Heterogeneous Recommendations: What You Might Like To Read After Watching Interstellar

Rachid Guerraoui
EPFL
rachid.guerraoui@epfl.ch

Anne-Marie Kermarrec
Inria
anne-marie.kermarrec@inria.fr

Tao Lin
EPFL
tao.lin@epfl.ch

Rhicheek Patra
EPFL
rhicheek.patra@epfl.ch

ABSTRACT

Recommenders, as widely implemented nowadays by major e-commerce players like Netflix or Amazon, use collaborative filtering to suggest the most relevant items to their users. Clearly, the effectiveness of recommenders depends on the data they can exploit, i.e., the feedback of users conveying their preferences, typically based on their past ratings.

As of today, most recommenders are *homogeneous* in the sense that they utilize one specific application at a time. In short, Alice will only get recommended a movie if she has been rating movies. But what if she has been only rating books and would like to get recommendations for a movie? Clearly, the multiplicity of web applications is calling for *heterogeneous* recommenders that could utilize ratings in one application to provide recommendations in another one.

This paper presents X-Map, a heterogeneous recommender. X-Map leverages *meta-paths* between heterogeneous items over several application domains, based on users who rated across these domains. These *meta-paths* are then used in X-Map to generate, for every user, a profile (*AlterEgo*) in a domain where the user might not have rated any item yet. Not surprisingly, leveraging *meta-paths* poses non-trivial issues of (a) *meta-path-based inter-item similarity*, in order to enable accurate predictions, (b) *scalability*, given the amount of computation required, as well as (c) *privacy*, given the need to aggregate information across multiple applications.

We show in this paper how X-Map addresses the above-mentioned issues to achieve accuracy, scalability and differential privacy. In short, X-Map weights the meta-paths based on several factors to compute inter-item similarities, and ensures scalability through a layer-based pruning technique. X-Map guarantees differential privacy using an exponential scheme that leverages the meta-path-based similarities while determining the probability of item selection to construct the *AlterEgos*. We present an exhaustive experimental evaluation of X-Map using real traces from Amazon. We show that, in terms of accuracy, X-Map outperforms alternative heterogeneous recommenders and, in terms of throughput, X-Map achieves a linear speedup with an increasing number of machines.

Proceedings of the VLDB Endowment, Vol. 10, No. 10

1. INTRODUCTION

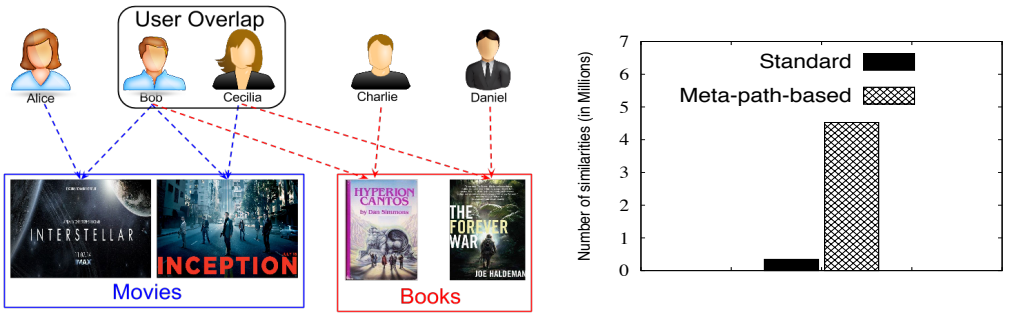
The vast amount of information available in the Internet calls for *personalization* schemes, such as *recommenders*, that aid the users in their web navigation activities. Recommenders seek to suggest *relevant* items to users based on their *preferences* (expressed through ratings), by typically relying on *Collaborative Filtering* (CF) algorithms [28] to utilize similarities between users. Such similarities are computed using the rating history of users. The axiom here is that *if Alice and Bob liked the same items in the past, they will like the same items in the future with a high probability.*

