Predictive Anonymization: Utility-Preserving Publishing of Sparse Recommendation Data

Chih-Cheng Chang
Department of Computer Science, Rutgers University
Piscataway, NJ, USA
geniusc@cs.rutgers.edu

Brian Thompson
Department of Computer Science, Rutgers University
Piscataway, NJ, USA
bthom@cs.rutgers.edu

Hui (Wendy) Wang
Department of Computer Science, Stevens Institute of Technology
Hoboken, NJ, USA
hwang@cs.stevens.edu

Danfeng Yao
Department of Computer Science, Rutgers University
Piscataway, NJ, USA
danfeng@cs.rutgers.edu

ABSTRACT

Recently, recommender systems have been introduced to predict user preferences for products or services. In order to seek better prediction techniques, data owners of recommender systems such as Netflix sometimes make their customers' reviews available to the public, which raises serious privacy concerns. With only a small amount of knowledge about individuals and their ratings to some items in a recommender system, an adversary may easily identify the users and breach their privacy. Unfortunately, most of the existing privacy models (e.g., k-anonymity) cannot be directly applied to recommender systems.

In this paper, we study the problem of privacy-preserving publishing of recommendation datasets. We represent recommendation data as a bipartite graph, and define several attacks on the graph that can re-identify users and determine their rated items and ratings. To deal with these attacks, we give formal privacy definitions in recommender systems. We develop a robust and efficient anonymization algorithm, Predictive Anonymization, to achieve the privacy goals. Our experimental results show that Predictive Anonymization can prevent the attacks with very little impact to prediction accuracy.

1. INTRODUCTION

To help consumers make intelligent buying decisions, many websites provide *recommender systems* [28] that give users a list of items that are potentially interesting to them. It also predicts user preferences for products or services by learning from past user-item relationships. Recent years have witnessed the rapid increase in online recommender systems. The recommender systems collect users' inputs (e.g., reviews, ratings, etc.), compare the collected data to others, and calculate a list of recommended items for the user. It has been proven to be effective at delivering the user more intelligent and proactive recommendations [29].

To support advanced data mining and prediction algorithms, data owners sometimes publicly release their recommendation datasets. The released datasets may include information that is legally protected, or otherwise private or sensitive data, such as buying records and movie-viewing histories. For example, Netflix, the world's largest online DVD rental service, recently announced a one million dollars *Netflix Prize* for improving their movie recommendation algorithm. To aid contestants, Netflix released a Netflix Prize dataset containing around 100 million movie ratings, created by around 500 thousands Netflix subscribers between December 1999 and December 2005. As the dataset contains users' private preferences to the movies, Netflix removes customers' names to protect their privacy. However, this naively anonymized data suffers from re-identification attacks as recently demonstrated [23].

1.1 Motivation Examples

With some additional knowledge on a user's review history, the adversary may be able to uniquely identify the user and consequently learn additional information about the user. For example, the adversary knows her co-worker Alice watched the movie *Pretty in Pink*, a movie from the eighties which has not been reviewed by many people in recent years. By matching with the released dataset in Figure 1 (a), the adversary can identify that ID 0004 corresponds to Alice. Consequently the adversary learns all movies that Alice has reviewed and her preferences to these movies (e.g., she likes the movie *Star Wars*). This example shows that removing user names is not sufficient to protect users' privacy; with certain auxiliary knowledge of users' reviews, the adversary is still able to intrude privacy. Such background knowledge can be easily obtained from personal blogs, [1] public bulletin board systems (BBS), and other related recom-

[1] An example of an individual's Netflix review on AOL blog:

VLDB '09, August 24-28, 2009, Lyon, France

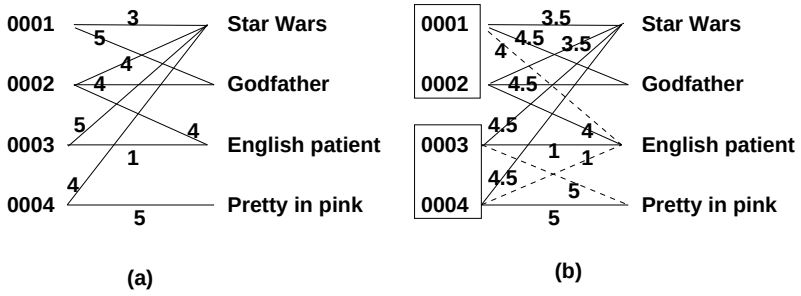

Figure 1: An example: (a) The original graph with user names replaced with IDs, (b) The k-anonymized graph where $k = 2$. The rectangle boxes represent anonymization groups. Dotted lines represent fake edges added for anonymization purpose. New ratings are computed as the average of non-zero ratings in the anonymization group.

mender systems (e.g., the IMDB website). Narayanan et al. showed that only a little bit of knowledge about an individual subscriber can easily identify his/her record if it is present in the dataset [23]. For instance, 84% of subscribers can be uniquely identified if the adversary knows 6 out of 8 movies outside the top 500 popular movies. The *sparsity* of the Netflix Prize data [23] also contributes to the ease of attacks, i.e., there are a very small number of non-null ratings in the dataset. Sparsity is a common property for recommendation data. The intuition is that unpopular movies are rarely rated by users; thus, by rating unpopular movies the user distinguishes herself from the crowd.

We believe that releasing anonymized data to the public for research is an unavoidable trend and has the potential to provide society with substantial benefits in many fields, including healthcare, medical sciences, and social sciences. Conceivably, there is a tradeoff between the utilities and privacy of anonymized data. For example, in recommender systems, as the anonymized data deviates from its original in attempts to preserve privacy, the predictions based on the anonymized data may become less accurate. Sparsity in recommendation data significantly increases the difficulty of such anonymization tasks in real-world datasets, which is explained/addressed later. In this paper, we take on the task of developing a utility-preserving and efficient approach for anonymizing large-scale real-world datasets.

1.2 Challenges

Although privacy preservation in data publishing has been studied extensively with several privacy models (e.g., k-anonymity [32] and l-diversity [21]) and proposed algorithms, most of them can only deal with relational datasets. There have also been several studies on privacy-preserving publishing of graphs (e.g., [36], [11], [16], [34]). Unlike our work, their graphs are not labeled, while we consider labeled bipartite review graphs in recommender systems, where the labels can be used as part of privacy attacks.

Most importantly, none of existing anonymization work (on relational databases or graphs) have studied the effect that sparsity has on privacy. Real-world recommender system datasets are quite sparse, that is, each individual rating record contains values only for a small fraction of attributes (i.e., the items) [23]. This problem, commonly referred to as *the sparsity problem*, has a major negative impact on anonymization: it increases the probability that de-anonymization succeeds, and increases the difficulty of designing anonymization schemes that provide acceptable predication accuracy. Unfortunately, the existing anonymization algorithms are not effective when applied to the sparse datasets. In comparison, we develop a general and efficient approach, called *predictive anonymization*, that preserves both user privacy and data utility.

In addition, most of the existing privacy methods are built around the assumption that there are two *non-overlapping* value sets: sensitive values, which need to be kept private, and quasi-identifier values, which can be used by the adversary to identify individuals. In recommender systems, these two sets are not disjoint; *all information in a recommendation dataset is sensitive, and can also potentially be used as quasi-identifiers.* Due to this additional challenge, the existing privacy models cannot be directly applied to recommender systems.

An elegant correlation-aware anonymization approach was proposed to handle sparse high-dimensional graph with no edge labels by capturing underlying data correlation [12]. In comparison, Netflix data contains non-binary edge labels, which demand different privacy and attack models and new anonymization approaches.

