SlopPy: Slope One with Privacy

Sébastien Gambs and Julien Lolive

Université de Rennes 1 - INRIA / IRISA,
Avenue du Général Leclerc
35042 Rennes, France
sgambs@irisa.fr, julien.lolive@inria.fr

Abstract. In order to contribute to solve the personalization/privacy paradox, we propose a privacy-preserving architecture for one of state-of-the-art recommendation algorithm, Slope One. More precisely, we describe SlopPy (for *Slope One with Privacy*), a privacy-preserving version of Slope One in which a user never releases directly his personal information (*i.e*, his ratings). Rather, each user first perturbs locally his information by applying a Randomized Response Technique before sending this perturbed data to a semi-trusted entity responsible for storing it. While there is a trade-off to set between the desired privacy level and the utility of the resulting recommendation, our preliminary experiments clearly demonstrate that SlopPy is able to provide a high level of privacy at the cost of a small decrease of utility.

Keywords: Privacy, Recommender Systems, Collaborative Filtering, Randomized Response Technique.

1 Introduction

The advent of personalization is strongly tied to the development and rapid growth of Electronic Commerce during the last decade. Indeed, major Internet companies provide services that are tailored to the interests of their users, which in turn as lead to the collection of large amount of personal data and the construction of detailed profiles of these users. For instance, Google personalizes the news pushed towards a specific user according to the topics of the news that he had consulted in the past [4]. Another approach consists in using a *recommender system* that creates a user's profile and compare it to the profiles of other users in order to select in accordance with the tastes of similar users, the most relevant items (*e.g.*, advertisements, movies, ...) for this particular user. For example, Amazon relies on a recommendation engine using an *item-based collaborative filtering* to propose suggestions to users about books that they might like [10].

The massive gathering of personal information generated by the development of personalized services is clearly at odds with the privacy rights of the users of these systems. More precisely, the two fundamental questions summarizing the interplay between personalization and privacy are the following [12]:

> *"... (1) to what extent users have to disclose personal information in order to enjoy personalized and context-aware services in a user-controlled,*

R. Di Pietro et al. (Eds.): DPM 2012 and SETOP 2012, LNCS 7731, pp. 104–117, 2013.

privacy-preserving and trusted way, as well as (2) to find a reasonable balance between user-centric requirements and natural business interests of service providers and authorities, who offer and regulate such services."

While we recognize that the second issue is also fundamental, in this paper we focus mainly on addressing the first one. In particular, we believe that in order to protect the privacy of their users and limit the risks of privacy breaches, the recommender systems must integrate the privacy issues directly into the conception of their architecture, following the *privacy-by-design* paradigm. Indeed, while personalization and privacy may seem to be antagonist at first glance, privacy-preserving personalized services and architectures have been developed in the past few years that aim at reconciling these seemingly conflicting goals, in particular in the context of personalized target advertising systems [17,7].

In this paper, we present our approach towards this direction by proposing a privacy-preserving architecture for one of state-of-the-art recommendation algorithm, Slope One [9]. More precisely, we describe SlopPy (for *Slope One with Privacy*), a privacy-preserving version of Slope One in which a user *never* releases directly his personal information (*i.e*, his ratings) to a trusted third party. Rather, each user first perturbs locally his information by applying a *Randomized Response Technique* before sending this perturbed data to a semi-trusted entity responsible for storing it. Out of the perturbed ratings, the semi-trusted entity construct two matrices (*i.e.*, the deviation matrix and the cardinality matrix) following the standard Slope One algorithm. When a user needs a recommendation on a particular item (*i.e.*, a movie), he fetches particular information from these matrices through a *private information retrieval* scheme [13] hiding the content of his query (*i.e.*, the item he is interested in) to the semi-trusted entity. By combining the data retrieved with his true ratings (which are only stored locally on his machine), the user can then locally compute the output of the recommendation algorithm for this particular item. While there is often a trade-off to set between the desired privacy level (quantified in terms of the magnitude of the noise added) and the utility of the resulting recommendation (measured by the Mean Absolute Error and the Root Mean Squared Error), our experiments clearly demonstrate that SlopPy is able to provide a high level of privacy at the cost of a small decrease of utility.

The outline of the paper is the following. First, in Section 2, we give an overview of the background on the Slope One recommendation algorithm as well as local perturbation approaches such as Randomized Response Techniques. then in Section 3, we briefly review privacy-preserving versions of Slope One that have been developed in recent years. Afterwards, in Section 4, we describe SlopPy, which is both a privacy-preserving version of Slope One and a recommendation architecture built around this algorithm. Finally, in Section 5, we report on experimentations performed with SlopPy on the classical Movielens dataset before concluding with a discussion and some future work in Section 6.