1.1 Motivation

Although widely used today, recommenders are mainly applied in a *homogeneous* way: movie recommenders like IMDb or Netflix, news recommenders like Google News or Yahoo News, as well as music recommenders like Last.fm or Spotify, each focuses on a single specific application domain. In short, you will be recommended books only if you rated books, and you will be recommended movies only if you rated movies. Given the growing multiplicity of web applications, homogeneity is a major limitation. For example, with most state-of-the-art recommenders, Alice who just joined a book-oriented web application, and never rated any book before, cannot be recommended any relevant book, even if she has been rating many movies.

We argue that the next level to personalization is *heterogeneity*, namely *personalization across multiple domains* [11]. Heterogeneous preferences on the web, i.e., preferences from multiple application domains, should be leveraged to improve personalization, not only for users who are new to a given domain (i.e, *cold-start* situation), but also when the data is *sparse* [2] (e.g, very few ratings per user). In fact, if a user, say Alice, likes the *Interstellar* movie, then a heterogeneous personalization scheme could actually recommend her books such as *The Forever War* by Joe Haldeman. To get an intuition of how such recommendation can be made by going beyond standard schemes, consider the scenario depicted in Figure 1(a) where five users rated *at most* one book. Indeed, according to a standard metric (*adjusted cosine* [29]), the similarity between *Interstellar* and *The Forever War* is 0, for there are no common users who rated both. However, a closer look reveals the following *meta-path* [1] between these two heterogeneous items: $Interstellar \xrightarrow{Bob} Inception \xrightarrow{Cecilia} The\ Forever\ War$.

[1]We call *meta-path* any path involving heterogeneous items, e.g., movies and books.



(a) **A simple scenario depicting heterogeneity across two domains.**

(b) **The effect of *meta-paths* in computing heterogeneous similarities.**

Figure 1: Heterogeneous recommendation using meta-paths.

Figure 1(b) compares the number of heterogeneous similarities that could be exhibited with or without using meta-paths on real-world traces from Amazon (using two domains: movies and books). Meta-path-based heterogeneous similarities clearly lead to better recommendation quality as we show later in § 6.4.

1.2 Challenges

While appealing, building a practical heterogeneous meta-path-based recommender raises several technical challenges.

Meta-path-based similarity. Consider an undirected graph G where the vertices represent the items and each edge e_{ij} is associated with a weight s_{ij}, representing the similarity between items i and j. A meta-path in G can be defined as a sequence of adjacent vertices (movies or books) connected by edges in G. Computing a *heterogeneous* similarity based on these meta-paths is, however, not straightforward. Such similarity could be affected by factors like the number of users involved, directly or indirectly (in the meta-paths), as well as the strength of the ties between item-pairs connected by (shorter) meta-paths. The challenge here is to capture these factors in a way that improves the accuracy of heterogeneous similarities.

Scalability. Clearly, the computational complexity increases many-fold while computing meta-path-based similarities. Computing all possible meta-paths on a large-scale graph with millions of vertices (heterogeneous items) can quickly become computationally intractable.

Privacy. Heterogeneous recommendations also raise privacy concerns. For example, the new transitive link between Alice and Cecilia (Figure 1(a)) provides the opportunity for a curious user, say Alice, to discover the *straddlers*: people like Bob or Cecilia who connect multiple domains. Alice can actually determine the item(s) that allows her to get this recommendation by pretending to be another user and incrementally rating items until she gets the recommendation. This is similar to the privacy risk in statistical database queries where inferences can be derived from combinations of queries [26]. As pointed out in [27], such straddlers are at a privacy risk, and information about their preferences could be used in conjunction with other data sources to uncover identities and reveal personal details. This can be particularly problematic across different applications like Netflix (movies) and Last.fm (music).

Recent heterogeneous recommenders [11, 30], extending classical homogeneous recommendation schemes across domains, are neither scalable nor private, and hence are not suitable for applications involving millions of users and items.

