A Privacy-Preserving Framework for Personalized, Social Recommendations

Zach Jorgensen
North Carolina State University
890 Oval Dr. Raleigh, NC
zjorgen@ncsu.edu

Ting Yu
North Carolina State University &
Qatar Computing Research Institute
Doha, Qatar
tyu@{ncsu.edu, qf.org.qa}

ABSTRACT

We consider the problem of producing item recommendations that are personalized based on a user's social network, while simultaneously preventing the disclosure of sensitive user-item preferences (e.g., product purchases, ad clicks, web browsing history, etc.). Our main contribution is a privacy-preserving framework for a class of social recommendation algorithms that provides strong, formal privacy guarantees under the model of differential privacy. Existing mechanisms for achieving differential privacy lead to an unacceptable loss of utility when applied to the social recommendation problem. To address this, the proposed framework incorporates a clustering procedure that groups users according to the natural community structure of the social network and significantly reduces the amount of noise required to satisfy differential privacy. Although this reduction in noise comes at the cost of some approximation error, we show that the benefits of the former significantly outweigh the latter. We explore the privacy-utility trade-off for several different instantiations of the proposed framework on two real-world data sets and show that useful social recommendations can be produced without sacrificing privacy. We also experimentally compare the proposed framework with several existing differential privacy mechanisms and show that the proposed framework significantly outperforms all of them in this setting.

1. INTRODUCTION

In an age where Internet users are constantly overloaded with information, recommendation systems have become an essential tool for separating relevant and interesting content from the mounds of chaff. Recent years have witnessed the astounding growth and penetration of online social networks, spurring interest in the idea of leveraging social relations to generate more useful recommendations. A wide variety of social recommendation approaches have appeared in the literature, some of which augment traditional collaborative filtering approaches with social similarity measures (e.g. [13, 21, 35]) while others are based on random walks (e.g. [18]), probabilistic matrix factorization (e.g. [7]) and collaborative topic regression (e.g. [28]). In many cases, these social recommendation approaches have been shown to outperform traditional, socially agnostic approaches.

Although the benefits of using social information in the recommendation process can be significant, it also poses a significant threat to personal privacy. In particular, social recommendations may allow one user to easily make valid inferences about another user's preferences for sensitive items. Although this threat is also to some extent present in traditional, socially-agnostic recommenders, social recommenders are more vulnerable to privacy attacks for several reasons: (1) the recommendations are, by nature, derived from smaller, more targeted sets of users, (2) the recommendations sources are typically people known by the receiver, and (3) the users of a social network have considerable power to affect the structure of the social network (and therefore the input to the social recommender). These facts enable simple but powerful privacy attacks (Section 2.3). In addition to protecting individual privacy and reducing liability for recommendation service providers, privacy-preserving approaches for social recommendation will likely encourage broader participation by providing users with "peace of mind"; this is clearly a win-win for all parties.

It is well known that syntactic privacy approaches, like k-anonymity [30], do not guarantee privacy in the presence of colluding adversaries or those with auxiliary information [23]. Thus, in this work we aim for provable privacy guarantees, enabled by a mathematically rigorous privacy model known as *differential privacy* [8]. Informally, differential privacy guarantees that almost nothing can be learned about individual input records based on the output of the algorithm; thus a user loses nothing by participating. Differential privacy is usually achieved by injecting random noise that is carefully calibrated according to the algorithm's *sensitivity*—the maximum change in the output resulting from the addition or removal of a single input record. The main challenge in applying differential privacy is how to balance privacy with utility. Social recommendation presents an especially difficult challenge due to its inherent high sensitivity. The high sensitivity comes from the fact that a single user may have many neighbors in the social graph, and hence may affect the recommendations received by many users.

Machanavajjhala et al. recently presented somewhat negative theoretical results suggesting the unfeasibility of producing social recommendations that are both accurate and private [24]. However, the social graph model used in their

10.5441/002/edbt.2014.51

analysis made no distinction between item preferences and social relations—both were considered to be *equally sensitive*. We observe that this is often not the case in practice. For instance, in many popular online social networks, including (but not limited to) Orkut, Twitter, MySpace, RenRen and Weibo, the user-to-user edges are visible to everyone, and there is no option to change that setting. On the other hand, item preferences are typically considered to be sensitive in practice; for example, the content that a user "likes" is usually only shared with the user's friends. Item preferences could also represent behavior such as ad clicks, product purchases, photo views, or website visits—these are all things that might be useful in a social recommendation system, but they are also things that most people would not want displayed to the public, or even to their friends in some cases. Motivated by the observations above, we explore differentially private social recommendation in the setting where social relations are *public knowledge*, while user-to-item edges are *private* and must be protected. Within this context, our task is to generate top-n ranked lists of item recommendations, personalized for each individual user based on his/her social relations, while simultaneously preventing the leakage of information about the presence (or absence) of individual item preferences through the output (i.e., the recommendations themselves).

Our contributions are summarized as follows. First, we present a formalization of the social recommendation problem that explicitly differentiates user nodes from item nodes and social relations from item preferences; this distinction enables us to decouple the computations on the typically insensitive social graph from those on the often sensitive item preferences.

Second, we develop a privacy-preserving framework for a natural class of social recommendation algorithms based on structural similarity measures, that provides differential privacy guarantees for user-item preferences. Our approach achieves strong privacy and high utility by incorporating a novel user clustering phase that groups users according to the social graph structure, significantly reducing sensitivity, and hence the amount of noise required to satisfy differential privacy. Although the clustering phase introduces some approximation error, we show that its impact is small compared to that of the noise it replaces. To the best of our knowledge, this is the first work to propose an effective approach for achieving differential privacy in social recommendation systems, without significantly degrading accuracy.

Finally, we present empirical results for several concrete instantiations of the proposed framework on two real-world data sets, demonstrating that it is possible to make accurate social recommendations while simultaneously providing a high degree of privacy for item preferences. Additionally, we experimentally compare our approach against two traditional methods for achieving differential privacy, as well as to two recently proposed differential privacy mechanisms ([17,34]) that have proven successful in similar settings. Our results show our approach to significantly outperform these other approaches under both high and low privacy settings.

The rest of the paper is organized as follows. Section 2 formalizes our model and setting, while Section 3 reviews differential privacy. In Section 4 we review relevant related work and then in Section 5 we present our framework for privacy-preserving social recommendation. Section 6 presents our experimental results and Section 7 concludes the paper.

2. MODEL

In this section we formalize our setting and specify the basic class of non-private social recommenders that will be targeted by our privacy-preserving framework.

2.1 Setting

Our setting assumes the existence of two distinct graphs, a social graph and a preference graph, that comprise the input to a social recommendation system. They are defined as follows:

DEFINITION 1 (SOCIAL GRAPH). *A* social graph, $G_s = (U, E_s)$, *consists of a set of user nodes,* U, *and a set of edges,* E_s, *where a* social edge $(u, v) \in E_s$ *represents a social connection (e.g. friendship) between two users* $u, v \in U$.

DEFINITION 2 (PREFERENCE GRAPH). *A* preference graph, $G_p = (U, I, E_p)$, *is a bipartite graph consisting of the same set of users* U, *a set of items,* I, *and a set of directed edges,* E_p. *A* preference edge $(u, i) \in E_p$ *expresses a positive preference of user* $u \in U$ *for item* $i \in I$.

For simplicity, we assume that the preference graph is unweighted, or equivalently that every edge (u, i) has weight $w(u, i) = 1$. (For notational convenience, we assume that $w(u, i) = 0$, $\forall (u, i) \notin E_p$). Many real-world recommendation settings can be modeled with unweighted preference graphs; for example, an edge (u, i) might indicate that u has purchased product i, that u has listened to a song by artist i, that u 'Likes' content i, or that u has visited web page i, to name a few. Our approach could be easily extended to support weighted edges (e.g., ratings).

2.2 Personalized, Social Recommenders

We consider a family of personalized, social, top-n recommendation systems (henceforth referred to simply as *social recommenders*, or just *recommenders* where the context is clear) that are based on a utility function that takes only G_s and G_p as input and computes for all user-item pairs v, i a utility value μ_u^i indicating the utility of recommending item i to user u. The social recommender outputs, for each *target user* $u \in U$, a personalized *recommendation list*, R_u, consisting of the top-n items, ranked by utility.