2 Background

Slope One recommendation algorithm. While several approaches exist to recommendation such as collaborative filtering, item-based recommendation and content-based recommendation, thereafter we focus on Slope One [9], which is a collaborative filtering algorithm (*i.e.*, the recommendation is based on the tastes of similar users) designed by Lemire and Maclachlan.

Let us first fix the notation. For the rest of this paper, let the variable u refers to a specific user while i and j will be used as the indexes of particular items (such as movies). Let also r denotes a true rating that a user has really given to an item (*e.g.*, a movie he has seen), while $\hat{r}$ denotes a predicted rating, which is the effective prediction made by the recommendation algorithm (in our case Slope One or SlopPy) on an item that a user has never rated. For instance, $r_{u,i}$ corresponds to the rating given by user u on the item i while $\hat{r}_{u,j}$ is the prediction of the rating that user u will give to item j according to the recommendation algorithm. In general, the rating given to an item will be either an integer or a real value drawn from a finite range. For example, in the case of the Movielens dataset, the interval considered is $[1, 5]$ in which a rating of 5 is an excellent rating while a value of 1 corresponds to the worst possible one. Finally, the variable n denotes the total number of items in the domain considered (*e.g.*, the number of possible movies).

To predict the rating of a particular item i for a specific user u, the Slope One algorithm combines the information about all the items that u has rated with the information about the ratings of all users that have also rated i. First, Slope One constructs a *cardinality matrix* of size n by n containing the value of $\phi_{i,j}$, which corresponds to the number of users that have rated both items i and j. This can be done easily in a non-private version of the algorithm by having all users sending all their ratings to a central entity. Afterwards, Slope one computes a *deviation matrix* out of the ratings provided by users. More precisely, the (standard) deviation matrix of size n by n is constructed by computing the standard deviation $\overline{\delta_{i,j}}$ between each pair of items i and j by considering the subset of users that have rated *both* items i and j:

$$\overline{\delta_{i,j}} = \frac{\sum_u (r_{u,i} - r_{u,j})}{\phi_{i,j}} \tag{1}$$

in which $r_{u,i}$ and $r_{u,j}$ are respectively the ratings of user u for the item i and j, while $\phi_{i,j}$ is the number of users that have rated both items i and j. Note that the construction of the cardinality and deviation matrices can be done in a time directly proportional to their sizes, which is $O(n^2)$ for n the number of items in the domain.

The standard (*unweighted*) version of Slope One relies only on the deviation matrix to perform a recommendation. More precisely, the following equation summarizes the formula used by unweighted Slope One to predict the rating (the sum is performed over the subset of items that have been rated by user u):

$$\mathsf{UnweightedSlopeOne}(u, j) = \frac{\sum_{i|i \neq j} (\overline{\delta_{i,j}} + r_{u,i})}{n_u} \tag{2}$$

in which j is the item on which the rating will be predicted for the user u, $\overline{\delta_{i,j}}$ is the standard deviation between the pair of items i and j, $r_{u,i}$ is the rating given by user u to item i and n_u is the number of items that have been rated by user u. The *Weighted Slope One* is another variant of the Slope One algorithm also due to Lemire and Maclachlan [9]. In a nutshell, compare to the standard (*i.e.*, unweighted) version, the main difference is that the weighted version of Slope One considers that the pairs of items that are the most relevant for the prediction are the ones that have been rated by a large number of users. This is reflected by the following formula for the prediction:

$$\mathsf{WeightedSlopeOne}(u,j) = \frac{\sum_{i|i\neq j}(\overline{\delta_{i,j}} + r_{u,i})\phi_{i,j}}{\sum_{i|i\neq j}\phi_{i,j}} \tag{3}$$

in which j is the item on which the rating will be predicted for the user u, $\phi_{i,j}$ corresponds to the number of users that have rated both items i and j, $\overline{\delta_{i,j}}$ is the standard deviation between the pair of items i and j and $r_{u,i}$ is the rating given by user u to item i. Standard metrics for measuring the accuracy of a recommendation algorithm are the *Mean Absolute Error* (MAE) and the *Root Mean Square Error* (RMSE), which basically are two different ways to quantify the difference between the true rating r and the prediction one $\hat{r}$. Both prediction methods (*i.e.*, the weighted and the unweighted ones) have a computational cost of $O(n)$, for n the number of possible items of the domain as they basically require to use all the n entries of a row of the deviation matrix (and possibly also the cardinality matrix in case of Weighted Slope One).