1.3 Contributions

In this paper, we present a recommender we call X-Map: **Cross**-doma**i**n **p**ersonalization system. X-Map fully utilizes the overlap among users across multiple domains, as depicted in Figure 1(a). This overlap is often derived from profiles maintained by users across various web applications along with interconnection mechanisms for cross-system interoperability [10] and cross-system user identification [9]. At the heart of X-Map lie several novel ideas.

- We introduce a novel similarity metric, X-Sim, which computes a meta-path-based transitive closure of inter-item similarities across several domains. X-Sim involves adaptations, to the heterogeneous case, of classical *significance weighting* [16] (to account for the number of users involved in a meta-path) and *path length* [27] (to capture the effect of meta-path lengths) schemes.
- We introduce the notion of *AlterEgos*, namely artificial profiles (created using X-Sim), of users even in domains where they have *no* or *very little* activity yet. We generate an AlterEgo profile (of Alice) in a target domain leveraging an item-to-item mapping from a source domain (e.g., movies) to the target domain (e.g., books). AlterEgos enable to integrate any standard recommendation feature in the target domain and preserve, for example, the temporal behaviour of users [12].
- We use an effective layer-based pruning technique for selecting meta-paths. AlterEgos, acting as a caching mechanism, alleviate computational intractability by only using the information from the target domain. Combined with our layer-based pruning technique, AlterEgos enable X-Map to scale almost linearly with the number of machines (a major requirement for the deployment of a recommender in a practical environment). We illustrate this scalability through our implementation of X-Map on top of Apache Spark [38].
- We introduce an obfuscation mechanism, based on meta-path-based similarities, to guarantee differentially private AlterEgos. We adapt, in addition, a probabilistic technique, inspired by Zhu et al. [39,40], to protect the privacy of users in the target domain. Interestingly, we show that, despite these privacy techniques, X-Map outperforms the recommendation accuracy of alternative non-private heterogeneous approaches [5, 6, 11].
- We deployed an online recommendation platform, using X-Sim on a database of 660K items, to recommend books and movies to users based on their search queries at:

http://x-map.work/

Books like *The Da Vinci Code* are indeed recommended when the search query is the *Angels & Demons* (2009) movie. Currently, we support Chrome, Safari and Firefox browsers.

1.4 Roadmap

The rest of the paper is structured as follows. We recall some background on collaborative filtering and differential privacy before formulating the heterogeneous recommendation problem in § 2. Next, we introduce our X-Sim metric along with our layer-based pruning technique for selecting meta-paths in § 3. For pedagogical reasons, we first present a non-private variant of our recommendation scheme (NX-Map [2]) before the private one (X-Map) in § 4. We then present our scalable implementation of X-Map in § 5 and our experimental results in § 6. We review the related work in § 7 and conclude our paper in § 8. Our open-sourced implementation of X-Map as well as the detailed proofs of our privacy guarantees are provided in a GitHub repository [1].

2. BACKGROUND AND PROBLEM

2.1 Collaborative Filtering

Collaborative Filtering (CF) algorithms fall mainly in two categories: *memory-based* [28, 33] and *model-based* [17, 20, 37]. Memory-based algorithms compute the *top-k* like-minded users for any given user (say Alice), denoted as the *neighbors* of Alice, from the training database, and then make recommendations to that user based on the rating history of her neighbors. In contrast to memory-based algorithms, model-based ones first extract some information about users (including Alice) from the database to train a *model* and then use this model to make recommendations for the users (including Alice). Memory-based algorithms are more flexible in practice compared to model-based ones [18]. It is usually more difficult to add new incoming data to model-based systems because building a model is often a time-consuming and resource-consuming process.