Our model assumes that the utility function depends on an underlying *structural similarity measure*, denoted $sim(u, v)$, that operates *solely on the structure* of G_s and returns a positive numeric value indicating the similarity of users $u, v \in U$ (or 0 if not similar). Since these similarity measures operate solely on the social graph, we refer to them henceforth as *social similarity measures*. Abusing this notation slightly, we will frequently use $sim(u)$ to denote the set of users with non-zero similarity to u, i.e. $\{v \in U | sim(u, v) > 0\}$. We call $sim(u)$ the *similarity set* of u.

Many existing structural similarity measures that can be plugged into this model (see [22] for a good survey). Prior work in the link analysis literature has demonstrated the usefulness of structural similarity measures for predicting missing edges in social networks [20], and other graphs [22]; moreover, several measures have been shown to outperform collaborative filtering approaches for recommendation on certain data sets [13], making them appealing candidates for the social recommendation task. For concreteness, we will consider four well-known social similarity measures in

this paper. Let $\Gamma(u)$ denote the set containing u's immediate neighbors in G_s. In the context of two arbitrary users $u, v \in U$, the four measures are defined briefly below (see [20, 22] for more details):

- **Common Neighbors (CN)**: $sim(u, v) = |\Gamma(u) \cap \Gamma(v)|$.
- **Graph Distance (GD)**: $sim(u, v) = \frac{1}{d}$, where d is the length of the shortest path between u and v.
- **Adamic/Adar (AA)**: $sim(u, v) = \sum_{x \in \Gamma(u) \cap \Gamma(v)} \frac{1}{log|\Gamma(x)|}$.
- **Katz (KZ)**: $sim(u, v) = \sum_{l=1}^{k} \alpha^l \cdot |paths^l_{uv}|$, where $paths^l_{uv}$ is the set of all length-l paths between u and v, and α is a small damping factor (e.g. 0.05 or 0.005).

For GD and KZ it is common to limit d and k (e.g. to 2 or 3) respectively, since in social graphs, the number of reachable users explodes after 2 hops due to the small-world property [27] of social graphs.

The utility of recommending an item i to a user u is computed by a *utility query*:

Definition 3 (Utility Query). *Given a similarity measure,* sim*, the utility of recommending an item* i *to user* u *is given by:*

$$\mu^i_u = \sum_{v \in sim(u)} sim(u, v) \times w(v, i) \quad (1)$$

We now formally define *top-N social recommender* as follows:

Definition 4 (Top-N Social Recommender). *Given a social graph $G_s = (U, E_s)$, a preference graph $G_p = (U, I, E_p)$ and a social similarity measure* sim*, a* top-N social recommender *identifies, for all users $u \in U$, a size N ranked list of items $R_u \subset I$ with the highest utility for user u.*

We remark that although it can be beneficial to use both social and non-social data in the recommendation process, our focus is on purely social recommenders in this paper. We plan to study such hybrid recommenders in a future work.

2.3 Assumptions and Attacker Model

The social recommender, as well as the underlying data, is assumed to reside with a trusted and secured central party and that the recommender has full, unfettered access to the data. In practice, the recommender would typically (though not always) reside with the owner of the preference data, while the social graph data would be obtained from a third-party social network though an API (e.g. Facebook's Open Graph, Twitter API, etc.). We remark that the trusted server model is not uncommon in previous works on differential privacy; in particular, the work of [26] that studies differential privacy for non-social recommenders makes the same assumption.

Second, we assume that G_s and G_p are static at the time that recommendations are computed. That is, we take a snapshot of the graphs at time t_i and generate recommendations for all users from that snapshot. In this paper, we focus on the problem of making private social recommendations *only for a single snapshot*. Enforcing differential privacy over dynamic graphs is a non-trivial extension and we leave it as a subject for future work. We remark that most of the related works discussed in this paper are also subject to the same limitation (e.g., [17, 26, 34]).

Adversary Model. We consider a powerful, informed adversary, A, whose goal is to deduce the existence (or absence) of a specific preference edge, the *target edge*. A is assumed to have complete knowledge of the social graph and of the inner-workings of the recommendation algorithm. Also, consistent with the definition of differential privacy, A could potentially have knowledge of all preference edges in G_p, except for the target edge. Finally, A may observe *all* recommendations output by the social recommender. The latter two assumptions account for collusion and arbitrary background knowledge.

The adversary modeled above may, at first glance, seem unrealistically powerful; however, we argue, by way of the following example, that this is in fact the appropriate model. For concreteness, consider the CN (or AA) similarity measure defined in Section 2.2. One possible attack proceeds as follows: (1) A identifies an immediate neighbor a of the victim in the social graph that has no other neighbors; (2) A then creates a fake user account b and establishes a new social edge (b, a), either through collusion or a simple profile-cloning attack[1]; (3) A observes the recommendations received by b. Since b's only friend is a, and only the victim and b are friends with a, all recommendations received by b reveal the existence of one of the victim's preference edges. In the case where the victim has no neighbors with a single friendship edge, the attacker could link two Sybil nodes together and then trick the victim into accepting a friendship request from one of them using a profile cloning attack. This attack could also be extended to the GD and KZ similarity measures by chaining $d - 1$ or $k - 1$ Sybils together, respectively, depending on the maximum distance cutoff being used.

The thing to note from the example above is that, from the defender's point of view, any node in the graph could be the attacker, any node could be the victim, and there is no way to know which nodes are colluding; moreover, there is no way to know which edge is the target edge and which edges the attacker has knowledge of due to auxiliary information. Thus, to ensure privacy, we must provide privacy guarantees for every preference edge as though it were the target edge.

2.4 Measuring Recommendation Accuracy

The primary challenge to incorporating differential privacy into social recommenders (and indeed any system) is how to do so without significantly harming the utility, or accuracy, of the system. Common accuracy metrics, such as *precision* and *recall*, are inappropriate here because they fail to take into account the rank and utility of the items. For example, failing to recommend the top ranked item would incur the same penalty as failing to recommend the N^{th} item, even though higher ranked items have higher utility for the user; similarly, if an item i with utility μ appears in the true ranking but is replaced by a different item j, also with utility μ, in the private ranking, it will incur an unnecessary penalty.

Based on these considerations, we use *normalized discounted cumulative gain (NDCG)* [15] as our accuracy metric in this paper. Formally, let $\widehat{A}$ denote a top-N, private social recommender and let $\widehat{R^N_u}$ denote the list of ranked

[1] A profile cloning attack involves setting up a fake node to impersonate an existing friend of a victim and then issuing a friend request to the victim. [3] showed a success rate > 60% for this attack in a recent study of real Facebook users.

recommendations output by $\widehat{A}$ for user u. Likewise, let A denote the non-private counterpart to $\widehat{A}$ and let R_u^N denote the ranked recommendations output by A. We will use μ_u^i to denote the *ideal utility* for item i with respect to user u—that is, the utility computed by A. Then, the NDCG for the top N recommendations, for a given data set, is defined as:

$$NDCG@N = \sum_{u \in U} \frac{DCG(\widehat{R_u^N}, u)}{DCG(R_u^N, u)} \times \frac{1}{|U|} \quad (2)$$

where $DCG(X, u) = \sum_{i \in X} \mu_u^i / \max(1, \log_2 p(i)+1)$ and $p(i)$ is the index of item i in X. An *NDCG@N* of one indicates that the top N items recommended by the private recommender are perfectly ranked by their true utilities, while an *NDCG@N* of zero has the opposite meaning.

3. PRIVACY MODEL

To meet the challenges of our adversary, we adopt the mathematically rigorous differential privacy model [8], which offers provable privacy guarantees against such adversaries. The guarantee provided by differential privacy bounds the ratio of the adversary's prior and posterior belief that any one element was present in the input, given an output. Differential privacy is traditionally defined as follows:

Definition 5 (ϵ-Differential Privacy [8, 9]). *A randomized algorithm A satisfies ϵ-differential privacy if for all inputs D_1 and D_2, differing in one element (i.e. $|D_1| = |D_2| + 1$ and $D_1 \subset D_2$, or vice-versa), and for any subset S of the possible outputs Range(A),*

$$Pr[A(D_1) \in S] \leq exp(\epsilon) \times Pr[A(D_2) \in S]$$

The intuition is that, privacy is preserved when the output of an algorithm is insensitive to the presence or absence of any single element in the input data set. The *privacy parameter* ϵ determines the strength of the privacy guarantee and is typically chosen to be in the range $[0.01, 1.0]$. Extending definition 5 to our setting, we have

Definition 6 (Private Social Recommender). *For a fixed social graph, G_s, a social recommender R is ϵ-differentially private if for* all *preference graphs $G_{p1} = (U, E_{p1})$ and $G_{p2} = (U, E_{p2})$ that differ by at most one edge (i.e. $E_{p1} \subseteq E_{p2}$ and $|E_{p2}| - |E_{p1}| \leq 1$, or vice versa), and for any subset S of the possible recommendation lists in Range(R),*

$$Pr[R(G_{p1}, G_s) \in S] \leq exp(\epsilon) \times Pr[R(G_{p2}, G_s) \in S]$$

Thus, a social recommender preserves the privacy of preference edges if the probability of outputting any list of recommendations is (almost) the same, with or without *any* one preference edge.