A key observation to make is that the prediction of weighted Slope One is based on both local information (*i.e.*, $r_{u,i}$) that the user can store on his computer and global information that is retrieved out of the deviation and cardinality matrices (*i.e.*, $\overline{\delta_{i,j}}$ and $\phi_{i,j}$). Out of the different recommendation algorithms available, we have chosen to focus on the development of the privacy-preserving variant of Slope One, both because it has an accuracy that is closed to state-of-the-art algorithms (as measured by the MAE and the RMSE), and also because its structure, which decouples the local and global information needed for the recommendation, makes it natural to introduce privacy in its framework.

Local computation and Randomized Response Technique. Local computation consists in keeping the profile of a user under his control on his own machine and to perform all the computations (or at least the sensitive ones) needed for the personalization on the clients' side. With respect to privacy, the main advantage of this technique is that the information of the user never leaves his computer, thus limiting the risks of privacy leaks. For instance, in the case of a recommender system, the local computation can be done in a transparent manner to the user by a module directly integrated within his browser. Several systems based on this approach have been proposed in the recent years, in particular in the context of privacy-preserving targeted advertisement systems, such as Privad [7], Adnostic [17] and RePriv [6].

Randomized Response Technique (RRT) can be seen as a specific form of local perturbation method in which the user perturbs himself his data before releasing

it. Originally, this technique was invented by Warner [18] in the 1960's as a survey tool enabling individuals to randomize their answer on questions that are deemed sensitive while still preserving global statistical properties of the sample. In privacy, RRT is a general term referring to any local perturbation technique in which the user locally perturbs his data independently of the data of other users (in contrast to other methods such as k-anonymity sanitizing the data in a global manner).

3 Related Work

RRT methods for recommendation algorithm. Within the context of recommendation systems, Polat and Du [16] have developed a multi-group scheme to randomize the items of a profile. Using this technique, each user partitions the items of his profile into groups of same size and then each group is perturbed independently with some predefined probability p. One of the limits of this approach is that if the adversary has some knowledge about a particular group then this may help him to de-randomize this part of the profile. For instance, if the adversary knows that the first item of the group should be 1, then depending on the value observed on the released data, it can detect whether this group was perturbed or not and then potentially retrieve the original data for this group (at least provided that each item is binary).

In general, one of the main difficulty for assessing the privacy offered by a perturbation method (such as a RRT) is to have a meaningful measure of the risk that the adversary is likely to de-randomize this method. More precisely, to reason about the privacy guarantees provided by such method requires to understand how the adversary might infer the true ratings out of the perturbed ones. For instance, a recent study of Pashalidis and Preneel [15] has evaluated the privacy/utility trade-off provided by different classes of obfuscation strategies in the context of personalized services.

Privacy-preserving variants of Slope One. Basu, Vaidya and Kikuchi have proposed in the recent years three variants of Slope One integrating privacy into the design of the recommendation algorithm [1,2,3]. These three privacy-preserving variants of Slope One rely on very different approaches. For instance, one of the approach achieves privacy through the arbitrary partition of matrices [2] while the second is based on the use of cryptographic techniques [3]. More precisely, the main objective of the protocols proposed in [3] is to compute in a secure and distributed manner the output of the Slope One algorithm (namely the deviation and cardinality matrices) from rating data that is split among several sites.

The third approach is the closest to our work as it also corresponds to a perturbation technique combined with the use of homomorphic encryption [1]. This method adds noise to the deviation matrix during the training phase (*i.e.*, the time at which the matrices are computed) and to the ratings themselves at prediction time. In a nutshell, the prediction query of the user representing the item in which he is interested is protected through the use of homomorphic

encryption scheme, such as the Paillier's cryptosystem [14], that has been set up by the user and which allows to perform arithmetic operations (such as addition and/or multiplication) on encrypted values. More precisely, the true ratings of the user are encrypted homomorphically and then send to the server storing the deviation and cardinality matrices. The server then perform the necessary computations in order to generate an encrypted version of the prediction before sending back the result to the user. Finally, the user can decrypt the result in order to fetch the corresponding prediction. The privacy of this scheme is ensured through the use of the homomorphic encryption for which the user is the only one to know the secret key. The experiments conducted on the Movielens dataset shows that proposed method is still accurate compared to the standard version of the algorithm (as measured in terms of MAE).

4 SlopPy

In this section, we describe SlopPy (for *Slope One with Privacy*), a privacy-preserving version of Slope One in which a user *never* releases directly his personal information (*i.e*, his ratings). In the rest of this section, we give an overview of the general architecture of the system in Section 4.1 (due to space limitations we leave the details of the architecture to the full version of this paper) before describing the SlopPy recommendation algorithm in Section 4.2.