Among memory-based CF schemes, *neighbor-based CF* (based on *k nearest neighbor* (kNN) algorithms) are more popular and widely used in practice [29]. We summarize the basic notations used in this paper in Table 1. A user-based CF scheme, as depicted in Algorithm 1, computes the k neighbors of Alice with highest user-to-user similarities (Phase 1 in Algorithm 1). These neighbors' profiles are then used to compute the best recommendations for Alice (Phase 2 in Algorithm 1). An item-based CF scheme, as depicted in Algorithm 2, computes the k most similar items for every item based on item-to-item similarities (Phase 1 in Algorithm 2) and then computes the recommendations for Alice based on her ratings for similar items (Phase 2 in Algorithm 2). As we will explain in § 4, X-Map currently supports user-based as well as item-based CF schemes while providing recommendations to its users. Different practical deployment scenarios benefit from the proper choice of the recommendation algorithm. One requirement, which is crucial to any deployment scenario, is *Scalability*. From the scalability aspect, item-based recommenders are suitable for big e-commerce players with more users than items whereas user-based recommenders are preferable for new players with more items than users due to the number of item-item or user-user similarities leveraged by the respective recommenders. Moreover, the user-user similarities are more dynamic than item-item similarities for movies/books [3] due to the temporality in users' online behaviour and hence require frequent updates. For interested readers, we provide an empirical demonstration of the influence of item-based and user-based recommenders in two different practical deployment scenarios in our companion technical report [1].

[2] NX-Map illustrates the effectiveness of X-Sim and AlterEgos for heterogeneous recommendations without the privacy overhead and could be used for applications with heterogeneous intra-company services like *Google Play*.

Table 1: Notations

Notations	
$\mathcal{D}$	any single domain
$\mathcal{U}$	the set of users in a domain
$\mathcal{I}$	the set of items in a domain
$r_{u,i}$	the rating provided by a user u for an item i
$M_{\mathcal{D}}$	the matrix for the ratings provided by the users [3]
$\bar{r}_u$	the average rating of u over all items rated by u
$\bar{r}_i$	the average rating for i over all users who rated i
X_u	the set of items rated by u (user profile)
Y_i	the set of users who rated i (item profile)
$\tau(u, v)$	the similarity measure between u and v

Algorithm 1 User-based CF

Input: $\mathcal{I}$: Set of all items; $\mathcal{U}$: Set of all users; X: set of profiles of all users where X_A denotes the profile of Alice (with user-id as A) which contains the items rated by A.
Output: R_A: Top-N recommendations for Alice

Phase 1 - Nearest Neighbor Selection: kNN(A,$\mathcal{U}$,X)

Input: $\mathcal{U}$, X
Output: k nearest neighbors for Alice
1: var τ_A; ▷ Dictionary with similarities for Alice
2: var $\bar{r}$; ▷ Average rating for each item
3: **for** u in $\mathcal{U}$ **do**
4:
$$\tau_A[u] = \frac{\sum_{i \in X_A \cap X_u} (r_{A,i} - \bar{r}_i)(r_{u,i} - \bar{r}_i)}{\sqrt{\sum_{i \in X_A} (r_{A,i} - \bar{r}_i)^2} \sqrt{\sum_{i \in X_u} (r_{u,i} - \bar{r}_i)^2}} \quad (1)$$
5: **end for**
6: $\gamma_A = \tau_A.sortByValue(ascending = false)$;
7: $N_A = \gamma_A[:k]$ ▷ Top k neighbors
8: **return:** N_A;

Phase 2 - Recommendation: Top-N(N_A,$\mathcal{I}$)

Input: $\mathcal{I}$, N_A
Output: Top-N recommendations for Alice
9: var Pred; ▷ Dictionary with predictions for Alice
10: var $\bar{r}$; ▷ Average rating for each user
11: **for** i : *item* in $\mathcal{I}$ **do**
12:
$$Pred[i] = \bar{r}_A + \frac{\sum_{B \in N_A} \tau(A, B) \cdot (r_{B,i} - \bar{r}_B)}{\sum_{B \in N_A} |\tau(A, B)|}; \quad (2)$$
13: **end for**
14: $\phi_A = Pred.sortByValue(ascending = false)$;
15: $R_A = \phi_A[:N]$ ▷ Top N recommendations
16: **return:** R_A;

