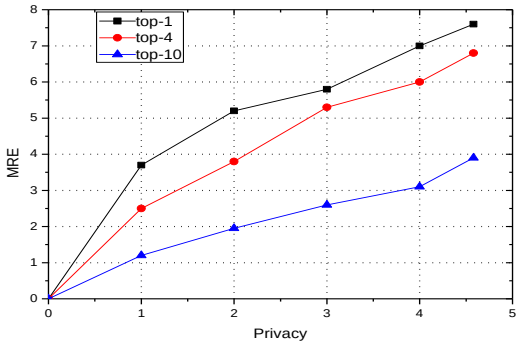


Figure 7. *MRE* without Item Conflicts

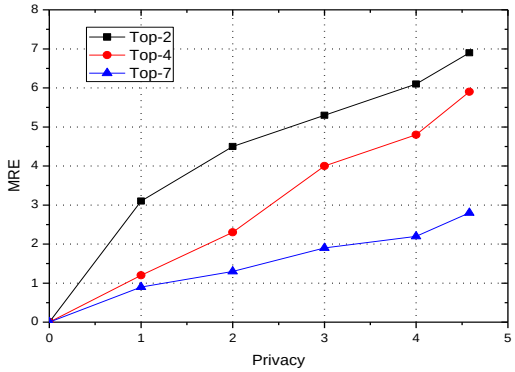


Figure 8. *MRE* for Recommending *k* Items with Conflicts Existing Only in Each Subset

When recommending items without conflicts, the server follows the algorithm in Figure 4, which output the optimal set of *k* items, from the point of view of the server, given a generalized context. However, this item set usually is not the true optimal set, assuming that the actual user context is given. We vary *k* to evaluate the *MRE* metric for recommending items without conflicts, with different privacy achieved. The average MRE over all the contexts is shown in Figure 7.

As shown in Figure 7, similar with the *MAE* metric, when a user gets more privacy protection, the *MRE* metric will increase. This is easy to understand, as more user privacy means a more generalized context that the server uses and more imprecision in user rating prediction. As a result, the difference between the expected revenue and the true revenue will also increase. In addition, Figure 7 also shows that, the more items recommended to a user, the less *MRE* will be incurred. The reason is that, *MRE* is defined as the average difference between two item sets. As the item set increases, the difference decreases.

When conflicts may exist between items, we only consider a special case: all the items are divided into several subsets, and conflicts only exist between items from the same subset. The server follows algorithm in Figure 5 to recommend items to a user. We also measure *MRE* for this special case. We artificially divide the items in varying number of subsets (*i.e.*, vary k for the top-*k* strategy), and the result is shown in Figure8.

Again, more privacy will result in more *MRE*, *i.e.*, more imprecise recommendations from the point of view of the server. Interestingly, in this case, *MRE* is less than that in recommendation without item conflicts for the same k. In addition, the larger k, the less *MRE* is. The reason is probably that, the algorithm in Figure 5 selects one single item from each subset. When the items are divided into more subsets, each subset contains fewer items, and the prediction become more easy and precise.

6. Conclusion

In this paper, we studied the privacy-preservation problem in context-aware recommender system. We proposed a generalization-based scheme to conceal the user's real context. Specifically, this scheme employs a context generalization based approach, in which the real context is generalized to a more general one. We presented the context generalization algorithm for users and the item recommendation algorithm for the server. In this scheme, there is a tradeoff between recommendation accuracy and privacy. The experimental evaluation showed that, the scheme can protect user context privacy, without much service quality reduction.

In the future, we will evaluate our schemes in a real world dataset, once available. Especially, we want to evaluate the efficiency of the scheme in a large dataset with different rating sparsity, which determines the applicability of the schemes in real-world applications.

Acknowledgments

The work has been supported by National Science Foundation of China (Grant No. 61373016) and A Project Funded by the Priority Academic Program Development of Jiangsu Higer Education Institutions (PAPD).