1.3 Contributions

In this paper, we study how to publish sparse recommender data with a sufficient amount of privacy and utility. Our anonymization goal is to group and average *similar* user profiles together. However, in a sparse dataset, finding similar users is extremely challenging and inaccurate because of insufficient evidence to work with. That is, a user only has a small number of non-null ratings, which render even powerful similarity measures ineffective. *Our idea is that before anonymization, we pad the null entries to reduce data sparsity by performing a round of prediction.* This type of predict-then-anonymize sequence is able to uncover and leverage the latent interests of users that would otherwise be lost without the pre-processing. We call our approach *predictive anonymization*, which is a powerful and general approach for publishing all types of recommender data. We summarize our contributions as follows.

- We formalize the privacy and attack models in recommender data. We model the review data as a labeled bipartite graph, where nodes are divided into two dis-

http://journals.aol.com/slowmotionlife/slowmotionlife/entries/2005/01/08/december-netflix-reviews/619

joint sets, one for users and one for items. We formally define two types of adversary attacks, namely *structure-based* attack and *label-based* attack. We give the *k-anonymity* and *l-diversity* privacy definitions for the bipartite graphs.

- To combat sparsity and preserve data utility, we develop a novel *predictive anonymization* technique to pad, cluster, and anonymize the recommendation data. As mentioned earlier, the uniqueness of our approach lies in the padding phase where we aim to discover the underlying and latent information of the original dataset in order to group similar users together. As a desirable side effect, the padding step adds fake edges, which are noise data that ultimately help us achieve our privacy goal. For anonymization, k users of similar preferences are grouped together. We achieve k-anonymity by adding fake edges so that every user will review the same set of items as the other $k-1$ users in his/her group. We analyze the privacy guarantee of the *predictive anonymization* approach. We also discuss how to extend our approach to a more strict privacy model, *l-diversity*.

- We carry out a non-trivial set of experiments to test the effectiveness and efficiency of our anonymization method with a large dataset. We experiment with the entire Netflix Prize dataset in our evaluation, which contains 480,189 users and 17,770 movies, i.e., we perform the padding, clustering, and homogenization for all users and all movies in the Netflix Prize data. We experiment with different sizes of anonymous groups and data publishing mechanisms. We also characterize the data sparsity and various user-similarity metrics.

  Our experiment results show that (1) naive anonymization methods incur high information loss and destroy data patterns even with a small k value, and (2) our predictive anonymization approach is extremely effective in reducing data sparsity while preserving data utility during anonymization. Our experiment results highlight promising new directions for the database, data mining, and security communities in pursuing data anonymization for publishing large-scale sensitive data.

2. RECOMMENDER SYSTEMS

Recommender systems produce automatic predictions about the interests of users by collecting preference information from many users. A recommender system consists of:

- A set of users $U = \{u_1, \ldots, u_m\}$
- A set of items $O = \{o_1, \ldots, o_n\}$
- An ordered set of possible rating values S
- A set of user ratings $\{(u, o, r)\}$ where $u \in U$, $o \in O$, and $r \in S$ is the rating value assigned by the user u to an item o (only if u has rated o).

Given a recommender system, the ratings can be represented as an $m \times n$ matrix R. Each cell $r_{i,j}$ is either a real number r, which corresponds to the triplet (u_i, o_j, r), or 0 if user u_i has not rated item o_j. This leads to a natural representation of the system as a bipartite graph. We refer to the vertices that represent users, denoted by V_U, as *user nodes*. Vertices representing the items, denoted by V_O, are called *item nodes*.

DEFINITION 2.1. **[Bipartite Review Graph]** A recommender system (U, O, R) corresponds to a *bipartite review graph* $G = (V_U \cup V_O, E, L)$, where each user $u_i \in U$ corresponds to a node $v_{u_i} \in V_U$, each item $o_j \in O$ corresponds to a node $v_{o_j} \in V_O$, and each non-zero entry $r_{i,j}$ in the rating matrix corresponds to the edge $(v_{u_i}, v_{o_j}) \in E$. $L : E \to S$ is the *label function*, which assigns to each edge $(v_{u_i}, v_{o_j}) \in E$ the label $r_{i,j} \in S$, one of the possible rating values. Thus the rating (u_i, o_j, r) corresponds to the edge $e = (v_{u_i}, v_{o_j}) \in E$ being labeled with $L(e) = r$.

Furthermore, the *review graph of user* u is the subgraph $G_u = (\{v_u\} \cup N(v_u), E_u, L_u)$, where $N(v_u) \subseteq V_O$ is the neighborhood of v_u, i.e. $N(v_u) = \{v_o \mid (v_u, v_o) \in E\}$, and $E_u \subseteq E$ and $L_u \subseteq L$ are the corresponding edges and labels in the subgraph induced by $\{v_u\} \cup N(v_u)$.

An example of a bipartite review graph is shown in Figure 1 (a). In this graph, the review graph of user 0001 consists of his/her user node, two movie nodes, *Star wars* and *Godfather*, and two labeled edges connecting the user node with the two movie nodes. Note that bipartite graphs can be applied to not only review data but also other applications, for example, network tracing, web browsing and search history, etc.

The task of recommender systems is to predict the rating r associated with a user $u \in U$ and an item $o \in O$, i.e., the rating r that u would give to o. This predicted rating must be within the same range S as the other ratings.

Much research has been done to improve the quality of prediction. This task is called collaborative filtering. Most approaches to this task, described so far in the literature, are variations of k-nearest neighbors (e.g., [2], [13]) or singular value decomposition (SVD) (e.g., [14]). We refer the readers to [7] for a good survey of collaborative filtering algorithms.

3. PRIVACY MODEL

In this section, we define our privacy model.

3.1 Privacy Goals, Adversary Knowledge, and Attacks

In released recommendation data, e.g., Netflix, usernames are replaced with unique integer IDs. Item names and ratings are released publicly for data analysis purposes. In this paper, we formalize two privacy goals in anonymizing recommendation data.

- *Node identification privacy*: after being removed, the identification of individuals is considered sensitive.
- *Link existence privacy*: the knowledge of which items are reviewed by specific users is considered to be private information of the users.

We assume that from external data resources, the adversary knows a subset of items that a specific user has reviewed. Using this information, the adversary can try to uniquely identify a user by matching the background knowledge with the released dataset. Based on this, we define the *structure-based attack*.

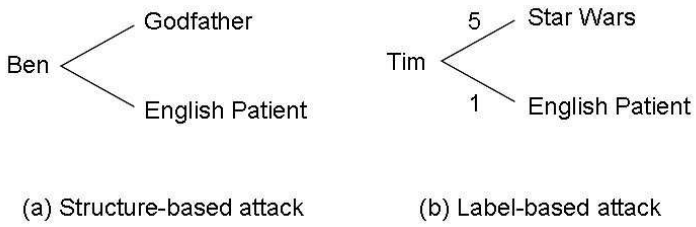

Figure 2: Adversary Graphs

DEFINITION 3.1. **[Structure-based Attack]** Given a released bipartite review graph $G^* = (V_U \cup V_O, E^*, L^*)$, let $G_u^A = (\{v_u\} \cup N^A(v_u), E_u^A, \emptyset)$ be the subgraph representing the adversary knowledge for a user u. If there are k user nodes, each of which $v_{u'} \in V_U$ has $G_u^A \subseteq G_{u'}^*$, then we say the user u is identified by the *structure-based attack* with probability $1/k$.

Figure 2 (a) shows an example of the adversary knowledge for a structure-based attack. The adversary knows that Ben has watched the movies *Godfather* and *English Patient*. By matching the background knowledge to the released data in Figure 1 (a), the attacker uniquely identifies Ben as user 0002.

In addition to the structure-based attack that is based on knowledge of which items have been reviewed by a user, the attacker may also know the ratings of these items. Such additional adversary knowledge enables the *label-based attack*.

DEFINITION 3.2. **[Label-based Attack]** Given a released bipartite review graph $G^* = (V_U \cup V_O, E^*, L^*)$, let $G_u^A = (\{v_u\} \cup N^A(v_u), E_u^A, L_u^A)$ be the subgraph representing the adversary knowledge for a user u. If there are k nodes, each of which $v_{u'} \in V_U$ has $G_u^A \subseteq G_{u'}^*$, then we say the user u is identified by the *label-based attack* with probability $1/k$.