2.2 Differential Privacy

Differential privacy [14] was initially devised in a context where statistical information about a database is released without revealing information about its individual entries. Differential privacy provides formal privacy guarantees that

[3] $M_{\mathcal{D}}$ is typically a sparse matrix. If u has not rated i, we use the average rating of i as $r_{u,i}$ to complete $M_{\mathcal{D}}$.

Algorithm 2 Item-based CF

Input: $\mathcal{I}$: Set of all items; $\mathcal{U}$: Set of all users; Y: Set of profiles of all items where Y_j denotes the list of users who rated an item with item-id j.
Output: R_A: Top-N recommendations for Alice (A)

Phase 1 - Nearest Neighbor Selection: kNN(j,$\mathcal{I}$,Y)

Input: $\mathcal{I}$, Y
Output: k most similar items to item j
1: var τ_j; ▷ Dictionary for item similarities with j
2: var $\bar{r}$; ▷ Average rating for each user
3: **for** i in $\mathcal{I}$ **do**
4:
$$\tau_j[i] = \frac{\sum_{u \in Y_j \cap Y_i}(r_{u,j} - \bar{r_u})(r_{u,i} - \bar{r_u})}{\sqrt{\sum_{u \in Y_j}(r_{u,j} - \bar{r_u})^2}\sqrt{\sum_{u \in Y_i}(r_{u,i} - \bar{r_u})^2}} \quad (3)$$
5: **end for**
6: $\gamma_j = \tau_j.sortByValue(ascending = false)$;
7: $N_j = \gamma_j[:k]$ ▷ Top k neighbors
8: **return:** N_j;

Phase 2 - Recommendation: Top-N(Y,$\mathcal{I}$)

Input: $\mathcal{I}$, Y
Output: Top-N recommendations for Alice
9: var Pred; ▷ Dictionary with predictions for Alice
10: var $\bar{r}$; ▷ Average rating for each item
11: **for** i : *item* in $\mathcal{I}$ **do**
12:
$$Pred[i] = \bar{r}_i + \frac{\sum_{j \in N_i} \tau(i,j) \cdot (r_{A,j} - \bar{r}_j)}{\sum_{j \in N_i} |\tau(i,j)|}; \quad (4)$$
13: **end for**
14: $\phi_A = Pred.sortByValue(ascending = false)$;
15: $R_A = \phi_A[:N]$ ▷ Top N recommendations
16: **return:** R_A;

do not depend on an adversary's background knowledge (including access to other databases) or computational power. More specifically, differential privacy is defined as follows.

DEFINITION 1 (DIFFERENTIAL PRIVACY [14]). *A randomized function $\mathcal{R}$ ensures ϵ-differential privacy if for all datasets D_1 and D_2, differing on at most one user profile, and all $t \in Range(\mathcal{R})$, the following inequality always holds:*

$$\frac{Pr[\mathcal{R}(D_1) = t]}{Pr[\mathcal{R}(D_2) = t]} \leq exp(\epsilon) \quad (5)$$

2.3 Heterogeneous Recommendation Problem

Without loss of generality, we formulate the problem using two domains, referred to as the *source* domain ($\mathcal{D}^S$) and the *target* domain ($\mathcal{D}^T$). We use superscript notations ^{S} and ^{T} to differentiate the source and the target domains. We assume that users in $\mathcal{U}^S$ and $\mathcal{U}^T$ overlap, but $\mathcal{I}^S$ and $\mathcal{I}^T$ have no common items. This captures the most common heterogeneous personalization scenario in e-commerce companies such as Amazon or eBay nowadays. The heterogeneous recommendation problem can then be stated as follows.