3.1 Achieving Differential Privacy

Differential privacy can be achieved by injecting carefully chosen random noise. The magnitude of the noise is calibrated according to the *global sensitivity* of the algorithm, or the maximum extent to which its output can be affected by any individual input element. Formally,

Definition 7 (Global Sensitivity [9]). *The* global sensitivity *of an algorithm $A : D \rightarrow R^d$, is*

$$\Delta_A = \max_{D_1, D_2} \|A(D_1) - A(D_2)\|_1$$

for all D_1, D_2 differing on one element.

The *Laplace mechanism* is commonly used for achieving differential privacy [9]. In general, an algorithm with sensitivity Δ can be made ϵ-differentially private by adding random noise drawn from the Laplace distribution with mean 0 and scale $\frac{\Delta}{\epsilon}$ to its output. We will subsequently use the notation $Lap(\frac{\Delta}{\epsilon})$ to denote such noise.

Theorem 1 (Laplace Mechanism). *An algorithm $A : D \rightarrow R^d$ with sensitivity Δ_A, that adds independent noise from $Lap(\frac{\Delta_A}{\epsilon})$ to each of the d output terms satisfies ϵ-differential privacy [9].*

The noise depends only on Δ_A and ϵ and is therefore independent of the underlying data. ϵ can be adjusted to provide the desired level of privacy protection; a smaller ϵ leads to a stronger privacy guarantee, while a larger ϵ weakens the guarantee. The variance of $Lap(\lambda)$ is $2\lambda^2$, and thus the expected error is $\sqrt{var(Lap(\lambda))} = \frac{\sqrt{2}\Delta}{\epsilon}$.

The next two theorems state important properties of differentially private algorithms that we will refer to in later discussions.

Theorem 2 (Sequential Composition [25]). *For a set of computations $A_1, \ldots, A_i, \ldots, A_n$, each of which satisfies ϵ_i-differential privacy, the sequential execution of those computations on non-disjoint inputs satisfies $(\sum_i \epsilon_i)$-differential privacy.*

In other words, the privacy guarantee degrades when multiple differentially private computations operate on the same input. When the inputs are disjoint, on the other hand, the resulting sequence of computations is (max_{ϵ_i})-differentially private. This is captured in the following theorem:

Theorem 3 (Parallel Composition [25]). *For a set of computations, each of which satisfies ϵ_i-differential privacy, the sequential execution of those computations on disjoint inputs also satisfies max_{ϵ_i}-differential privacy.*

In the next section, we briefly discuss related work. We then present our proposed framework in Section 5.

4. RELATED WORK

Aside from the previously mentioned theoretical work of Machanavajjhala et al. [24], there have been few works to study privacy in social recommendation. Danezis et al. [6] proposed a k-anonymity like approach for private social recommendations; however, this approach, like k-anonymity in general, is not resistant to collusion or auxiliary information attacks. It is these limitations that motivate our use of differential privacy, which provides strong guarantees in the presence of such attacks.

McSherry and Mirinov [26] integrated differential privacy into traditional, *non-social* recommendation systems, such as those used in the NetFlix Prize competition. Specifically, they considered *item-based* collaborative filtering algorithms that make recommendations to users based on a sanitized, global, item-item covariance matrix that is constructed using item ratings contributed by all users. This is different from our setting, in which the recommenders are necessarily *user-based* and where recommendations are personalized according to each user's social network. In the social setting, the sensitivity–and hence the amount of noise required to satisfy differential privacy–grows with the size of the social graph, which can be very large in practice.

Private approaches for computing social recommendations in distributed settings have recently been proposed [12, 16]. However, these approaches use cryptographic techniques to eliminate the need for a central aggregator with access to all of the raw preference data; these approaches do not prevent privacy leaks through the *output* of the recommendation system (i.e., the recommendations themselves), which is our goal in the present work.

Since differential privacy was first formalized, a significant body of work has emerged leading to more advanced mechanisms for achieving differential privacy. We now briefly review a selection of recent works that we feel are most relevant to the present work.

A number of recent works ([1, 11, 31–33]) focus on private histogram publishing, where the goal is to release a noisy version of a histogram that can be used to answer arbitrary range queries. Histogram counts have a sensitivity of one, since each tuple of the underlying data is used in a single count. In contrast, the utility queries that are computed in the social recommendation problem are highly correlated, since a single preference edge can be used in many different utility queries. Thus these approaches are unsuitable for our problem.

Differentially private frequent itemset mining (FIM) for transaction data is studied by [2] and [36]. The goal of FIM, in general, is to identify top K sets of items that appear together most frequently in the transactions, which is similar to the recommendation problem in some respects. The critical difference is that in FIM, a single global ranking is produced, while in social recommendation a *personalized* ranking is produced *for every user* from *overlapping* sets of preference edges. Personalization implies significantly higher sensitivity, and hence more noise.

The *Matrix Mechanism (MM)* [19], and the closely related *Low-rank Mechanism (LRM)* [34] consider how to optimize utility for a workload of correlated, linear queries under differential privacy. The idea is to find a different, smaller set of queries, called a *strategy* that can be answered more accurately than the original workload, and to use noisy answers to the strategy queries to derive approximate answers for the original workload. [19] formalizes this idea but does not provide an efficient method for deriving suitable strategies for arbitrary workloads. The low-rank mechanism (LRM) [34] uses low-rank approximation to derive strategies. In Section 6, we show that LRM can be adapted naturally to our setting, but with negative results.

Very recently, [17] considered a similar problem to ours (although in an entirely different context), in which the goal was to privately publish a set of counts (e.g., user 'check-ins' at different locations) where a single user's data can affect multiple counts. The proposed *group and smooth* (GS) approach involves grouping similar counts, which could be approximated by their noisy mean. In Section 6.4, we show that the ideas used in GS can be adapted to our task, but that the resulting approach is unable to effectively cope with the high sensitivity of our task.

5. PROPOSED FRAMEWORK

Our goal in this paper is to create a framework that can be used to easily transform non-private, top-n social recommenders (as defined in Section 2.2), into differentially private social recommenders that are able to provide non-trivial privacy guarantees while also maintaining good recommendation accuracy for most users.

At a high level, our approach involves clustering the preference edges of G_p into *disjoint* clusters such that the edges in each cluster frequently co-occur in many different utility queries. This process minimizes the sensitivity of the utility queries and greatly reduces the amount of noise required to satisfy differential privacy. However, the clustering process essentially trades perturbation error for approximation error and the challenge is to find a clustering such that the approximation error is outweighed by the reduction in perturbation error. The problem is further complicated by the fact that if the preference graph is used to guide the clustering process in any way, then additional noise (and hence additional perturbation error) must be injected to ensure that no privacy is leaked through the clustering process. The surprising finding of this paper is that the insensitive structure of the social graph alone can be used to guide the creation of an effective clustering, without incurring any additional perturbation error.

5.1 Rationale

To better explain the rationale behind our approach, we begin with a high-level discussion of two strawman approaches that satisfy differential privacy, but that fail to provide useful recommendation accuracy. We then use the short-comings identified in the strawman approaches to motivate our proposed user clustering approach, which achieves the same level of privacy, but with a significantly higher utility. We start by making some simplifying observations and introducing additional notation to simplify the discussion.