4.1 Overview of the SlopPy Architecture

SlopPy architecture. Figure 1 illustrates the architecture of the SlopPy recommender system. More precisely in SlopPy, each user first perturbs locally his data (Step 1) by applying a *Randomized Response Technique* (RRT) before sending this information to the entity responsible for storing this information through an *anonymous communication channel* (Step 2). This entity is assumed to be *semi-trusted*, also sometimes called *honest-but-curious* in the sense that it is assumed to follow the directives of the protocol (*i.e.*, it will not for instance corrupt the perturbed ratings send by a user or try to influence the output of the recommendation algorithm) but nonetheless tries to extract as much information as it can from the data it receives. Out of the perturbed ratings, the semi-trusted entity constructs two matrices (*i.e.*, the deviation matrix and the cardinality matrix) following the Weighted Slope One algorithm (Step 3). When a user needs a recommendation on a particular movie, he queries these matrices through a variant of a *private information retrieval* scheme [13] (Step 4) hiding the content of his query (*i.e.*, the item he is interested in) to the semi-trusted entity. By combining the data retrieved (Step 5) with his true ratings (which once again are only stored on his machine), the user can then locally compute the output of the recommendation algorithm for this particular item (Step 6).

Philosophy behind SlopPy. Overall, the philosophy of SlopPy is that users contribute to the common good (*i.e.*, the construction of the deviation and cardinality matrices that are needed to perform the recommendation) but still protect

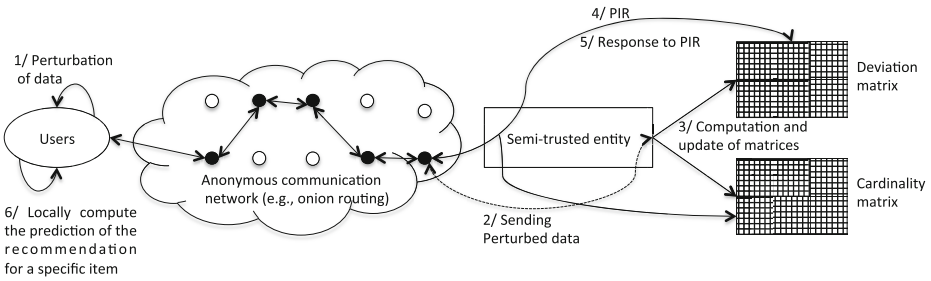

Fig. 1. Overview of the architecture of SlopPy

their privacy by not sending directly their true ratings but rather a perturbed version of it. The semi-trusted entity is responsible for administrating the common good in the sense of computing and storing the deviation and cardinality matrices, which are considered to be *public data* that any user should have the right to access[1]. On the other hand, the true ratings of a particular user are considered to be *private information* and are only stored locally on his machine. The computation of the prediction for a particular recommendation is done by the user querying the matrices through a *private information retrieval* scheme. In a nutshell, this technique allows a user to learn the content of a particular row of the matrix without the semi-trusted entity learning his query. One of the advantage of the centralized aspect of our system is that as the semi-trusted entity has access to all the perturbed ratings of the users, and therefore it can easily maintain and update the deviation and cardinality matrices.

Protection of privacy in SlopPy. In a recommender system, we believe that ensuring privacy amounts to design a system that can hide at least the three following information :

1. The true ratings of the user.
2. The identity of the user behind an action (*e.g.*, an action could be the submission of ratings during the training phase or a prediction request).
3. The content of the query itself (*i.e.*, the item the user is interested in).

In SlopPy, the ratings of the user are protected by randomizing them through the application of a RRT before they are released (see Section 4.2 for more details). In order to hide the identity of the user behind the submission of the ratings, the architecture of SlopPy relies on the use of an anonymous communication channel. This anonymous channel could be either implemented through the use of a proxy that is assumed to be independent of the semi-trusted entity

[1] We acknowledge that this assumption is not necessary compatible with a business model in which the knowledge of such matrices is considered as a business secret by the semi-trusted entity. In such situation, additional mechanisms should be used in order to limit the number of queries that a particular user can perform, for instance by relying on techniques such as oblivious transfer and anonymous e-cash.