PROBLEM 1. *Given any source domain $\mathcal{D}^S$ and any target domain $\mathcal{D}^T$, the heterogeneous recommendation problem consists in recommending items in $\mathcal{I}^T$ to users in $\mathcal{U}^S$ based on the preferences of $\mathcal{U}^S$ for $\mathcal{I}^S$, $\mathcal{U}^T$ for $\mathcal{I}^T$ and $\mathcal{U}^S \cap \mathcal{U}^T$ for $\mathcal{I}^S \cup \mathcal{I}^T$.*

In other words, we aim to recommend items in $\mathcal{I}^T$ to a user who rated a few items (sparsity) or no items (cold-start) in $\mathcal{I}^T$. Figure 1(a) conveys the scenario that illustrates this problem. The goal is to recommend new relevant items from $\mathcal{D}^T$ (e.g., books) either to Alice who never rated any book (cold-start) or to Bob who rated only a single book (sparsity). Both the users rated items in $\mathcal{D}^S$ (e.g., movies).

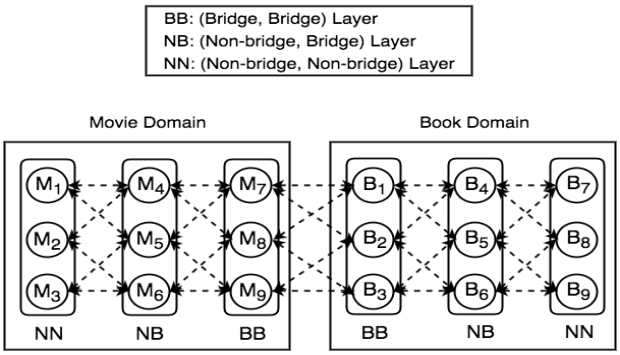


Figure 2: Layer-based pruning in X-MAP.

3. X-SIM

In this section, we present X-SIM, our novel similarity metric designed for heterogeneous recommendation along with our meta-path pruning technique.

3.1 Baseline Similarity Graph

We first build a baseline similarity graph where the vertices are the items and the edges are weighted by the similarities. We could use here any classical item-item similarity metric like Cosine, Pearson, or Adjusted-cosine [29] for baseline similarity computations. We choose to use adjusted-cosine for it is considered the most effective [29]:

$$s_{ac}(i,j) = \frac{\sum_{u \in Y_i \cap Y_j}(r_{u,i} - \bar{r_u})(r_{u,j} - \bar{r_u})}{\sqrt{\sum_{u \in Y_i}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{u \in Y_j}(r_{u,j} - \bar{r_u})^2}} \quad (6)$$

In this first step, we compute the (baseline) similarities by integrating both $\mathcal{D}^S$ and $\mathcal{D}^T$ as a single domain. We denote by G_{ac} [4] the resulting similarity graph in which any two items are *connected* if they have common users. As shown in Figure 1(b), the limitation of adjusted-cosine similarity leads to sparse connections in G_{ac}. We address this sparsity issue of G_{ac} precisely by extending it with *meta-paths* connecting both domains.

Clearly, a brute-force scheme considering all possible meta-paths would be computationally intractable and not scalable. Assuming m items in the database, the time complexity of such a brute-force scheme (computing similarity for every pair of items) would be $O(m^2)$, which is not suitable for big datasets like the Amazon one with millions of items. X-MAP uses a layer-based technique to prune the number of meta-paths, thereby leading to $O(km) \simeq O(m)$ time complexity where $k \lll m$.

3.2 Layer-based Pruning

Based on the baseline similarity graph, we determine what we call *bridge items*, namely any item i in a domain $\mathcal{D}$ which connects to some item j in another domain $\mathcal{D}'$. Both i and j are bridge items in this case. These bridge items are ascertained based on the overlapping users from both domains. We accordingly call any item that is not a bridge item a *non-bridge item*.