Figure 2 (b) shows an example of adversary knowledge for the rating-based attack. The adversary knows that Tim has given a low rating to the movie *English Patient*. By matching this knowledge to the released data (Figure 1 (a)), the adversary uniquely identifies Tim as user 0003, since the other user who reviewed *English Patient* gave a high rating.

The label-based attack is a stronger model than the structure-based attack. Thus by giving privacy guarantees against the label-based attack, we are protecting against the structure-based attack as well. In the following, we mainly focus on the label-based attack.

3.2 Privacy Principle

In practice, it is hard to predict the amount of background knowledge that an adversary has gained. Therefore, we aim to provide protection against the strongest adversary that we have considered, the label-based attack. To achieve this goal, we adapt the definition of *k-anonymity* [31, 32] to our model. The conventional k-anonymity model defines quasi-identifier values (i.e., the non-ID values that can be used to identify individuals) and sensitive values. These two set of values normally do not overlap. In our problem, the nodes, edges, and labels in the released review graph are both quasi-identifiers and sensitive values. To address this problem, we define k-anonymity in recommender systems as follows:

DEFINITION 3.3. **[k-anonymity]** Given a bipartite review graph $G = (V_U \cup V_O, E, L)$, let $G^* = (V_U \cup V_O, E^*, L^*)$ be the review graph of the released dataset. We say G^* satisfies *k-anonymity* if for every user $u \in U$, there are at least $k-1$ other users $u_i \in U$, $i \in I$ such that G_u^* is isomorphic to $G_{u_i}^*$. We say that $\{u\} \cup \{u_i\}_{i \in I}$ is the *anonymization group* of u.

Intuitively, Definition 3.3 requires that for each user node u, there are at least k - *1* other user nodes that have identical review graphs to u in terms of both structure and labels. Thus the k-anonymity model is effective for defending against the label-based attack.

4. PREDICTIVE ANONYMIZATION METHOD

In general, data owners publish their recommendation data to seek improved recommendation algorithms. Thus it is desirable that the released data strives to preserve the utility of the data as much as possible. To achieve this goal, we design a utility-preserving anonymization method, *Predictive Anonymization*, to produce an anonymized dataset that not only satisfies k-anonymity but also yields a small amount of utility loss.

The key step in our method is a *predictive padding* procedure that aims at amplifying the original data features without introducing much noise. Note that one can easily pad the data with random or arbitrary values. However, if not done carefully, padding may destroy original data patterns and cause information loss. As it will soon become clear, our predictive padding approach can strategically replace null entries with meaningful values.

In this section, we describe the details of our predictive anonymization procedure, which is to be performed by the data owner before publishing sensitive recommender data. Our anonymization procedure consists of three major steps: (1) strategically pad the data to reduce sparsity, (2) construct the anonymization groups from the preprocessed data, and (3) apply homogenization operations on each group.

4.1 Step 1: Predictive Padding

Recommender systems are typically sparse [29, 15, 30], that is, the number of items reviewed by an average user is small compared to the entire set of items. Sparsity increases the probability that re-identification succeeds and decreases the amount of auxiliary information needed for re-identification [23]. Most importantly, it hampers the effectiveness of anonymization that is based on grouping users with similar ratings – due to data sparsity, the pair-wise user similarities are low in general as two users may have very few overlaps in what they recommend. Thus, anonymization on the *original* data may not effectively group similar users, which impairs the accuracy of prediction.

We take a novel approach to anonymization by utilizing the SVD technique as a pre-processing method before performing anonymization. Singular Value Decomposition (SVD) is a well-known matrix factorization technique. Regularized SVD, a technique inspired by effective methods from the domain of natural language processing, was proposed for collaborative filtering by Simon Funk (Brandyn Webb) [35]. In particular, in regularized SVD, the original review matrix is decomposed into two matrices, the movie

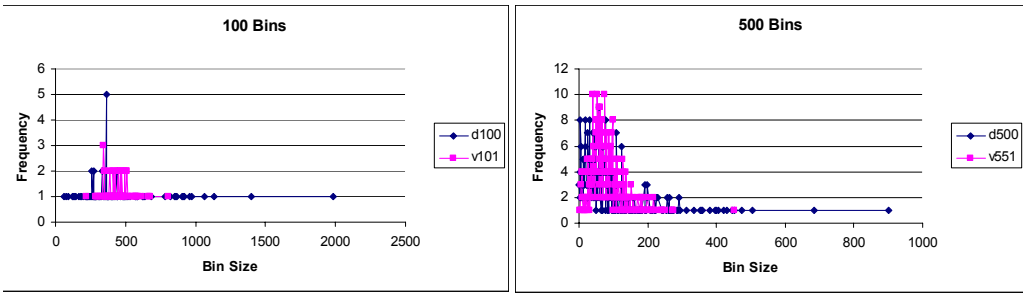

Figure 3: Real User vs. Virtual Center Points

aspect matrix and the user preference matrix. The prediction of user i's rating for movie j is made in the following way:

$$\bar{r_{i,j}} = \vec{u_i}^T \vec{o_j}$$

where $\vec{u_i}$ and $\vec{o_j}$ are the user and movie feature vectors for user i and item j, respectively. The purpose of using SVD is to find those two matrices which minimize the resulting approximation error.

We first perform a round of SVD prediction to obtain padded data, which effectively eliminates the sparsity problem. (Note that our predictive padding is a general approach that supports any prediction techniques beyond SVD.) This predictive padding does not affect the original data, which may still be used to construct the final released anonymized dataset. Yet, the padded data significantly helps us in improving clustering accuracy, which is explained next.

4.2 Step 2: Clustering and Construction of Anonymization Groups

Given the recommender dataset, the key to anonymization is to determine a partition of its users, so that each user u is included in a single partition which includes at least $k-1$ other users. We call the partition the *anonymization group* of the user u. Note that the anonymization groups are disjoint, i.e., each user belongs to only one group. We delineate several goals for the partitioning step: (1) all partitions must be of at least size k, (2) the partitioning technique must be able to deal with large datasets, e.g., the Netflix dataset, and (3) the partition must respect the utility goal of prediction accuracy. In the following, we explain how our anonymization group construction procedure addresses these issues.

The essence of our group construction is clustering. The main differences between our following method and existing clustering-based anonymization work (e.g., [1] are (1) we compute and utilize pair-wise similarity values for clustering weighted bipartite graphs (2) our method based on sampling and bin-assigning is scalable to large datasets. At the end of the procedure, each cluster has at least k users who have similar movie preferences. The clustering procedure works as follows:

Step 2.1: Sampling To deal with large recommendation datasets, instead of clustering the whole dataset at once, which is computationally demanding, we randomly pick a sample out of the preprocessed dataset. By choosing a large enough random sample, we get a sample that closely represents the trends of the original dataset with high probability.

Step 2.2: Cluster on Sample We apply clustering on the sample to find t center points. The task of clustering data has been studied in great depth. One classic clustering technique is known as the k-means algorithm [20, 22]. To avoid confusion with the k in k-anonymity, we use the term t-means throughout the paper. The t-means algorithm takes an input parameter, t, and partitions a set of n objects into t clusters so that the resulting intra-cluster similarity is high.

In practice, however, it has been observed that classic clustering algorithms frequently produce clusters whose sizes are of skewed distribution (i.e., some clusters are huge while some are small or even empty), especially when clustering high-dimensional datasets [6]. In contrast, a balanced distribution of cluster sizes is desired to ease computation for subsequent steps. To achieve this, we use the *bounded t-means* algorithm from [34], which guarantees that the sizes of all t clusters are no smaller than a pre-defined lower bound. By choosing an appropriate value of t, we can ensure that there will not exist unwieldily large clusters.