Our goal is to design a differentially private algorithm that takes a social graph $G_s = (U, E_s)$ and a preference graph $G_p = (U, I, E_p)$ and returns, for every user in U, a personalized list of the top N items ranked by utility. Recall that in our model, generating a list of recommendations for an arbitrary user u involves first computing the set of utility queries $\mu_u = \{\mu_u^i | i \in I\}$, then sorting the items in I according to their utilities μ_u, and finally returning the top N items. Our challenge, then, is to design an algorithm that can compute perturbed answers to the utility queries in a differentially private manner; if we can answer the utility queries privately, then the act of sorting and outputting the top N items also preserves privacy, since post-processing over sanitized data does not pose any additional privacy risk.

A second simplifying observation is that answering utility queries for an arbitrary item i, i.e., $\mu_u^i = \sum_{v \in sim(u)} sim(u, v) \cdot w(v, i)$, requires only the social graph (to compute the similarity scores) and the subset $G_p^i \subset G_p$ of preference edges to item i, i.e., $G_p^i = (U, i, E_p^i = \{(v, i) \in E_p | v \in U\})$. Let A denote the algorithm that takes as input $G_s = (U, E_s)$, an item $i \in I$ and the subgraph G_p^i, and outputs the utilities for all users with respect to i, i.e., $A(G_s, i, G_p^i) = \{\mu_u^i | u \in U\}$. If we can devise an ϵ-differentially private version of A, denoted $\hat{A}$, then it follows from parallel composition (Theorem 3), that calling $\hat{A}(G_s, i, G_p^i)$ for all $i \in I$, in sequence, also satisfies ϵ-differential privacy. Thus, our focus in the following discussion will be on devising algorithm $\hat{A}$. At the end of our discussion, we will present pseudocode for the end-to-end algorithm, as well as a proof of its privacy guarantees.

5.1.1 Strawman Approaches

One naïve approach, which we will refer to as *Noise on Utility* (NOU), to satisfying ϵ-differential privacy for A in-

volves a direct application of the Laplace mechanism (Theorem 1) to the utility values output by A. That is, $\hat{A}(G_s, i, G_p^i) = \{\mu_u^i + Lap(\frac{\Delta_A}{\epsilon}) | u \in U\}$. The sensitivity Δ_A is determined by considering the greatest possible impact that adding or removing any arbitrary preference edge could have on the output of A, i.e., $\Delta_A = max_{u \in U} \sum_{v \in U} sim(v, u)$. Thus, although this approach satisfies ϵ-differential privacy, the sensitivity may be arbitrarily large in practice. For most social similarity measures, the sensitivity will be determined by the user with the highest degree in the social graph and, consequently, the magnitude of the noise introduced into each utility value will greatly exceed the actual value, leading to very poor results. Our experimental results (Section 6) confirm that NOU leads to nearly a complete loss of utility, even for very lenient privacy settings.

A second naïve approach, which we refer to as *Noise on Edges* (NOE), involves injecting independent Laplace noise directly into the weight of each preference edge (i.e., $\hat{w}(u, i) = w(u, i) + Lap(\frac{1}{\epsilon})$) and using the resulting sanitized edges as input to the original algorithm A. Note that non-existent edges are represented as edges with zero weight. The expected error for an arbitrary utility query μ_u^i would be $\frac{\sqrt{2}}{\epsilon} \sum_{v \in sim(u)} sim(u, v)$. Meanwhile, the true value of μ_u^i is *at most* $\sum_{v \in sim(u)} sim(u, v)$ (but often much less, due to the sparsity of real-world preference graphs). Thus for any non-trivial ϵ the error is expected to drown out the true signal, leading to poor recommendation accuracy. We demonstrate the poor accuracy of this approach via empirical results in Section 6.

5.1.2 Proposed Approach

The approach that we propose in this paper is essentially an extension of NOE. The general idea is to group multiple preference edges together, forming disjoint *clusters*, and then to add noise to the *average* edge weight of each cluster. We then use the noisy averages in place of the true weights to derive privacy-preserving estimates for the utility values, which are then used to determine what items to recommend. As we will explain, this allows us to spread a small quantity of noise over multiple edges. More formally, let E_p^i denote the set of preference edges in G_p^i with the addition of zero-weight edges for all $(v, i) \notin G_p^i$. Let $S(E_p^i) = \Phi_S = \{c_1, \ldots, c_n\}$ denote a clustering of E_p^i into disjoint clusters c_i, produced using a *clustering strategy* S [2]. For now, let us assume that S simply clusters the edges randomly, without regard to their weights. Equation (3) formalizes the process of computing the noisy averages, while equation (4) uses them to derive a perturbed estimate $\hat{\mu}_u^i$ for utility query μ_u^i (i.e., the utility of recommending i to u). Note that since we are only considering the preference edges for a *single* arbitrary item i at the moment, one may think of a cluster as containing *users* or *preference edges*, interchangeably—each user is associated with a single preference edge, which in-turn belongs to a single cluster. We will use this fact to simplify the equations and discussions that follow.

$$\bar{c} = Lap\left(\frac{1}{|c|\epsilon}\right) + \sum_{v \in c} \frac{w(v, i)}{|c|} \tag{3}$$

[2]We assume, in the worst case, that the adversary knows S; that is, he knows which edges map to which clusters.

$$\hat{\mu}_u^i = \sum_{c \in \Phi_S} \sum_{v \in sim(u) \cap c} sim(u, v) \cdot \bar{c} \tag{4}$$

Note that the sensitivity of the Laplace noise in equation (3) is $\frac{1}{|c|}$, because adding/removing a single edge e to/from E_p^i could change $\bar{c}$ by at most $\pm\frac{1}{|c|}$, where c is the cluster to which e was assigned by S. To clarify, since S randomly assigns edges to clusters without regard to their edge weights, adding or removing edge e (which, is equivalent to changing its weight to one or zero, respectively) does not change e's cluster assignment; thus, the average edge weight for only one cluster is affected by the addition/deletion. Moreover, since S does not use the sensitive edge weights to determine the clustering, it does not violate privacy.

Computing the utility estimates with equations (3) and (4) introduces two sources of error into each estimate: *perturbation error*, due to the Laplace noise, and *approximation error*, due to averaging. Equation (5) quantifies the total error introduced into an arbitrary utility estimate.

$$\begin{aligned} Err[\hat{\mu}_u^i] &= AE_u^i + \sum_{c \in \Phi_S} \sum_{v \in sim(u) \cap c} sim(u, v) \cdot Err\left[Lap\left(\frac{1}{|c|\epsilon}\right)\right] \\ &= AE_u^i + \sum_{c \in \Phi_S} \frac{\sqrt{2}}{\epsilon |c|} \sum_{v \in sim(u) \cap c} sim(u, v) \end{aligned} \tag{5}$$

The AE_u^i term refers to the approximation error and will be discussed shortly. The right-hand side of (5) characterizes the perturbation error due to the Laplace noise. Observe that as the cluster size increases, the expected perturbation error quickly disappears. Intuitively, the bigger a cluster gets, the less sensitive its average weight is to any one preference edge, allowing a smaller amount of noise to provide the same level of privacy.

The reduction in noise does not come for free, however. As equation (6) shows, the price that we pay comes in the form of approximation error, AE_u^i, due to averaging:

$$\begin{aligned} AE_u^i &= \mu_u^i - \sum_{c \in \Phi_S} \sum_{v \in sim(u) \cap c} sim(u, v) \times \bar{c} \\ &= \sum_{v \in sim(u)} sim(u, v) \times w(v, i) - \\ &\quad \sum_{c \in \Phi_S} \sum_{v \in sim(u) \cap c} sim(u, v) \times \bar{c} \\ &= \sum_{c \in \Phi_S} \sum_{v \in sim(u) \cap c} sim(u, v) \times (w(v, i) - \bar{c}) \end{aligned} \tag{6}$$

From the equation above, it is obvious that simply assigning edges to clusters at random will lead to a significant amount of approximation error; however, by using a more intelligent clustering strategy, we can exchange a large amount of perturbation error for a relatively small amount of approximation error.

Devising a Clustering Strategy. An immediate observation from equation (6) is that approximation error occurs when the weight of a preference edge differs significantly from the average weight of its cluster. Based on this observation, we could attempt to design a clustering strategy that minimizes approximation error by clustering the edges such that those in the same cluster have the same weight. However, unlike the random clustering strategy that we have considered up to now, this strategy requires looking at the

private edge weights to determine the proper cluster assignment for each edge; consequently, additional noise would have to be introduced to ensure that the clustering strategy itself satisfies differential privacy. Intuitively speaking, for such a strategy to work, the noisy average of a cluster would need to be close in value to the individual weights of the edges in the cluster, and at the same time, an adversary (who knows the clustering strategy) should not be able to deduce, with high probability, the true weight of an edge by looking at the noisy average for its cluster, or by knowing the true weights of other edges in the same cluster (i.e., background knowledge). The inherent incompatibility of these goals led us to take a different approach to clustering.