(*i.e.*, not colluding with him) or through an anonymous communication network such as TOR (*The Onion Routing*) [5]. The advantage of the latter solution is that preservation of the identity of the user does not only rest on the shoulders of the single entity but rather the trust is spread among several nodes of the anonymous communication network. If the user needs to update his profile on a regular basis, it is possible to envision that the user can choose a long-term pseudonym. This pseudonym could be for instance the public verification key of a digital signature scheme that has been set up by the user. When the user needs to update his profile, he simply sends the ratings of new items that he has not rated previously through an anonymous communication channel along with a proof of his identity in the form of a signature on his updated profile. In particular, the user should not send another fresh randomized version of items that he has rated in the past. Otherwise, by observing several randomizations of the same rating, it may become possible for the server to de-randomize the ratings of a particular user by doing a simple average. Finally, the protection of the content of a query is ensured through a variant of a private information retrieval scheme based on homomorphic encryption scheme. This protocol allows to learn the content of a particular row of one of the matrices for an optimal communication cost of $\Theta(n)$, for n the number of items in the domain considered. We defer the details of this protocol as well as the detailed analysis of the architecture for the full version of the paper but overall the asymptotic complexity of SlopPy is the same as the Slope One algorithm. However, in practice the use of an anonymous channel as well as the use of homomorphic encryption induce an overhead that is likely to impact the performance in a non-negligible manner.

4.2 SlopPy Recommendation Algorithm

Randomization operator. The core of the SlopPy architecture is a recommendation algorithm, which is effectively a privacy-preserving variant of Weighted Slope One. The main modification of the algorithm consists in the participants sending a perturbed version of their ratings rather than their true ratings. More precisely, each participant applies a RRT on his profile in order to obtain the perturbed version. Only the perturbed version is released publicly to the semi-trusted third party responsible for constructing the deviation and the cardinality matrices while the true ratings are stored locally on the user's machine. We have constructed four different randomization operators for the RRT that we describe thereafter.

1. *Independent randomization.* With the method IndRand, each item in the profile is randomized in an independent manner with a probability p, which is an input parameter of the RRT method, and left untouched with the complementary probability $1-p$. More precisely, if $p = 0$ then no randomization occurs (*i.e.*, the true ratings are unmodified) while a value of $p = 1$ means that *each* item of the profile is randomized. When a rating is randomized

then the perturbed rating is drawn uniformly at random among all the values except the original one. (Another possibility would have been to randomize over all possible values.)

2. *Deviation.* With the method Deviation, each rating is perturbed such that its value is likely to be slightly increased or decreased compared to the original value. This randomization operator is implemented by first drawing a random number a predefined interval chosen according to the rating scale and then generating a perturbed rate that corresponds to the addition of this random number to the original rating. If the generated rating is above or below the maximal (or minimal) possible rating then to ensure that this value remains within the set of possible rates by rounding it to the nearest integer within this set. For instance, in the Movielens dataset, the set of possible rates is $S = \{1, 2, 3, 4, 5\}$ and we choose to draw the random number in the interval $[-2, 2]$. In this setting, if the true rate had originally a value of 5 and the randomization should result in the rate being modified to 6 due to the rounding, instead the rating of 5 is chosen in order to remain within the set of possible values. We acknowledge that this method is not perfect in the sense that it makes some ratings more probable than others, which could be an issue with respect to privacy. In the future, we plan to investigate the potential risks for privacy induced by this method in order to better understand its limits.
3. *Deviation and randomization.* The method DevAndRand is simply composed of the application of the Deviation method on all ratings, followed by the application of the IndRand method on the ratings resulting from the previous method.
4. *Block randomization.* The method BlockRand decomposes the profile of the user into block of ratings of same size and then for each block, all the ratings of this block are randomized with probability p (as with the IndRand but with the difference that each rating is not randomized independently) or all left untouched. For the experiments reported in this paper, we have chosen to use a block size of 10 items for all users but of course other block sizes are possible (*e.g.*, block size of 5 or 15 items).

Prediction method. Beside the different randomization operators, we have also studied two different ways to predict a rating (both are based on Equation 3). The first method called PerturbedRec performs the recommendation by taking into account the perturbed version of the ratings while computing Slope One. On the other hand, the second method called OriginalRec injects the true ratings in the part of the Slope One algorithm that can be computed locally. This is made possible by the fact that all the users keep locally a copy of their true ratings. More precisely, using the method PerturbedRec means that the perturbed ratings generated during Step 1 will also be considered as the local input of the prediction during Step 6 while with the method OriginalRec the local inputs to Step 6 are the original ratings unmodified.

5 Experimental Results

In this section, we briefly report on the preliminary results we have obtained by testing the SlopPy algorithm. More precisely, we have tested the different randomization and prediction methods detailed in Section 4.2 on the Movielens dataset and compared the accuracy obtained for different parameters as well as against the "non-private" Weighted Slope One algorithm. The Movielens dataset contains 100 000 ratings provided by 943 users on 1682 different movies (*i.e.*, items), with an average number of ratings of 106 ratings per user but a high variance. Therefore, the number of items $n = 1682$ and the dataset is sparse in the sense that an overall of 100 000 ratings out of the 1 586 126 possible ratings corresponds to a density of 6.3%.