We apply the bounded t-means algorithm to the sample to cluster the users into t groups, based on their similarities. To reduce the complexity of the entire clustering procedure, we choose the value $t = \sqrt{n}$. More details of why we pick this value are explained in Section 6.1.

Similarity Metric To perform bounded t-means clustering, we must first define our user similarity metric. Let $MaxDiff$ be the difference of the highest and the lowest possible ratings in the dataset, i.e. $MaxDiff = max(S) - min(S)$. Then given two users u_1 and u_2 and their corresponding rating vectors $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$, we define $d_i = |x_i - y_i|/MaxDiff$ for $1 \leq i \leq n$. The similarity between u_1 and u_2 is then defined to be

$$sim(u_1, u_2) = 1 - \frac{\sum_{1 \leq i \leq n}(1-d_i)^2}{n}$$

It is straightforward to verify that $sim(u_1, u_2) = 0$ only when u_1 and u_2 match exactly on every common entry, and $sim(u_1, u_2) = 1$ only when u_1 and u_2 have opposite minimum and maximum ratings for each item. This method essentially gives a squared penalty for large differences in movie preferences between two users. We studied the effectiveness of this metric against other similarity metrics. More details can be found in Section 7.4.

After the users in the sample are clustered into groups by the bounded t-means algorithm, we compute the center points of the clusters. The bounded t-means algorithm in [34] uses *virtual centers*. We adapt this method to our problem. In particular, consider a cluster C of users $\{u_1, \ldots, u_k\}$, each u_i assigned with a rating vector $\langle r_{i,1}, \ldots, r_{i,n} \rangle$ to the items $o_1, \ldots, o_n$. Then the virtual center c is a vector of

ratings $\langle \hat{r_1}, \ldots, \hat{r_n} \rangle$ such that

$$\hat{r_j} = \sum_{i=1}^{k} r^{-}_{i,j}/n.$$

Note that here we are using the padded user rating vectors, which have no empty ratings.

Our experimental results demonstrate the effectiveness of using virtual centers as opposed to real users as center points; the sizes of clusters with virtual centers are of less skewed distribution. For example, Figure 3 (a) shows that the sizes of the clusters with real user centers vary from 50 to 2000, while the sizes of the clusters with virtual centers vary from around 200 to 900. Since clusters of balanced sizes facilitate easy data handling in later anonymization steps, we are encouraged to return t virtual centers as the result of Step 2.2.

Step 2.3: Construction of Anonymization Groups Let $\{a_1, \ldots, a_t\}$ be the t cluster centers resulting from Step 2.2. We now *partition all users in the dataset into t bins*, where bin B_i contains all the users whose closest center point is a_i. Finally, we partition the users in each bin into anonymization groups of size k. In particular, for each bin B_i, we apply the bounded t-means algorithm with $t = |B_i|/k$. Thus all clusters returned by Step 2.3 are guaranteed to contain at least k users. By grouping similar users into the same bin, we incur little information loss by first using our sampling procedure, compared to if we had clustered all the users at once. However, our procedure yields a significant speed-up in run-time for the clustering step, especially when performed on large datasets. See Section 6.1 for details.

4.3 Step 3: Homogenization

To defend against both the structure-based attack and label-based attack, our final step is to homogenize the k users in each anonymization group so that they have identical review graphs, including the ratings in the graph. A straightforward way to do this is to apply the popularly used generalization and suppression techniques [31, 32]. However, as pointed out by [23], generalization and suppression may completely destroy the utility of the data for collaborative filtering. Thus we take a different approach, *homogenization*, which consists of adding fake edges and labels so that all k users in the same anonymization group are connected to the same set of item nodes with the same ratings.

Formally, the homogenization of the anonymization group corresponding to cluster C of users $\{u_1, \ldots, u_k\}$ has the following operations: *union*, *complete*, and *average*. First, we construct the *union of the review graphs* of all users in C. Let the union result be $G_C = (C \cup N(C), E_C, L_C)$. Second, we add fake edges between users and items to create G^*_C, a *complete bipartite subgraph*; i.e., $E^*_C = C \times N(C)$. For instance, in Figure 1 (b), user 0001 and 0002 are in the same anonymization group. A fake edge is added between user 0001 and movie *English Patient*, so that both user 0001 and 0002 review the same set of movies in the review graph. Third, we re-label all the edges in G^*_C, including the fake ones, with the appropriate average rating:

$$(\forall u_i \in C, o_j \in N(C)) \;\; L^*_C(u_i, o_j) = \hat{r_j} = \sum_{u_i \in C, r_{i,j} \neq 0} r_{i,j}/k',$$

where k' is the number of users in C who have rated item o_j. Note that all users in C are assigned the same homogenized rating $\hat{r_j}$ for each movie o_j.

5. EXTENSIONS

5.1 Achieving l-diversity

The l-diversity model provides complementary privacy protection to k-anonymity. In relational data, l-diversity requires that sensitive attributes should have *diversity* by having at least l distinct values in each k-anonymous class [21]. However, the definition and security implications of l-diversity in recommender databases are unclear. Therefore, we give the first formal definition of l-diversity for labeled bipartite review graphs, and an algorithm to realize both k-anonymity and l-diversity in recommender systems.

To appreciate the need for l-diversity in recommendation data, we need to first understand a subtle attack against link privacy. Once the anonymization group of a target victim is identified via a structure-based or label-based attack, it becomes easier to target that user for more sophisticated attacks. Although the adversary cannot explicitly identify any user, more can be deduced from the anonymized data than what we want to allow. For example, suppose an adversary only has the background knowledge to perform a structure-based attack. After identifying the correct anonymization group, since the k users in that group have identical review profiles, the adversary can easily learn new information about the ratings that the target user gave to those items.

This problem is further exascerbated by the fact that some items are rarely reviewed. Exploring rare items to re-identify users in Netflix data was recently studied [23], and facilitates the easy identification of a user's anonymization group, leaving the target susceptible to the above attacks. We refer to these attacks as *homogeneity attacks* following [21]. The threat of these homogeneity attacks motivates the need for l-diversity.

DEFINITION 5.1. [**Homogeneity Attack in Bipartite Graphs**] Given a released bipartite review graph $G^* = (V_U \cup V_O, E^*, L^*)$, let $G^A_u = (\{v_u\} \cup N^A(v_u), E^A_u, L^A_u)$ be the subgraph representing the adversary knowledge for a user u. Let $\{v_{u'}\}$ denote the set of nodes, including v_u, each of which $v_{u'} \in V_U$ has $G^A_u \subseteq G^*_{u'}$. If all the nodes in $\{u'\}$ are *identical*, then we say the *review profile* of user u is uniquely identified by the homogeneity attack.

Unlike relational data, our k-anonymization algorithm alone provides some degree of privacy even in the face of a homogeneity attack (See Theorem 6.3). Because we add fake edges during anonymization to average the k users, a rating in the anonymized data does not necessarily mean that the target user has rated that item, or if so, that the anonymized rating accurately reflects the true rating of that user. However, as explained above, homogeneity attacks can be effective in some scenarios.

First, we formally define $(1, l)$-diversity. We present an extension to the *Predictive Anonymization* algorithm that realizes both k-anonymity and $(1, l)$-diversity. In $(1, l)$-diversity, the number 1 represents that the adversary's prior knowledge is *at most one item* that is reviewed by the user. Intuitively, $(1, l)$-diversity requires that every single item must be included in at least l different anonymization groups.

DEFINITION 5.2. [$(1,l)$-**diversity**] Given a bipartite review graph $G = (V_U \cup V_O, E, L)$, let $G^* = (V_U \cup V_O, E^*, L^*)$ be the corresponding k-anonymized review graph with anonymization groups $\dot{\bigcup} C_i = U$. We say G^* satisfies $(1,l)$-*diversity* if for every item $o \in O$, there are at least l distinct anonymization groups C_i such that $v_o \in N(C_i)$.