Instead of clustering based on the edge weights, we propose a clustering strategy that uses *only the public social graph* as input, and importantly, does not require the introduction of additional noise. The key observation, based on equation (6), is that low approximation error can be obtained for an arbitrary utility estimate $\hat{\mu}_u^i$, without the need to look at the sensitive edge weights, by clustering such that the edges belonging to the users that are most similar to u are assigned to the same cluster and constitute a large fraction of the edges in that cluster[3]. The following simple example illustrates. Suppose that there is a cluster $c = \{(v,i)|v \in sim(u)\}$ and assume for simplicity that $sim(u,v) = x, \forall v \in sim(u)$; then, rewriting equation (6), we can see that the approximation error cancels out:

$$\begin{aligned} AE_u^i &= \sum_{c \in \Phi_S} \sum_{v \in sim(u) \cap c} sim(u,v) \times (w(v,i) - \bar{c}) \\ &= \sum_{v \in sim(u)} x \times (w(v,i) - \bar{c}) \\ &= x \times \left(\sum_{v \in sim(u)} w(v,i) - \sum_{v \in sim(u)} \bar{c} \right) \\ &= 0. \end{aligned} \tag{7}$$

Of course, in real social graphs, any given user will usually belong to many different similarity sets, so it will not be possible to produce clusters that are both disjoint and that perfectly correspond to the similarity sets of all users, as the example above assumes. However, the observation above suggests that approximation error can be significantly reduced by building clusters around groups of edges belonging to users that have *high mutual similarity to many of the same users*.

Remark. Before proceeding with our discussion, we point out that since we are now talking about clustering based only on the similarity sets—without any regard to the preference edge weights—we can view this conceptually as producing a single clustering of the *users*, from which the cluster assignments for the preference edges, with respect to any item j, can be derived. That is, regardless of which item we are computing utility estimates for, the cluster structure is the same. To clarify further, this implies that two preference edges (u,i) and (v,i) are in the same cluster if and only if edges (u,j) and (v,j) are in the same cluster (for any $u,v \in U$ and $i,j \in I$).

Clustering via Community Structure. For the social similarity measures discussed in Section 2.2 (and other natural choices), we make the intuitive observation that groups of users with high mutual similarity tend to be highly interconnected via many short paths in the social graph. In the graph analysis literature, such regions of highly interconnected nodes are called *communities*, and they are an inherent and well-studied characteristic of real social graphs [10]. In light of this connection, we investigated *community detection* as a clustering strategy for reducing approximation error.

For this work, we adopted the popular *Louvain method* [4], due to its demonstrated ability to quickly and accurately detect communities in very large graphs [10]. Louvain is a fast, heuristic approach that involves the greedy maximization of *modularity* to detect communities. Modularity is a measure of the density of edges within clusters relative to that of the same clusters in a hypothetical null model—a graph with the same nodes and degree distribution but with the edges randomly re-wired. The modularity of a clustering Φ, denoted $Q(\Phi)$, is defined as

$$Q(\Phi) = \sum_{c \in \Phi} \frac{|E_c|}{2|E_s|} - \left(\frac{\sum_{u \in c} deg(u)}{2|E_s|} \right)^2 \tag{8}$$

where E_c denotes the set of edges in cluster c, and $deg(u)$ is the degree of user u. Louvain begins by placing each user node in its own cluster. It then considers each node of the graph in sequence, moving it into one of its neighbors' clusters, if doing so would lead to a gain in modularity. This process is repeated until convergence, at which time the algorithm contracts each cluster into a single super-node and repeats the process on the new graph. These two steps are repeated until the modularity is maximized. We additionally incorporate a multi-level refinement step (see [29]) that helps to make the clustering output more stable under different initial node orderings. Louvain has a running time that is linear in the number of edges, allowing it to scale to graphs with billions of edges [10].

Remark. One may question why we did not choose to cluster the users by applying a matrix clustering algorithm, such as *K-Means*, directly to the user similarity matrix. We point out that, unlike community detection, matrix clustering algorithms typically require the number of clusters to be specified *a priori*, which can be difficult to determine in practice[4]. Moreover, scalability would likely be an issue for such approaches, given the enormous size of modern social graphs.

5.2 Algorithm and Analysis

Algorithm 1 presents the end-to-end algorithm for our framework. The algorithm takes as input the two graphs, the number of recommendations N, and the privacy parameter, ϵ. It returns the set R containing, for each user $u \in U$, a personalized list R_u of the top N item recommendations. In lines 1–7, the set of users is clustered into disjoint clusters and noisy averages of the edge weights are computed for each cluster, item pair, i.e., equation (3). Lines 8–17 compute the noisy utility values for every user-item pair, following equation (4). Finally, in lines 18–20, a personal-

[3]Recall that computing the similarity of two users, $sim(u,v)$ uses only the public social graph; thus, no privacy is leaked by this clustering strategy.

[4]Note that we cannot simply try different values of k and pick the one that yields the most accurate utility estimates with equation (4), as comparing the estimates against the true utilities would violate differential privacy.

Algorithm 1 Privacy-Preserving Social Recommender

Input: G_s, G_p, N, ϵ
Output: The set R containing a recommendation lists R_u for each user with the top N items ranked by utility.

1: $\Phi \leftarrow$ createClusters(G_s)
2: **for** item i in I **do**
3: **for** cluster c in Φ **do**
4: $w_c^i \leftarrow 0$
5: **for** user u in cluster c **do**
6: $w_c^i \leftarrow w_c^i + \mathrm{w}(u, i)$
7: $\hat{w}_c^i \leftarrow \frac{w_c^i}{\mathrm{size}(c)} + \mathrm{Lap}\left(\frac{1}{\mathrm{size}(c)\epsilon}\right)$
8: **for** user u in U **do**
9: Initialize $R_u \leftarrow \oslash$
10: **for** item i in I **do**
11: $\hat{\mu}_u^i \leftarrow 0$
12: **for** cluster c in Φ **do**
13: Initialize $sim\_sum \leftarrow 0$
14: **for** user v in $c \cap sim(u)$ **do**
15: $sim\_sum \leftarrow sim\_sum + \mathrm{sim}(u, v)$
16: $\hat{\mu}_u^i \leftarrow \hat{\mu}_u^i + sim\_sum \times \hat{w}_c^i$
17: Append the tuple $(i, \hat{\mu}_u^i)$ to the list R_u
18: Sort R_u in descending order of utility
19: Truncate R_u to the top N items
20: Add R_u to R
21: **return** R

ized recommendation list is produced for each user from the noisy item utilities. We reiterate that *createClusters*(G_s) technically clusters the *users* (not the *preference edges*) into disjoint clusters; thus, $size(c)$ in the algorithm refers to the number of *users* in cluster c. The same clustering of the users is used for computing the noisy utility estimates for all items (i.e., for every iteration of the loop beginning on line 2).

The framework can be easily customized by plugging in different social similarity measures for $sim(u, v)$ in line 15 or alternate different clustering algorithms on line 1. As long as both the similarity measure and the clustering algorithm operate solely on the social graph, G_s, then ϵ-differential privacy is satisfied, as we will prove next.

5.2.1 Privacy Analysis

THEOREM 4. *Algorithm 1 satisfies ϵ-differential privacy for preference edges.*

PROOF. Consider a social graph $G_s = (U, E_s)$ and two arbitrary preference graphs, $G_p = (U, I, E_p)$ and $G_p' = (U, I, E_p')$, such that E_p' is obtained from E_p by adding or removing a single preference edge. We can view Algorithm 1 as consisting of three modules:

1. createClusters(G_s), on line 1, clusters the user nodes into a set of disjoint clusters, Φ, based on the social graph.

2. $A_w(\Phi, \epsilon, G_p)$ is the module consisting of lines 2–7, which takes the clustering Φ and the preference graph G_p, and outputs $\hat{w} = \{\hat{w}_c^i | i \in I; c \in \Phi\}$, containing the noisy averages for each cluster-item pair.