To evaluate the accuracy of the recommendation algorithm of SlopPy, we rely on two metrics: the standard Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE). The MAE measures the average absolute error between the true rating $r_{u,i}$ provided by user u on item i and the rating $\hat{r}_{u,i}$ predicted by the recommendation algorithm. More precisely, the MAE is generated by computing the absolute difference $\mid \hat{r}_{u,i} - r_{u,i} \mid$ for each user u and item i whose rating is known and then averaging by dividing by the total number of predictions. The smaller the MAE, the more accurate is the recommendation. Indeed, a MAE whose value is equal to zero would correspond to a recommendation algorithm making a perfect prediction, which is never observed in practice. Similarly to the MAE, the RMSE also quantifies the difference between a predicted rating $\hat{r}_{u,i}$ and a true rating $r_{u,i}$. More precisely, the RMSE corresponds to the square root of the sum of the squared difference between $(\hat{r}_{u,i} - r_{u,i})^2$ normalized by the total number of predictions. Like the MAE, a small RMSE is an indication of a good accuracy for the recommendation.

We compare SlopPy by using the standard Weighted Slope One algorithm as the baseline. On the Movielens dataset, the baseline displays a MAE of 0.68 and a RMSE of 0.85. As SlopPy perturbs the inputs provided to the recommendation algorithm by performing the RRT on the ratings used to build the deviation and cardinality matrices, we expect intuitively that the accuracy of the recommendation will be degraded as well, thus resulting in a higher MAE and RMSE. This intuition was confirmed by the results of experiments as demonstrated by Figures 2 and 3. Furthermore, Figure 3 clearly shows that predicting a rating while using the original ratings kept locally lead to a high accuracy even if the ratings transmitted have been perturbed heavily, which is not the case when the prediction is done by using also the perturbed ratings (*cf.* Figure 2). In addition, SlopPy seems to be robust as the MAE and the RMSE smoothly vary when the number of perturbed ratings increases. With respect to the Deviation method (which is equivalent to the DevAndRand method with a randomization probability $p = 0$), the MAE and the RMSE were respectively around 0.74 and 1.05 when PerturbedRec was used as a prediction method, and 0.70 (MAE) and 0.88 (RMSE) when relying on OriginalRec for the prediction. In terms of computational time, SlopPy (much like Slope One) is very efficient and running it

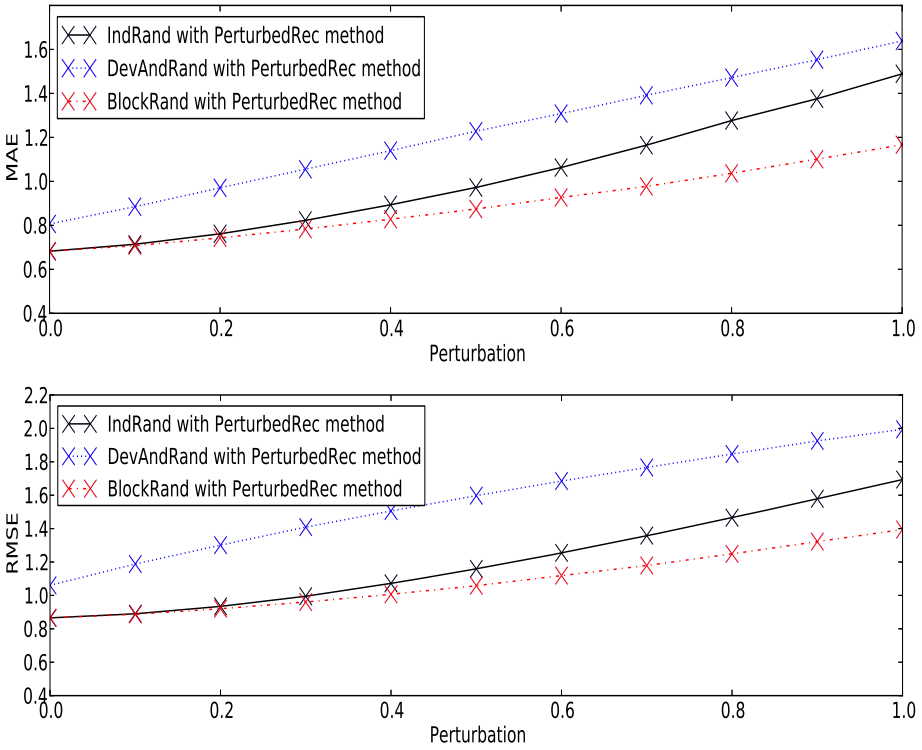

Fig. 2. Computation of the MAE and RMSE for the IndRand, DevAndRand and BlockRand randomization methods for the prediction method PerturbedRec. The x-axis indicates the *perturbation* in terms of the probability p. For instance, a perturbation of 0.9 means that on average 9 ratings out of 10 are randomized (or conversely that 1 rating out of 10 is left untouched).

on the whole Movielens dataset takes approximately two minutes on a MacBook Pro with a 2.4GHz Intel Core i7 and 4GB of RAM.