$(1,l)$-diversity can be generalized to (b,l)-diversity in order to allow a stronger adversarial model, where the adversary's has background knowledge of at most b items that a user has reviewed.

DEFINITION 5.3. [(b,l)-**diversity**] Given a bipartite review graph $G = (V_U \cup V_O, E, L)$, let $G^* = (V_U \cup V_O, E^*, L^*)$ be the corresponding k-anonymized review graph with anonymization groups $\dot{\bigcup} C_i = U$. We say G^* satisfies (b,l)-*diversity* if for every set of b items $B = \{o_1, \ldots, o_b\} \subset O$ that have been rated by a user, there are at least l distinct anonymization groups C_i such that $B \subset N(C_i)$.

To achieve $(1,l)$-diversity, we modify our algorithm as follows: After Step 2 in Section 4, i.e., the anonymization groups have been constructed, for each item o, we check whether it has been covered by l groups. If it is not, we pick anonymization groups C such that $o \notin C$, and add fake edges between item o and all user nodes in C, so that every item o is connected to at least l groups. The labels on these fake edges are computed using the padded values for the corresponding users. The above method can be easily generalized to realize (b,l)-diversity, the details of which are omitted here.

5.2 Padded Anonymization

When calculating the average ratings $\hat{r}_o$ in the homogenization step (see Section 4.3), there are two choices for $r_{i,j}$: we can use either the ratings in the original dataset or the ones in the padded dataset. We observe that using original ratings may bring privacy leakage, which gave us motivation for looking to the l-diversity model; on the other hand, if we average on padded data that contains both the real and predicted ratings, actual ratings become more obscured, providing stronger privacy protection against homogeneity attacks. We call this variant of the predictive anonymization algorithm *Padded Anonymization*.

By homogenizing over the padded data, sparsity will be completely eliminated from the released data. Although it may initially seem that the released data does not preserve the characteristics of the original dataset, and thus would reduce the utility of the released dataset, we argue that such variation is necessary for the purpose of privacy protection. As shown by [23], sparsity increases the likelihood that re-identification succeeds and decreases the amount of auxiliary information needed for re-identification. Thus reducing sparsity will strengthen the robustness of the released dataset against attack. Furthermore, as it will be shown in Section 7, homogenizing over padded data does not decrease the utility of the released dataset. On the contrary, *our padded anonymization algorithm produces very accurate prediction results even with a large k value.* As a result, our predictive anonymization approach is effective for preserving both privacy and utility.

6. ANALYSIS

6.1 Complexity Analysis

The pseudo code of the our *Predictive Anonymization* algorithm is given in Algorithm 1. Let m be the number of users, and let n be the number of items ($m = 480,189$ and $n = 17,700$ in the Netflix dataset). Using recent techniques for optimization of SVD, Line 1 can be performed in $O(mn)$ time. Line 2 takes $O(s)$ time, where s is the size of the sample. Line 3 has complexity $O(st_1n)$. Line 4 runs in $O(mt_1n)$ time. For Line 5, clustering on each bin takes time $O(|B_i|tn) = O((|B_i| * |B_i|/k)n)$, where $|B_i|$ is linear in m/t_1. Thus the complexity is $O(m^2n/(t_1^2k))$. There are t_1 bins overall, so the total complexity is $O(m^2n/(t_1k))$. To reduce the quadratic complexity in m, we set $t_1 = \sqrt{m}$, which results in the complexity of this step being $O(m^{3/2}n/k)$. Note that the complexity of Line 4 then becomes $O(m^{3/2}n)$. For Line 6, homogenization of each cluster C takes complexity $O(|C|n) = O(kn)$. There are m/k clusters, thus the total complexity is $O(mn)$. Based on the above, the complexity of the entire algorithm is $O(mn + s\sqrt{m}n + m^{3/2}n + m^{3/2}n/k + mn)$. Since $s \ll m$, the complexity of Algorithm 1 is $O(m^{3/2}n)$.

Algorithm 1 Algorithm: Padded Predictive Anonymization

1: Pre-process the data by using SVD. All 0s in the matrix will be replaced with a predicted rating after SVD.
2: Pick a random sample of size s from the pre-processed dataset.
3: Apply the bounded t-means algorithm on the sample to find t_1 virtual center points $a_1, ..., a_{t_1}$, where $t_1 = \sqrt{m}$ (where m is the number of users in the original dataset).
4: Use the t_1 center points from the sample to partition the whole dataset into t_1 bins, with bin B_i containing all points in the dataset whose closest center point is a_i.
5: Cluster each bin B_i using the bounded t-means algorithm with $t = |B_i|/k$. Each cluster will correspond to an anonymization group. Thus the final set of anonymization groups is the union of all clusters from all bins.
6: In each cluster, homogenize all ratings to be the average of user ratings in the pre-processed dataset.

6.2 Privacy Analysis

In this section, we analyze the guarantee that our *Predictive Anonymization* algorithm can provide for both *node re-identification privacy* and *link existence privacy* defined in Section 3.1.

First, analogous to the correctness of the k-anonymity model on relational databases, we have:

THEOREM 6.1. **(Node Re-identification Privacy) :** Let G be the bipartite review graph for a recommender dataset, and let G^* be the corresponding released review graph. If G^* is k-anonymous, then any user vertex in G cannot be re-identified in G^* with confidence probability larger than $1/k$.

Second, as shown in Section 4.3, fake edges are added between user vertices and item vertices during the homogenization step, which prevents the adversary from explicitly determining which edges exist in the original dataset (or

furthermore their labels). We formalize this notion in the following theorems.

Assume that all (user, item) ratings are independent, both the existence and the values of the ratings. Furthermore, assume the adversary has no prior knowledge about the likelihoods that users have rated items. Then the confidence with which the adversary can learn the existence of a link is at most $1/k$. This claim is stated concisely as follows.

THEOREM 6.2. **(Link Existence Privacy) :** Thm 4.2: Assume all ratings are independent. Then an adversary with no prior knowledge employing a label-based attack can not predict the existence of an edge (v_u, v_o) with confidence greater than $\frac{1}{k}$.

Proof Suppose an adversary, using a label-based attack, is able to identify the anonymization group containing user u. Based on the existence of a link to item o in the released review graph, the adversary would like to infer whether user u gave a rating for o in the original dataset. However, the existence of the link in the anonymized graph only implies that at least one user in that anonymization group had rated o. With no prior knowledge, the adversary can only infer that user u had rated o with probability at least $\frac{1}{k}$.

Now suppose an adversary has prior knowledge that the probability a user has rated item o is p. This is often feasible in practice by learning aggregate information about the database. For example, in the Internet Movie Database (IMDB), [2] the most frequently rated movie is "The Shawshank Redemption," which has been rated by 2.4% of registered users. Assume that $p \leq 1/k$ (a reasonable assumption due to the sparsity of recommender datasets). We claim that even with this additional prior knowledge, the adversary can not significantly improve his confidence that a user has rated item o.

THEOREM 6.3. **(Link Existence Privacy With Prior Knowledge) :** Thm 4.3: Assume all ratings are independent. Then an adversary with prior knowledge $p \leq \frac{1}{k}$ for item o employing a label-based attack can not predict the existence of an edge (v_u, v_o) with confidence greater than $\frac{1}{k} + \frac{p}{2-kp}$.

Proof See Appendix.

Note that the greatest confidence gain occurs when $p = 1/k$, at which point the adversary has a $2/k$ confidence probability. Therefore, assuming that $p \leq 1/k$, the maximum confidence in predicting the existence of a link is bounded by $2/k$. Furthermore, recommendation data is typically very sparse, so it is of note that the adversary confidence tends to $1/k$ as $p \to 0$.

Note: While it may be difficult to learn the existence of a link, learning the non-existence of links is easy with a label-based attack. However, we claim that due to the sparsity of the data and the practical significance of a link, it is reasonable to assume that only positive link existence should be considered sensitive.