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of- the-art and possible extensions", IEEE Trans Knowl Data Eng, vol. 17, **(2005)**, pp. 734–749.
[2] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems", In: Ricci F, Rokach L, Shapira B, Kantor P (eds) Recommender systems handbook, Springer, Berlin, **(2011)**, pp. 217–256.
[3] L. Baltrunas and F. Ricci, "Context-based splitting of item ratings in collaborative filtering", In proc. ACM RecSys, **(2009)**, pp. 245–248.
[4] C. Palmisano, A. Tuzhilin and M. Gorgoglione, "Using context to improve predictive modeling of customers in personalization applications", IEEE Trans Knowl Data Eng., vol. 20, no. 11, **(2008)**, pp. 1535–1549.
[5] P. Jagtap, A. Joshi, T. Finin and L. Zavala, "Preserving Privacy in Context-Aware Systems", In Proc. ICSC, **(2011)**, pp. 149-153.
[6] J. Tsai, P. G. Kelley, L. F. Cranor and N. Sadeh. "Location sharing technologies: Privacy risks and controls", I/S: A Journal of Law and Policy for the Information Societ, vol. 6, no. 2, **(2010)**.
[7] P. Alves and P. Alves, "Privacy in Distributed Context-Aware Systems", Technical Report RT/23/2011, INESC-ID, **(2011)**.
[8] M. Wernke, P. Skvortsov, F. Durr and K. Rothermel, "A classification of location privacy attacks and approaches", Pers Ubiquit Comput, DOI 10.1007/s00779-012-0633-z.
[9] J. Zhan, C.-L. Hsieh, I-C. Wang, T.-S. Hsu, C.-J. Liau and D.-W. Wang, "Privacy-Preserving Collaborative Recommender Systems", IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews, vol. 40, no. 4, **(2010)** July.
[10] A. C. Yao, "Protocols for secure computations", In Proc. FOCS, **(1982)**.
[11] O. Goldreich, S. Micali and A. Wigderson, "How to play any mental game", In proceedings of the 19th annual ACM Symposium on Theory of Computation, **(1987)** May, pp. 218-229.
[12] W. Du and M. J. Atallah, "Privacy-Preserving Cooperative Scientific Computations", In 14th IEEE Computer Security Foundations Workshop, Nova Scotia, Canada, **(2011)** June 11-13, pp. 273-282.
[13] W. Du and M. J. Atallah, "Protocols for Secure Remote Database Access with Approximate Matching", in Proc. ACM CCS, the first workshop on security and privacy in e-commerce, Athens, Greece, **(2000)** November 1-4.
[14] J. Biskup and U. Flegel, "On Pseudonymization Of Audit Data For Intrusion Detection", In Workshop on Design Issues in Anonymity and observability, **(2000)** July, pp. 161-180.
[15] M. J. Atallah and W. Du, "Secure Multiparty Computational Geometry", in Proc. WADS, **(2001)** August 8-10, pp. 165-179.
[16] T. Jung, X.-Y. Li and S.-J. Tang, "Privacy-Preserving Data Aggregation without Secure Channel: Multivariate Polynomial Evaluation", in Proc. Infocom., **(2012)**.
[17] L. Baltrunas, B. Ludwig, S. Peer and F. Ricci, "Context relevance assessment and exploitation in mobile recommender systems", Pers Ubiquit Comput, vol. 16, **(2012)**, pp. 507–526.
[18] M. Naor and B. Pinkasy, "Oblivious Transfer and Polynomial Evaluation (Extended Abstract)", in Proc. ACM STOC, **(1999)**, pp. 245-254.

[19] S. Berkovsky, T. Kuflik and F. Ricci, “The impact of data obfuscation on the accuracy of collaborative filtering", Expert Systems with Applications, vol. 39, **(2012)**, pp. 5033–5042.
[20] E. A¨ımeur, G. Brassard, J. M. Fernandez and F. S. M. Onana. Alambic, “a privacy-preserving recommender system for electronic commerce", Int. J. Information Security, vol. 7, no. 5, **(2008)**, pp. 307–334.
[21] J. F. Canny. “Collaborative filtering with privacy", In IEEE Symposium on Security and Privacy, **(2002)**, pp. 45–57.
[22] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl, “Evaluating collaborative filtering recommender systems", ACM Transactions on Information Systems, vol. 22, no. 1, **(2004)**, pp. 5–53.
[23] C. Desrosiers and G. Karypis, “A comprehensive survey of neighborhood-based recommendation methods”, Recommender systems handbook, Springer US, **(2011)**, pp. 107-144.
[24] R. Shokri, *et al.*, "Protecting location privacy: optimal strategy against localization attacks", Proceedings of the ACM conference on Computer and communications security, **(2012)**.
[25] A. Narayanan, *et al.*, "Location Privacy via Private Proximity Testing", In Proc. NDSS, **(2011)**.
[26] C. A. Ardagna, *et al.*, "An obfuscation-based approach for protecting location privacy”, Dependable and Secure Computing, IEEE Transactions on, vol. 8, no. 1, **(2011)**, pp. 13-27.
[27] U. Panniello and M. Gorgoglione, "Incorporating context into recommender systems: an empirical comparison of context-based approaches", Electronic Commerce Research, vol. 12, no. 1, **(2012)**, pp. 1-30.
[28] O. Kwon and J. Kim, "Concept lattices for visualizing and generating user profiles for context-aware service recommendations", Expert Systems with Applications, vol. 36, no. 2, **(2009)**, pp. 1893-1902.