6 Discussion and Future Work

In the future, we would like to investigate other randomization operators such as an erasure/creation operator that will delete some true ratings and on the contrary also generate a random rating for a particular item when there was none. In particular, we would like to observe how this type of randomization affects the utility of the recommendation. We believe that the use of such an operator (possibly combined with other randomization operators) is one of the few ways to avoid linking attacks by which the adversary (which could be for instance the semi-trusted entity) uses some *a priori* knowledge to de-anonymize

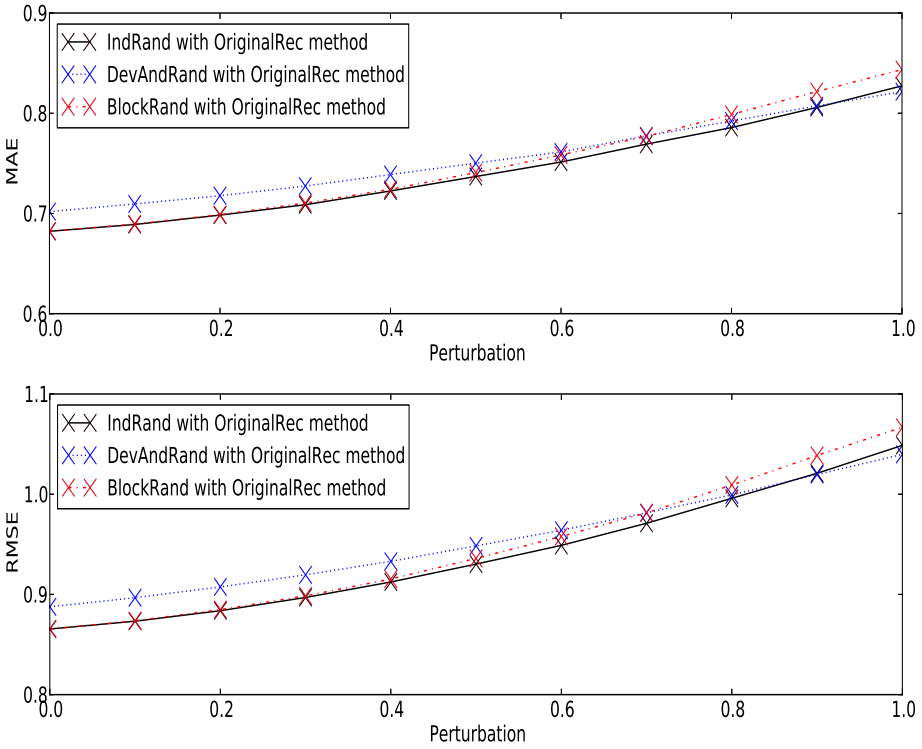

Fig. 3. Computation of the MAE and RMSE for the IndRand, DevAndRand and BlockRand randomization methods for the prediction method OriginalRec. The x-axis indicates the *perturbation* in terms of the probability p. For instance, a perturbation of 0.9 means that on average 9 ratings out of 10 are randomized (or conversely that 1 rating out of 10 is left untouched).

a particular dataset. For instance, if the adversary knows that a Alice has seen 3 movies among which two are quite uncommon, the combination of these 3 movies could act as quasi-identifiers and be potentially used to de-anonymize Alice when she submit her ratings to SlopPy even if she submit her records anonymously after having perturbed them. This type of inference attack is similar in spirit to the de-anonymization attack conducted by Narayanan and Shmatikov on the Netflix dataset [11].

With respect to the architecture, we would like to design a distributed version of the semi-trusted entity. Indeed in the current architecture, all the trust rests on the shoulders of this entity, which is assumed to follow the recipe of the Slope One algorithm and not to try to influence the outcome of the recommendation. This security assumption is not necessarily safe to make in some context in which the semi-trusted entity might be tempted to influence the recommendation in order for some items to be recommended more often (for instance the semi-trusted

entity might make more money out of these items). Distributing the semi-trusted entity, for instance by relying on techniques such as threshold cryptography, is a possible approach to split the trust on several entities instead of a single one. Moreover, we are planning to run experimentally a private information retrieval to evaluate the practical efficiency of such approach and to use TOR as a way to simulate the anonymous channel by which a user can submit a query to the private information retrieval scheme.