3. $A_R(G_s, \Phi, N, \hat{w})$, consisting of lines 8–21, uses the noisy averages $\hat{w}$ output by A_w to compute the utility estimates for every user-item pair and outputs R containing the top N items R_u for every user $u \in U$.

Observe that only module A_w makes use of the private preference data. The clusters output by *createClusters* are derived solely from the public social graph, and module A_R uses only public data and the sanitized output, $\hat{w}$, from A_w. Therefore, to show that Algorithm 1 is ϵ-differentially private, it suffices to show that module A_w satisfies definition 5; that is, we need to show that for any set of possible outputs $O \subseteq Range(A_w)$,

$$Pr[A_w(\Phi, \epsilon, G_p) \in O)] \leq \exp(\epsilon) Pr[A_w(\Phi, \epsilon, G_p') \in O)].$$

Observe that the outer loop of module A_w (beginning on line 2) is run one time for every item. Let us consider an arbitrary iteration j (that is, the iteration for item j). Ignoring the Laplace noise added on line 7, for the moment, we can view the output of iteration j as the vector $w^j = \langle w_c^j | c \in \Phi \rangle$. Observe that adding or removing a single arbitrary preference edge (v, j) to/from G_p could change w_c^j by at most $\pm 1/|c|$, where c is the cluster containing user v.[5] We can view the process of computing the noisy averages w^j as performing a sequence of disjoint computations, since each average uses a different subset of the preference edges (e.g., w_c^j is computed only from the subset of edges $\{(v, j) | v \in c\} \subset G_p$). In line 7, we add independent Laplace noise from $Lap(\frac{1}{\epsilon|c|})$ to each of the averages, which satisfies ϵ-differential privacy for each, by Theorem 1. By Theorem 3, computing the sequence of averages in w^j satisfies ϵ-differential privacy, and therefore iteration j as a whole satisfies ϵ-differential privacy. Finally, since each iteration of A_w operates on a disjoint set of the preference edges (i.e., those attached to a single item), the parallel composition property (Theorem 3) applies again, and module A_w as a whole satisfies ϵ-differential privacy. Therefore, we have

$$Pr[A_w(\Phi, \epsilon, G_p) = \hat{w})] \leq \exp(\epsilon) Pr[A_w(\Phi, \epsilon, G_p') = \hat{w})].$$

which holds for every $\hat{w} \in Range(A_w)$; thus, we have

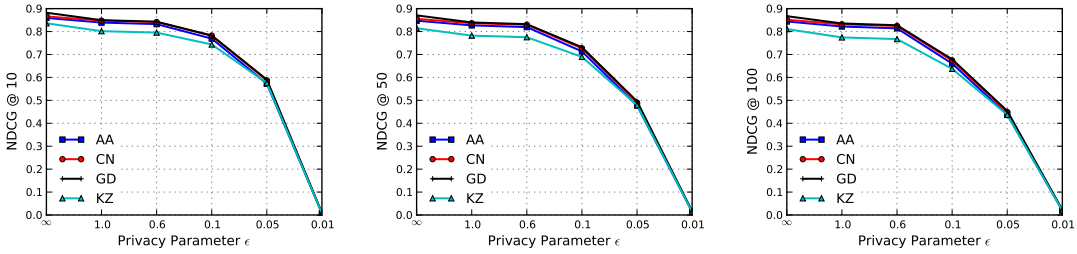
$$\begin{aligned} Pr[A_w(\Phi, \epsilon, G_p) \in O] &= \sum_{\hat{w} \in O} Pr[A_w(\Phi, \epsilon, G_p) = \hat{w})] \\ &\leq \sum_{\hat{w} \in O} \exp(\epsilon) Pr[A_w(\Phi, \epsilon, G_p') = \hat{w})] \\ &= \exp(\epsilon) Pr[A_w(\Phi, \epsilon, G_p) \in O], \end{aligned}$$

as desired. This completes the proof. □

6. EXPERIMENTAL ANALYSIS

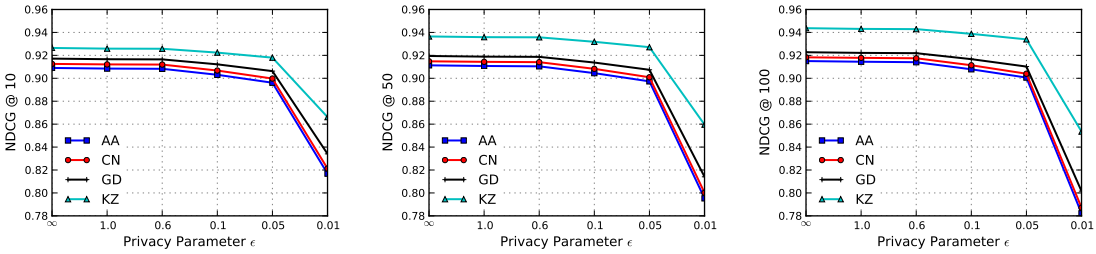
Due to the data-dependent nature of our proposed approach, we conducted a series of experiments designed to evaluate the trade-off between privacy and accuracy for different instantiations of our framework on two real-world data sets. In particular, we sought to validate our hypothesis that clustering according to the community structure of the social graph would produce clusters that effectively balance approximation error and perturbation error. We also sought to determine how the number of recommendations, N, and the privacy setting, ϵ, impacts the accuracy of the recommendations. Additionally, we compared our approach against the two naïve baseline approaches, NOU and NOE (Section 5.1.1), and against adaptations of two recent differential privacy approaches (GS [17] and LRM [34]) (Section 4).

[5] Recall that the clusters contain *users*, so adding or removing a preference edge does not change $|c|$.



(a) Avg. *NDCG@10* (b) Avg. *NDCG@50* (c) Avg. *NDCG@100*

Figure 1: (Last.fm) Comparison of average *NDCG@N* of each private social recommender for different settings of ϵ and N on Last.fm.



(a) *Avg. NDCG@10* (b) *Avg. NDCG@50* (c) *Avg. NDCG@100*

Figure 2: (Flixster) Comparison of average *NDCG@N* of each private social recommender for different settings of ϵ and N on Flixster. (Note the different y-axis values used in this figure compared to Fig. 1.)

6.1 Datasets

We used two publicly available[6] data sets that were crawled from two online, entertainment-oriented communities: (1) Last.fm, a social music service, and (2) Flixster.com, a social movie rating web site.

Last.fm data set [5]. This relatively small data set contains undirected social edges between user nodes that indicate mutual friendship, and directed edges from user nodes to artist nodes expressing *listened-to* relations where each edge is weighted by the number of times the user listened to a song by the artist. The social graph consists of one main connected component containing 97.4% of the user nodes and 19 small connected components with 2 to 7 nodes each. We constructed a preference graph G_p by discarding 636 listened-to edges ($< 0.07\%$ of the edges in G_p) with weight < 2, and assigning a weight of 1 to the remaining edges.[7] Table 1 summarizes the pre-processed data set.

Flixster data set [14]. This much larger data set contained more than 700K users and almost 49K movies. We chose to use the main connected component of the social graph induced by the set of users with at least one preference edge (movie rating). The resulting data set contained around 137K users and almost 49K items (movies). Each preference edge (u, i) had a weight in the range $[0.5, 5]$, indicating the rating given to movie i by user u. To get rid of ratings that were likely to indicate dislike, we discarded edges having weight < 2 (6.6% of $|E_p|$) and assigned a weight of 1 to the remaining edges.

[6] *Last.fm*: `http://ir.ii.uam.es/hetrec2011/datasets.html`; *Flixster*: `http://www.sfu.ca/~sja25/datasets`

[7] Listening to an artist only once is unlikely to indicate a positive preference.

Table 1: Summary of data sets.