7. EXPERIMENTS

We have done a set of experiments to evaluate both the effectiveness and efficiency of our k-anonymization algorithm Algorithm 1 (without l-diversity). Specifically, we want to evaluate the impacts of padding and anonymization on the utility and structure of the anonymized data, and the amount of information change introduced by the anonymization. In this section, we describe our experiment design and results.

7.1 Setup

We ran our experiments parallelized on 6 different machines. Four of the machines are equipped with eight Intel(R) Xeon(R) CPU 3.00GHz, 16GB memory and CentOS 5.2 Linux, and the other two machines are equipped with two Intel(R) Core(TM)2 Duo CPU at 3.00GHz, 3GB memory and Fedora 8 Linux. We implemented our algorithm in C++, Java, and Perl.

We use the entire Netflix dataset for our experiment. The original data contains a total of 480,189 users' ratings on 17,770 movies. The ratings range from 1 to 5, with 0 meaning a rating does not exist. The Netflix challenge set (a.k.a. probe set) is used to evaluate the performance of a prediction algorithm. It contains 459,178 users and 1,425,333 user-movie pairs to be predicted. The users are a subset of the original Netflix dataset. In our experiments, we use the challenge set to evaluate the utility and information loss in the anonymized data, by comparing to a fixed prediction algorithm, namely SVD. (We do not aim to develop a new collaborative filtering mechanism.) Briefly, the challenge set is used as follows. For each user-movie pair in the challenge set, one needs to predict the corresponding ratings based on the dataset $\hat{D}$, where $\hat{D}$ is the Netflix Prize set excluding the challenged entries.

We use the open-source SVD implementation in the Netflix Recommender Framework [37] for padding, and also for prediction in some experiments, which is described in more detail later. After running SVD padding, the size of the (padded) dataset is about 36GB. All data is written into hard disk in binary format, and accessed using the *mmap* system call. Due to the file size limit for mmap and in Linux, we split the padded dataset into 40 binary files.

After prediction (using SVD or user-based filtering), we measure the *root mean squared error* (RMSE) of the prediction result on the Netflix challenge dataset. Predicted ratings are between one and five. The actual user ratings can be found in the original Netflix dataset. In order to clearly quantify the amount of information loss caused by the anonymization process, we modify the conventional error computation by calculating the RMSE for users *before* and *after* their anonymization. This new error quantification approach for anonymized recommender data is called *target deviation* and is defined as follows: For user u in the challenge set, we (the data owner) identify u', the anonymized version of u, among other anonymized users, and then output the predicted ratings of u' as our predictions for u. The advantage of target deviation is the direct and simple quantification of differences before and after the anonymization by leveraging the background knowledge of the data owner [3]

Existing utility models of relational microdata (e.g., the ratio of nodes in the generalization taxonomy trees ([18, 33]), the distance of distributions of the original and anonymized datasets [19], and the estimation errors of aggregate query answers [27]) cannot be applied to the recommender system,

[2]The Internet Movie Database, http://www.imdb.com/

[3]The data owner is able to uniquely identify u' from u by keeping track of the anonymization process, whereas the public cannot.

whose targeting utility is the prediction accuracy of recommendations. Our target deviation method allows us to easily compare the original data values and their anonymized ones, which is a general approach for computing the amount of information change during anonymization.

7.2 Evaluation of the Utility of Anonymized Data

We design several sets of anonymization experiments that are described in Table 1.

Experiment Series	RMSE*
Original Data	0.951849
Padded Anonymization ($k = 5$)	0.95970
Padded Anonymization ($k = 50$)	0.95871
Simple Anonymization ($k = 5$)	2.36947
Simple Anonymization ($k = 50$)	2.3771

Table 1: The setup of experiments. *The target deviation RMSEs are reported in both padded anonymization and simple anonymization experiments (See Section 7.2).

Both the *simple anonymization* and *padded anonymization* experiments follow the 3-step padding-clustering-homogenization sequence described in Section 4. The only difference is in the homogenization step (Step 3). In the former, we compute and publish the averaged ratings on the original data, whereas in the latter we compute and publish the averaged ratings on the padded SVD data. The RMSE results of both experiments under different k values are shown in Table 1.

The high RMSE values (2.36947 and 2.3771) for the simple anonymization is due to both the limited data size and the sparsity of the anonymized data. With $k = 50$, there are only 9,294 anonymized users (i.e., groups) in the public released data. Among them, 80% of the ratings are null if the averaging is done on the original data (as in simple anonymization), as opposed to no null ratings in the padded data (used for padded anonymization).

7.3 Data Characterization and Clustering Evaluation

We characterize the data sparsity and compare the sparsity before and after our padding procedure. We simply count the number of ratings that fall within a range. The results are shown in Table 2. It is clear that padding significantly changes the distribution of ratings in the dataset, in particular, null ratings. Padding data with SVD tremendously reduces the data sparsity, and provides a rich context for identifying similar users, as the pair-wise user similarity computation is more accurate and meaningful.

We characterize the various user similarity metrics that may be used in the clustering algorithm. We evaluate four metrics: closeness-0.5, closeness-1.0, weighted similarity, and our weighted-squared similarity measure described in Section 4.2. Closeness-a is a simple similarity measure on two vectors V_1 and V_2 by counting two corresponding vector entries similar if their difference is within threshold a. Weighted similarity assigns a weight to various ranges to penalize discrepancies. All similarity values are normalized to within [0,1], and are categorized into 20 disjoint ranges, namely: $[0, 0.05), [0.05, 0.1), \ldots, [0.95, 1.0)$. Figure 4 (a) shows the distribution of pair-wise similarities for 5000 users under the four measures before clustering (after padding), and (b) shows the distribution of similarities between users within one single cluster. The shift in distribution to the left indicates that the clustering algorithm is able to group similar users. However, if a similarity measure is too relaxed (e.g., closeness-1.0), then the similarity values are artificially inflated, which does not provide a good indicator in clustering. In comparison, a more strict similarity measure such as ours (described in Section 4.2) has a more fine-grained ability to distinguish similarities in user profiles.

Rating Range	No. of Ratings in Original Dataset	No. of Ratings After Padding
[0]	98.84%*	0
[1]	0.053%	0.79%
[2]	0.117%	14.12%
[3]	0.334%	46.71%
[4]	0.390%	33.49%
[5]	0.267%	4.89%

Table 2: The rating histograms before and after SVD padding. *This value is the number of zero entries.

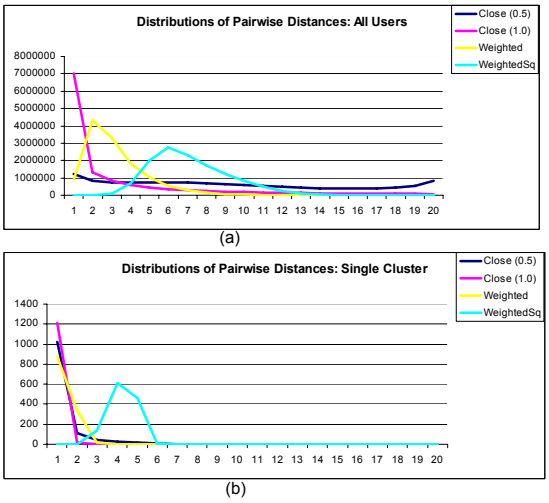

Figure 4: Comparison of four distance metrics in user-user similarity computation. WeightedSq denotes our similarity measure used in this work.

7.4 RMSE By Similar-User Deviation

Target deviation is more accurate in reflecting the information loss incurred during anonymization. It implicitly assumes that the anonymized user is the most similar to herself before the anonymization, as the RMSE is computed by directly comparing the ratings of a user *before* and *after* anonymization. To eliminate the assumption in target deviation, we define a *similar-user deviation*, which is a more realistic method for computing the utility of anonymized data. It is based on user-based collaborative filtering. For a user u in the challenge set, we find the anonymized user v (in the anonymized dataset) that is most similar to u according to a similarity measure. We apply v's ratings as

our prediction for u, and then compute the RMSE for the entire challenge set. The experimental results are shown in Figure 3.