We also want to study in more details the interplay between privacy and utility and how users might adjust their privacy level by themselves tailoring the level of perturbation that they apply to their needs. In SlopPy, this can be done naturally by allowing users to set up their level of noise for the RRT technique. For instance, Kobsa [8] differentiates between three types of individuals:

- *Extremely concerned.* These individuals want to know how their personal information are used and which data is possibly disclosed to third parties.
- *Somewhat concerned.* These individuals do care about their privacy but not at the same level as the extremely concerned.
- *Mildly concerned.* These individuals are not really concerned about their privacy in the sense that they do not even wish to know which information is collected about them and how this information is used.

We propose to adopt this taxonomy in order to model the different types of users of a recommender system. Depending on the chosen level of privacy for each group, the randomization applied will be more or less intense. We plan to investigate how varying the proportion of the different groups of individuals influence the overall quality of the recommendation. Intuitively, if most of the population is composed of the extremely concerned then it is to be expected to the accuracy of the recommendation will be lower than if the three populations are uniformly present but of course this intuition needs to be verified and quantified.

Finally, in order to give precise privacy guarantees we want to analyze the possible risks of de-randomization on the proposed methods by studying the distributions that they induced on the outputs (*i.e.*, ratings). Indeed, if the proposed randomized operator leads to distributions that are closed to each other whatever the original input then this means that the probability of de-randomizing a particular perturbed rating is small.

Acknowledgments. This research has been funded by the Quaero program and the Inria large scale project CAPPRIS (Collaborative Action on the Protection of Privacy Rights).

References

1. Basu, A., Vaidya, J., Kikuchi, H.: Perturbation Based Privacy Preserving Slope One Predictors for Collaborative Filtering. In: Dimitrakos, T., Moona, R., Patel, D., McKnight, D.H. (eds.) IFIPTM 2012. IFIP AICT, vol. 374, pp. 17–35. Springer, Heidelberg (2012)

2. Basu, A., Vaidya, J., Kikuchi, H.: Privacy preserving weighted Slope One predictor for item-based collaborative filtering. In: Proceedings of the International Workshop on Trust and Privacy in Distributed Information Processing (co-organized with IFIPTM 2011) (2011)
3. Basu, A., Vaidya, J., Kikuchi, H.: Efficient privacy-preserving collaborative filtering based on the weighted Slope One predictor. Journal of Internet Services and Information Security 1(4) (2011)
4. Das, A., Datar, M., Garg, A.: Google news personalization: Scalable online collaborative filtering. In: Proceedings of the 16th International Conference on World Wide Web (WWW 2007), pp. 271–280 (2007)
5. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium, pp. 303–320 (2004)
6. Fredrikson, M., Livshits, B.: RePriv: Re-imagining content personalization and in-browser privacy. In: Proceedings of the 32nd IEEE Symposium on Security and Privacy, pp. 131–146 (2011)
7. Guha, S., Cheng, B., Francis, P.: Privad: practical privacy in online advertising. In: Proceedings of the 8th USENIX Symposium on Networks, System Design and Implementation (2011)
8. Kobsa, A.: Privacy-enhanced personalization. Communications of the ACM 50(8), 24–33 (2007)
9. Lemire, D., Maclachlan, A.: Slope One predictors for online rating-based collaborative filtering. In: Proceedings of the 2005 SIAM International Data Mining Conference (SDM 2005) (2005)
10. Linden, G., Smith, B., York, J.: Amazon.com recommendations item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (2003)
11. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proceedings of the 29th IEEE Symposium on Security and Privacy, pp. 111–125 (2008)
12. Olesen, H., Noll, J., Hoffmann, M.: User profiles, personalization and privacy (2009)
13. Ostrovsky, R., Skeith III, W.E.: A Survey of Single-Database Private Information Retrieval: Techniques and Applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
14. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
15. Pashalidis, A., Preneel, B.: Evaluating tag-based preference obfuscation systems. IEEE Transactions on Knowledge and Data Engineering 24(9), 1613–1623 (2012)
16. Polat, H., Du, W.: Achieving Private Recommendations Using Randomized Response Techniques. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 637–646. Springer, Heidelberg (2006)
17. Toubiana, V., Narayanan, A., Boneh, D., Nissenbaum, H., Barocas, S.: Adnostic: Privacy preserving targeted advertising. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2010 (2010)
18. Warner, S.L.: Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association 60, 63–69 (1965)