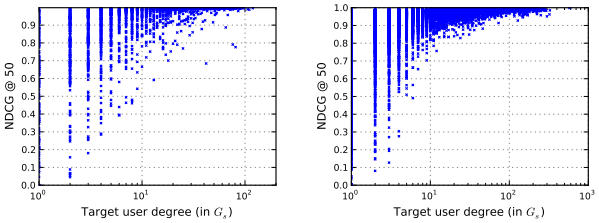
	Last.fm	Flixster
$\|U\|$	1,892	137,372
$\|E_s\|$	12,717	1,269,076
avg. user degree	13.4 (std. 17.3)	18.5 (std. 31.1)
$\|I\|$	17,632	48,756
$\|E_p\|$	92,198	7,527,931
avg. item degree	48.7 (std. 6.9)	54.8 (std. 218.2)
$sparsity(G_p)$	0.997	0.999

6.2 Experimental Setup

We instantiated our framework with each of the four social similarity measures introduced in Section 2.2, yielding four private social recommenders, denoted AA, CN, GD[8] and KZ[9], respectively. We used *NDCG* (Section 2.4) as our accuracy metric. The *NDCG* values reported for a given data set are averages over the users in the data set. Experiments were repeated 10 times. For the Flixster data set, we generated recommendations for a random subset of 10,000 users; however, we emphasize that we used all 137,372 users in the clustering phase and for computing the recommendations of the randomly selected users. To cluster the social graphs, we ran the Louvain algorithm 10 times (with different random node orderings each time) and selected the clustering with the highest modularity. For Last.fm, 35 clusters were produced. The main connected component of the graph was divided into 16 clusters containing an average of 115 (std. 164)

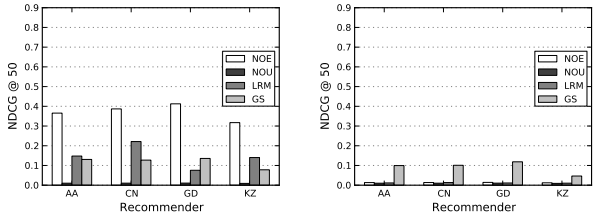
[8] For GD, we limited the maximum distance $d = 2$.

[9] For KZ, we limited the maximum path length $k = 3$ and fixed the damping factor $\alpha = 0.05$.



(a) Last.fm (b) Flixster

Figure 3: Relationship between user degree (log scale) and *NDCG@50* due to approximation error (i.e., $\epsilon = \infty$), observed for CN similarity measure on Last.fm (a) and Flixster (b). A similar relationship was seen for the other similarity measures (omitted).



(a) $\epsilon = 1.0$ (b) $\epsilon = 0.1$

Figure 4: (a) and (b) show *NDCG@50* of the naïve NOU and NOE baselines, as well as adaptations of the recent LRM [34] and GS [17] approaches, on Last.fm with $\epsilon \in \{1.0, 0.1\}$.

users each, while the remaining 19 clusters corresponded directly to the 19 small components of the social graph. The largest cluster contained 28.5% of the users. For Flixster, 46 clusters were produced with an average size of 2,986 users (std. 6,399 users). The largest cluster contained 18.3% of the users. We used the Louvain implementation with multi-level refinement from Pajek v3.07[10]. All experiments were performed on a single machine with an Intel Core i7 (2.8Ghz) and 12G of RAM.

6.3 Analysis of Privacy-Accuracy Trade-off

We evaluated our framework using a variety of privacy settings ($\epsilon \in \{\infty, 1.0, 0.6, 0.1, 0.05, 0.01\}$) and numbers of recommendations ($N \in \{10, 50, 100\}$) to investigate their effect on the recommendation accuracy (*NDCG@N*). The results for Last.fm and Flixster are shown in Figs. 1 and 2, respectively. Recall that a smaller ϵ leads to more noise but stronger privacy. We included $\epsilon = \infty$ (i.e., no noise) to see how much of the accuracy loss was due to the approximation error alone. For perspective we also include results for the two naïve baseline approaches, NOU and NOE (from Section 5.1.1), on Last.fm with $\epsilon \in \{0.1, 1.0\}$ and $N = 50$ (Figs. 4(a) and 4(b)). The baseline results for Flixster were comparable and thus omitted for space.

Effect of Approximation Error. We start by analyzing the impact of the approximation error alone (i.e., $\epsilon = \infty$). We will focus our discussion on NDCG@50 (Figs. 1(b) and 2(b)), as similar observations can be made for the other values of N. Looking at the left-most points ($\epsilon = \infty$) of each line in Fig. 1(b), we can see that for Last.fm, the approximation error accounted for a loss of between 0.13 and 0.19 in average accuracy, depending on the recommender. On Flixster (Fig. 2(b)), the accuracy loss due to approximation error was < 0.1 for all four recommenders. A closer look at the results from the user-level also revealed a strong relationship between user degree (in the social graph) and accuracy. On average, users with more neighbors in the social graph were less affected by approximation error than those with few neighbors. For example, Figs. 3(a) and 3(b) illustrate this effect, as observed for the CN similarity measure on Last.fm and Flixster, respectively. Each point on the scatter plots represents the NDCG@50 for one user (averaged over the 10 runs) as a function of the user's degree (log scale). For users with degree > 10 in the Last.fm social graph, the average NDCG@50 was 0.969, while for users with degree ≤ 10 the average was considerably lower at 0.809. For the Flixster social graph, users with degree > 10 had an average NDCG@50 of 0.975, while for users with degree ≤ 10 the average was 0.871. The explanation for this effect is that low degree users tend to have much smaller similarity sets that make up only a small fraction of the clusters that contain them; as a result, their utility estimates are heavily affected by the other non-similar users in the clusters. However, not all low-degree users experienced excessive amounts of approximation error; we observed several common exceptions. Some low-degree users with high-degree neighbors were able to accumulate large enough similarity sets. Other low degree users happened to have similarity sets that, despite being small, were attached to universally popular items, which are inherently more resistant to approximation error.

Effect of Privacy Parameter (ϵ). Let us now look at the results when noise is added into the mix to satisfy differential privacy. We will again focus our discussion on NDCG@50 (Figs. 1(b) and 2(b)). For Last.fm, the noise introduced by privacy settings $\epsilon = 1$ and $\epsilon = 0.6$ had very little effect on the average NDCG@50 compared to $\epsilon = \infty$. As ϵ became closer to 0.1, the Laplace noise had a more significant impact, reducing accuracy to between 0.7 and 0.73, depending on the recommender. Privacy settings below 0.1 led to poor accuracy in general. Comparing these results to that of the baseline approaches in Fig. 4, the benefits of our user clustering approach are substantial. Both baselines performed very poorly, even under weak privacy settings (Fig. 4(a)). For NOU in particular, the recommendations were essentially no better than random guessing. NOE performed much better than NOU under low noise ($\epsilon = 1.0$), but its accuracy was equally as poor as NOE with $\epsilon = 0.1$. Note that the results for LRM and GS, which are also included in Fig. 4, will be discussed below, in Section 6.4.

On the Flixster data set (Fig. 2), all four instantiations of our framework were remarkably resistant to the Laplace noise, even under the strongest privacy settings. For $\epsilon \geq 0.05$, the noise had little to no impact on average accuracy. With $\epsilon < 0.05$, the accuracy began to fall sharply; however, even with ϵ as low as 0.01 (which is considered to be a very strong privacy guarantee), the accuracy was ≥ 0.79 on average, for all four recommenders. The higher accuracy observed for the Flixster data set, compared with Last.fm, appears to be due primarily to the larger average user degree in the Flixster social graph (see Table 1), which led to much larger sets of mutually similar users on average, and

[10] `http://pajek.imfm.si`

in-turn much larger clusters. This enabled a more significant reduction in perturbation error without significantly increasing approximation error.

Effect of the Number of Recommendations (N). For Last.fm, we observed that the average *NDCG* generally decreased as N increased from 10 to 100 (comparing Figs. 1(a) to 1(c)), though the effect was most evident for small values of ϵ. This effect can be explained by the fact that the items that are farther down in the ranking have a lower utility and are thus more affected by noise; as N gets larger, there is a greater opportunity for such items to be displaced by the many zero-utility items. On Flixster, increasing N up to 100 only had a negative effect for the strongest privacy setting $\epsilon = 0.01$. For larger values of ϵ, increasing the number of recommendations actually had a positive impact. This appears to be a product of the higher user degree and the higher average item degree in the Flixster data set, which led to a greater number of high-utility items.

6.4 Comparison with Other Approaches

In this set of experiments, we compared our approach with two recently proposed differential privacy approaches: the *low-rank mechanism* (LRM) of [34], and an adaptation of the *group and smooth* (GS) approach of [17]. Although neither of these approaches was designed specifically for the private social recommendation task, they appear to be the closest competitors to our approach in the current literature. We first briefly explain how we adapted these approaches to our setting, and then present the results.