To evaluate the differences between the two deviation metrics, we define *self-rank* of a user u in the challenge set as the rank of her anonymized version u' among other anonymized users in terms of similarity to u. Specifically, compute and sort the similarities between u and all the anonymized users; identify the rank of u'. (Note that the data owner performing the anonymization is able to uniquely identify u' from u, whereas the public cannot.)

In target deviation, the self-ranks are assumed to be one for all users in the challenge set. With similar-user deviation, most self-ranks values are quite low, indicating that target deviation is a good approximation in the error computation. We categorize the self-rank values in Figure 5.

Experiments With Similar-User Deviation	RMSE
Padded Anonymization ($k = 50$)	1.00563
Simple Anonymization ($k = 50$)	1.17525

Table 3: RMSEs computed based on our similar-user deviation definition are reported for both padded anonymization and predictive anonymization experiments with k = 50.

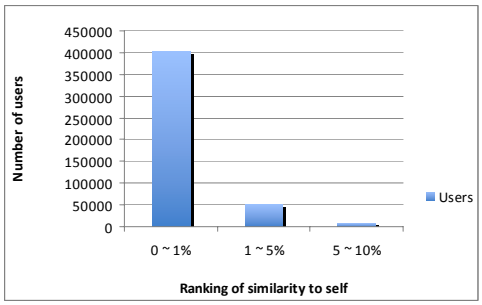

Figure 5: Number of users whose self-rank (See Section 7.4) is within a certain percentage.

Experiment summary Our approach of strategically replacing null entries with padded values has positive impacts on the utility of anonymized data. Namely, we perform a round of prediction before anonymization to reduce data sparsity of the original recommendation data. The experiments show that *padded anonymization* is extremely effective in preserving the data quality with low prediction errors. Our privacy analysis also shows that the padded data improves user privacy as a positive side effect. Surprisingly, the value of k does not have a significant impact on prediction accuracy. From the experiments, we conclude that *homogenization on SVD-padded data (as in our padded predictive anonymization algorithm) is a feasible and utility-preserving approach for anonymizing recommendation data.*

We also quantify and compare the information loss in different experiment setups, in particular when the original data is released as opposed to the padded data. Homogenization on the original data as in the *simple anonymization* method gives much higher RMSE than the homogenization on the padded data, indicating naive anonymization method incurs high information loss and destroys data patterns even with a small k value. This negative finding validates earlier predictions by others [23].

Although preserving prediction accuracy, the padded anonymization method loses authentic data properties and the released data cannot support statistical queries. For example, one is unable to find out what percentage of users have rated a movie. This underlines an intrinsic tradeoff between user privacy and data utility. To mitigate these issues, the padding procedure can be adjusted to more carefully control the degree of boosting, i.e., to what extent the sparsity in the original data is reduced. This study is subject to our future work.

8. RELATED WORK

Our data anonymization problem is related to privacy-preserving relational data, social network data, collaborative filtering, and recommendation systems in general. We described some of the related works in these areas in the following.

Privacy preserving relational databases It has been extensively studied recently. Most of the work considers the attack model as re-identification of individuals by joining the published table with some external tables modeling the background knowledge of users. To defend against this type of attacks, the mechanism of k-anonymity was proposed in [31, 32]. Specifically, a dataset is said to be *k-anonymous* if every group of tuples that are of the same values on the quasi-identifier attributes (i.e., the minimal set of attributes in the table that can be joined with external information to re-identify individual records) consists of at least k tuples. The larger the value of k, the better the privacy is protected. As the improvement of k-anonymity, new notions (e.g., l-diversity [21]) have been proposed to provide stronger privacy. We adapt the concepts of k-anonymity and l-diversity to our problem since they are the most essential and most applicable privacy models. However, as discussed in Section 1, due to various difference of relational databases from recommender system datasets, the k-anonymity techniques for relational cases cannot be applied to recommender systems.

Byun *et al.* came up with a clustering idea for anonymizing relational databases [8]. They presented a solution for database records that are in the form of vectors, where each dimension is a numerical value or a label in a fixed hierarchy. Again, their work is for relational databases, and does not have a direct application to data in the form of a graph.

Cormode et al. studied the problem of privacy-preserving anonymization of bipartite graphs [11]. Similar to our work, their privacy goal is to protect the association between the nodes in two partitions in the graph. Their work concerned unlabeled graphs, whereas we consider labeled bipartite graphs, with the possibility of labels being used as part of the adversary knowledge. We also have different target utility goals, as they aimed to preserve the accuracy of SQL aggregate queries on the released dataset.

Privacy preserving social networks Recommender systems enclose information of social connections. In fact, as in social networks, the data in recommender systems can be represented as graphs as well. Thus the problem of publishing recommender system with preserved privacy is related to the problem of preserving privacy of social network graphs. Privacy-preserving publishing of social network graphs have attracted much attention recently [16, 3, 36]. The identity

of the nodes can be de-anonymized by the structure related to the node [16, 36]. Furthermore, [3] pointed out the adversary can even have active attack by joining the social networks, creating a small number of new user accounts with edges to the targeted users, and consequently creating a pattern of links among the new accounts so that it will stand out in the anonymized graph structure. To address the de-identification issue of publishing general social network data, Hay, Miklau, *et al.* were the first to formally define and quantify the privacy risks. Their *k-candidate anonymity* requires that there are at least k candidate nodes that have the same structure as any node in the original data [16]. They used a generalization approach for anonymizing social network data, and demonstrated the success of it in preserving the utility of anonymized network data. Unlike recommendation data, social network is not believed to be sparse. On the contrary, the small world effect and six-degree of separation are commonly observed in social networks. Further, unlike our work, these work do not include the labels on the graph as part of the attack. These characteristics make generalization or randomization approach possible. In comparison, for sparse recommendation data, suppression and generalization are conjectured to be ineffective in preserving the utility of anonymized data [23]. Therefore, our approach is based on strategically inserting fake ratings with calculated values.

Zhou et al. requires that every node must have at least *k-1* nodes that have the same neighbors in the anonymized graph [36]. Compared with social network graphs, recommender system differs in a few aspects. First, unlike in the released social network graphs that every node is de-identified, recommendation graphs contain considerable amounts of nodes whose identifications are revealed (e.g., the name of movies, music, books, etc.). These nodes make the adversary easier to infer private information and consequently more challenging to publish a privacy-preserved review graph. Second, unlike social network graphs which are normally dense, most real-world recommender system datasets are very sparse [23]. Sparsity increases the difficulty of finding similar individuals for privacy concern. It may also degrade the utility of the anonymized recommendation data. Therefore, the anonymization techniques on social network graphs cannot be directly applied to recommendation data.

Privacy preserving collaborative filtering Canny proposes two schemes for privacy-preserving collaborative filtering [9, 10]. These schemes consider that a community of users compute a public *aggregate* of their ratings without exposing any individual users' rating. Therefore the ratings are encrypted by homomorphic encryption mechanism. These two schemes are built on peer-to-peer systems. With the similar fashion, the work by [17] also considers encryption on ratings. Polat et al. considers the similar problem in a centralized framework [25], in which users send their data to a central server that will conduct the CF. Instead of encryption, Polar et al. proposed to use randomized perturbation techniques to disguise private ratings before they are sent to a central place (the data collector), such that the data collector cannot derive the truthful ratings while he/she still be able to conduct collaborative filtering from the disguised data. They also proposed the randomization-based approach for SVD-based collaborative filtering[26]. The privacy concern of all the above work is the real value of the ratings, while we consider identification of users as well as their ratings as private. By our attack model, even though the ratings may be encrypted/randomized by the above work, the adversary still can re-identify the users by knowing which items are rated. Baraglia et al. considers a different privacy attack model that is applied on the classfication in Web Usage Mining (WUM) system [4]. It is orthogonal to our work.

9. CONCLUSIONS

In this paper, we showed that utility-preserving anonymization for recommendation data is feasible, if careful padding is performed to reduce data sparsity. We gave SVD as a concrete padding technique before anonymization. Our approach is called predictive anonymization, as a round of prediction is sufficient for the padding step. We also formally defined the model and developed a practical and efficient anonymization algorithm called *Predictive Anonymization*. Our empirical studies using the Netflix dataset demonstrate the effectiveness of our *Predictive Anonymization* algorithm in preserving utility of the anonymized data.

10. ADDITIONAL AUTHORS

11. REFERENCES

[1] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu, "Achieving Anonymity via Clustering," in Proc. of ACM PODS, 2006, pp. 153-162.

[2] K. Ali and W. van Stam, "Tivo: making show recommendations using a distributed collaborative filtering architecture", KDD, 2004.

[3] L. Backstrom, C. Dwork, J. Kleinberg, "Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography", WWW 2007.

[4] R. Baraglia, C. Lucchese, S. Orlando, M. Serrano, F. Silvestri, "A Privacy Preserving Web Recommender System", SIGAPP, 2006.

[5] R. M. Bell, Y. Koren, C. Volinsky, "The BellKor Solution to the Netflix Prize", http://www.research.att.com/ volinsky/netflix/ProgressPrize2007BellKorSolution.pdf.

[6] P.S. Bradley, K.P Bennett, A Demiriz, "Constrained K-Means Clustering", Microsoft research technical report, MSR-TR-2000-65, 2000.

[7] J. S. Breese, D. Heckerman, C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", Technical report MSR-TR-98-12, Microsoft Research, 1998.

[8] Ji-Won Byun, Ashish Kamra, Elisa Bertino, Ninghui Li, "Efficient k-anonymization Using Clustering Techniques", DASFAA 2007.

[9] J. Canny, "Collaborative Filtering with Privacy", IEEE Symposium on Security and Privacy, May 2002.

[10] J. Canny, "Collaborative filtering with privacy via factor analysis", SIGIR, August 2002.

[11] G. Cormode, D. Srivastava, T. Yu, Q. Zhang, "Anonymizing Bipartite Graph Data Using Safe Groupings", VLDB, 2008.

[12] G. Ghinita, Y. Tao, P. Kalnis. "On the Anonymization of Sparse High-Dimensional Data", ICDE, 2008, pp. 715-724.

[13] D. Goldberg, D. Nichols, B. M. Oki, D. Terry, "Using collaborative filtering to weave an information tapestry", Communications of the ACM, vol. 35, No. 12, pp. 61-70, 1992.

[14] K. Y. Goldberg, T. Roeder, D. Gupta, C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm", Journal of information retrieval, 4(2):133-151, 2001.

[15] N. Good, B. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, J. Riedl. "Combining Collaborative Filtering With Personal Agents for Better Recommendations". AAAI'99.

[16] M. Hay, G. Miklau, D. Jensen, D. Towsley, P. Weis, "Resisting Structural Identification in Anonymized Social Networks", VLDB, 2008.

[17] C. Hsieh, J. Zhan, D. Zeng, F. Wang, "Preserving Privacy in Joining Recommender Systems", International Conference of Information Security and Assurance (ISA), 2008.

[18] V. S. Iyengar, "Transforming Data to Satisfy Privacy Constraints", SIGKDD, pp 279-288, 2002.

[19] D. Kifer, J. Gehrke, "Injecting Utility into Anonymized Datasets", SIGMOD 2006.

[20] S. Lloyd, "Least Squares Quantization in PCM", IEEE Transactions on Information Theory, 1982.

[21] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam. "l-Diversity: Privacy Beyond k-anonymity", ICDE 2006.

[22] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observation", In 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967.

[23] A. Narayanan, V. Shmatikov, "How To Break Anonymity of the Netflix Prize Dataset", S&P, 2008.

[24] M. Nergiz, C. Clifton, "Thoughts on k-anonymization", *Data Knowl. Eng.*, 63(3):622–645, 2007.

[25] H. Polat, W. Du, "Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques", ICDM, 2003.

[26] H. Polat, W. Du, "SVD-based Collaborative Filtering with Privacy", SAC, 2005.

[27] V. Rastogi, D. Suciu, S. Hong, "The Boundary Between Privacy and Utility in Data Publishing", VLDB 2007.

[28] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews", Proceedings of ACM Conference on Computer Supported Cooperative Work, pp. 175-186, 1994.

[29] B. Sarwar, M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., and Riedl, J.. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. CSCW, 1998.

[30] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, "Application of Dimensionality Reduction in Recommender System - A Case Study"', ACM WebKDD 2000 Workshop.

[31] P. Samarati, Latanya Sweeney, "Generalizing data to provide anonymity when disclosing information", PODS, 1998.

[32] L. Sweeney, "k-anonymity: a model for protecting privacy", International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5):557570, 2002.

[33] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, A. Fu, "Utility-Based Anonymization Using Local Recoding", SIGKDD, 2006.

[34] B. Thompson, D. Yao, "Union-Split Clustering Algorithm and Social Network Anonymization", AsiaCCS, 2009.

[35] B. Webb. Netflix update: Try this at home. http://sifter.org/∼simon/journal/20061211.html, 2006.

[36] B. Zhou, J. Pei, "Preserving Privacy in Social Networks Against Neighborhood Attacks", ICDE, 2008.

[37] Netflix Recommender Framework. http://benjamin-meyer.blogspot.com/2006/10/netflix-prize-contest.html?program=NetflixRecommenderFramework.

APPENDIX

Proof of Theorem 6.3

Suppose an adversary, using a label-based attack, is able to identify the anonymization group containing user u. Based on the existence of a link to item o in the released review graph, the adversary would like to infer whether user u gave a rating for o in the original dataset. Let $Pr(u,o)$ denote the probability that user u rated item o in the original dataset, and let $Pr(C,o)$ be the unconditional probability that at least one user in anonymization group C rated o. We wish to calculate $Pr((u,o)|(C,o))$, the probability that user u rated item o, given that the edge exists in the released anonymized review graph.

By Bayes' Rule, we have that $Pr((u,o)|(C,o)) = Pr((C,o)|(u,o)) * Pr(u,o)/Pr(C,o)$. First note that $Pr((C,o)|(u,o)) = 1$ directly from our anonymization procedure, and we also have that $Pr(u,o) = p$. Furthermore, since we have assumed that each user has rated item o independently with probability p, we can find a bound on $Pr(C,o)$ as follows:

$$\begin{aligned} & Pr(C,o) = 1-(1-p)^k \\ = \;& 1-(1-kp+\binom{k}{2}p^2-o(p^3)) \\ = \;& kp-\binom{k}{2}p^2+o(p^3) \geq kp-\binom{k}{2}p^2 \end{aligned}$$

Combining these results, we get that

$$\begin{aligned} & Pr((u,o)|(C,o)) \leq \frac{1\cdot p}{kp-\binom{k}{2}p^2} \\ = \;& \frac{1}{k}\left(\frac{kp}{kp-\binom{k}{2}p^2}+\frac{-\binom{k}{2}p^2}{kp-\binom{k}{2}p^2}\right)+\frac{1}{k}\left(\frac{\binom{k}{2}p^2}{kp-\binom{k}{2}p^2}\right) \\ = \;& \frac{1}{k}+\frac{1}{k}\left(\frac{\frac{1}{2}k(k-1)p^2}{kp-\frac{1}{2}k(k-1)p^2}\right) \\ = \;& \frac{1}{k}+\frac{1}{k}\left(\frac{(k-1)p}{2-(k-1)p}\right) \leq \frac{1}{k}+\frac{p}{2-kp} \end{aligned}$$