Low-rank Mechanism. LRM can be applied quite naturally to our task, as follows. Let W denote a $|U| \times |U|$ workload matrix of utility queries, where $W_{u,v} = sim(u, v)$. Let D_i be a length $|U|$ column vector containing a 1 at position v when $(v, i) \in G_p$, or a zero otherwise. LRM decomposes W into matrices B, of size $|U| \times r$, and L of size $r \times |U|$, such that $r < |U|$ and $W \approx BL$. It then adds Laplace noise to LD_i, yielding $\widehat{LD_i}$. Noisy answers to the utility queries for item i are then obtained via the product $B\widehat{LD_i}$. The key is that LD_i has lower sensitivity than WD_i, requiring less noise. We used the Matlab LRM implementation published with [34]. LRM takes two parameters: (1) r determines the rank of BL and it suggested in [34] to choose r between *rank(W)* and 1.2· *rank(W)*; we used *rank(W)* in our experiments. (2) γ determines the error tolerance for the decomposition of W; we used $\gamma = 10$ ($\gamma = 1$ for KZ, since it produces significantly smaller utility values), as suggested in [34].

Group and Smooth. Although the setting and task in [17] is different from ours, the idea behind the approach can be extended to our problem. GS can be thought of as extending NOU similarly to the way our approach extends NOE; however, GS relies on uniformity to reduce approximation error. The idea is to compute answers to all of the utility queries (non-privately), and then to arrange them in groups of size m, such that the values in each group are approximately uniform (using a separate, differentially private procedure discussed below); the values in each group can then be approximated by the noisy group average. The noise added to each group average is $Lap(\frac{\Delta}{\epsilon/2}) = Lap(\frac{2\Delta}{\epsilon})$, where $\Delta = \frac{1}{m} \times max_{u \in U} \sum_{v \in U} sim(v, u)$. The challenge is how to group the query answers so that they are approximately uniform, without leaking privacy. [17] proposes a sampling approach that amounts to computing rough estimates $\hat{\mu}_u^i$ for the query answers by ensuring that each preference edge (v, i) is used in at most one query estimate, selected at random from $\{\hat{\mu}_u^i | u \in sim(v)\}$; then noise is added to to each utility estimate $\hat{\mu}_u^i$ with $\frac{\epsilon}{2}$ and $\Delta = \max_{v \in sim(u)} sim(u, v)$. Finally, the true answers to the utility queries are sorted, using the rough estimates as the sorting keys, and then grouped consecutively in groups of size m. In [17], the m that gave the lowest error relative to the rough utility estimates (not the true utilities) was selected; however, for simplicity we used the m that gave the best NDCG relative to the true utilities.[11] The reader is referred to [17] for more details on GS and proof that it satisfies ϵ-differential privacy.

Results. Fig. 4 shows the NDCG@50 of LRM and GS on Last.fm, along side the results for NOE and NOU. Although both significantly outperformed the NOU baseline, both approaches were outperformed by the NOE baseline. The poor performance of GS appeared to be due to the ineffectiveness of the sampling procedure on the highly sparse preference data, which led to a poor groupings with non-uniform values. For LRM, we observed that all of the workload matrices had high rank—the average rank was 1808, which is close to the maximum rank of 1892 for workloads of this size—and LRM is known to perform poorly on workloads with high rank [34]. We also note that the number of utility queries, even for this small data set, was larger than the largest workload considered in [34] (1892 queries vs. 1024). Our results are consistent with the findings in [34], where LRM was shown to perform worse as the number of queries increases. Given that the number of queries in our task is proportional to the number of users, we would expect the results to be even worse in practice, where the number of users is often in the millions. The poor results of LRM and GS illustrate the futility of NOU-based approaches for the social recommendation task.

7. CONCLUSIONS AND FUTURE WORK

Our results on the Last.fm and Flixster data sets demonstrate the ability of the proposed framework to achieve strong differential privacy guarantees without significantly degrading the utility of the recommendations produced. These findings confirm our hypothesis that clustering according to the natural community structure of the social graph leads to clusters that strike an effective balance between approximation error and perturbation error for several natural social similarity measures. There are a number of avenues for extending this work. A main focus for future work will be extending our framework to provide differential privacy guarantees when recommendations are made over dynamic graphs. We also plan to explore improvements to the clustering strategy, including (1) optimizing it more for the specific similarity measure being used, and (2) investigating post-processing heuristics to clean up the clustering by, for example, pruning low-quality clusters. We also plan to extend our framework to handle weighted preference edges (e.g., ratings) and evaluate the impact of different weighting schemes. Finally, we would like to evaluate the framework for a larger variety of social similarity measures and on additional data sets as they become available.

[11] This technically violates DP and gives GS an unfair advantage that it would not have in practice.

Acknowledgment

The authors would like to extend their sincere thanks to Ashwin Machanavajjhala for helpful discussions and to all of the anonymous reviewers, whose insightful comments and suggestions helped to greatly improve the quality of this paper. This work is supported in part by the National Science Foundation under the awards CNS-0747247 and CNS-1314229, and by an NSA Science of Security Lablet grant at North Carolina State University.

8. REFERENCES

[1] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *ICDM*, pages 1–10. IEEE, 2012.

[2] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *ACM SIGKDD*, pages 503–512. ACM, 2010.

[3] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW*, pages 551–560. ACM, 2009.

[4] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.*, 2008(10):P10008, 2008.

[5] I. Cantador, P. Brusilovsky, and T. Kuflik. Hetrec. In *RecSys*, New York, NY, USA, 2011. ACM.

[6] G. Danezis, T. Aura, S. Chen, and E. Kıcıman. How to share your favourite search results while preserving privacy and quality. In *PETS*, pages 273–290. Springer, 2010.

[7] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Improving recommendation quality by merging collaborative filtering and social relationships. In *ISDA*, pages 587–592. IEEE, 2011.

[8] C. Dwork. Differential privacy. *ICALP*, pages 1–12, 2006.

[9] C. Dwork. Differential privacy: A survey of results. *TAMC*, pages 1–19, 2008.

[10] S. Fortunato. Community detection in graphs. *Phys Reps*, 486(3):75–174, 2010.

[11] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[12] T. Hoens, M. Blanton, and N. Chawla. A private and reliable recommendation system for social networks. In *SocialCom*, pages 816–825. IEEE, 2010.

[13] Z. Huang, X. Li, and H. Chen. Link prediction approach to collaborative filtering. In *JCDL*, pages 141–142, 2005.

[14] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142. ACM, 2010.

[15] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *TOIS*, 20(4):422–446, 2002.

[16] C. Kaleli and H. Polat. Providing private recommendations on personal social networks. *AWIC*, pages 117–125, 2010.

[17] G. Kellaris and S. Papadopoulos. Practical differential privacy via grouping and smoothing. In *PVLDB*, volume 6, 2013.

[18] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *SIGIR*, July 2009.

[19] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134. ACM, 2010.

[20] D. Liben-Nowell and J. M. Kleinberg. The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031, 2007.

[21] F. Liu and H. J. Lee. Use of social network information to enhance collaborative filtering performance. *Expert Syst. Appl*, 37(7):4772–4778, 2010.

[22] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):1150–1170, 2011.

[23] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *TKDD*, 1(1):3, 2007.

[24] A. Machanavajjhala, A. Korolova, and A. Sarma. Personalized social recommendations-accurate or private? *PVLDB*, 4(7), 2011.

[25] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30. ACM, 2009.

[26] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *SIGKDD*, pages 627–636. ACM, 2009.

[27] M. E. Newman. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. U.S.A.*, 98(2):404–409, 2001.

[28] S. Purushotham, Y. Liu, and C. Kuo. Collaborative topic regression with social matrix factorization for recommendation systems. In *ICML*, 2012.

[29] R. Rotta and A. Noack. Multilevel local search algorithms for modularity clustering. *JEA*, 16, 2011.

[30] L. Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzz.*, 10(5):557–570, 2002.

[31] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *TKDE*, 23(8):1200–1214, 2011.

[32] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. *SDM*, pages 150–168, 2010.

[33] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In *ICDE*, pages 32–43. IEEE, 2012.

[34] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. In *PVLDB*, volume 5, pages 1352–1363, 2012.

[35] Q. Yuan, S. Zhao, L. Chen, Y. Liu, S. Ding, X. Zhang, and W. Zheng. Augmenting collaborative recommender by fusing explicit social relationships. In *RSWEB, RecSys*, 2009.

[36] C. Zeng, J. F. Naughton, and J.-Y. Cai. On differentially private frequent itemset mining. *PVLDB*, 6(1):25–36, 2012.