X-MAP's pruning technique partitions the items from $\mathcal{D}^S$ and $\mathcal{D}^T$ into six possible layers, based on their connections with other items, as we explain below. In turn, the items in each domain, say $\mathcal{D}$, are divided into three layers (Figure 2).

- *BB-layer.* The (Bridge, Bridge)-layer consists of the bridge

[4] Here *ac* denotes adjusted cosine.

items of $\mathcal{D}$ connected to the bridge items of another domain.

- *NB-layer.* The (Non-bridge, Bridge)-layer consists of the non-bridge items of $\mathcal{D}$ which are connected to bridge items of $\mathcal{D}$.
- *NN-layer.* The (Non-bridge, Non-bridge)-layer consists of the non-bridge items of $\mathcal{D}$ which are not connected to other bridge items.

X-MAP then considers only the paths crossing different layers, which we call *meta-paths*. Since we use a k-nearest neighbor method in X-MAP, each item i in layer l is connected to the top-k items from every neighboring layer l' based on the item-item similarities. We describe our layered meta-path selection in more details in § 5.

3.3 X-SIM: A Novel Similarity Metric

Consider any two items i and j. We denote by $Y_{i\geq\bar{i}}$ the set of users who rated item i higher than or equal to the average rating for i over all the users in the database who rated i. We also denote by $Y_{i<\bar{i}}$ as the set of users who rated item i lower than the average rating for i. Additionally, we denote by $|Y_i|$ the cardinality of the set Y_i.

DEFINITION 2 (WEIGHTED SIGNIFICANCE). *Given any pair of items i and j, we define weighted significance ($S_{i,j}$) as the number of users who mutually like or dislike this given pair. Formally, we define the weighted significance ($S_{i,j}$) between i and j as follows.*

$$S_{i,j} = \underbrace{\left|Y_{i\geq\bar{i}} \cap Y_{j\geq\bar{j}}\right|}_{Mutual\ like} + \underbrace{\left|Y_{i<\bar{i}} \cap Y_{j<\bar{j}}\right|}_{Mutual\ dislike}$$

Intuitively, a higher significance value implies higher importance of the similarity value. For example, a similarity value of 0.5 between an item-pair (i,j) with $S_{i,j} = 1000$ is more significant than a similarity value of 0.5 between an item-pair (i,k) with $S_{i,k} = 1$ (for the latter may be a result of pure coincidence). [5]

DEFINITION 3 (META-PATH). *Given G and its six corresponding layers of items, a meta-path consists of at most one item from each layer.*

For every meta-path $p = i_1 \leftrightarrow i_2 \ldots \leftrightarrow i_k$, we compute the meta-path-based similarity s_p, weighted by its significance value, as follows.

$$s_p = \frac{\sum_{t=1}^{t=k-1} S_{i_t,i_{t+1}} \cdot s_{ac}(i_t, i_{t+1})}{\sum_{t=1}^{t=k-1} S_{i_t,i_{t+1}}}$$

For each pair of items (i,j) from different domains, if i, j are not connected directly, we aggregate the path similarities of all meta-paths between i and j. Due to the different lengths and similarities for meta-paths, we give different weights to different meta-paths. Shorter meta-paths produce better similarities in recommenders [27, 34] and hence are preferred over longer ones. We now explain the scheme behind assigning these weights and thereby computing the X-SIM values.

DEFINITION 4 (NORMALIZED WEIGHTED SIGNIFICANCE). *Given any pair of items i and j, we define normalized weighted significance ($\widehat{S}_{i,j}$) between i and j as their significance value weighted by the inverse of number of users rating either i or j. Formally, we denote normalized weighted significance as follows.*

$$\widehat{S}_{i,j} = \frac{S_{i,j}}{n(Y_i \cup Y_j)}$$

Next, we determine the notion of *path certainty* (c_p) of a meta-path to take into account the factor of varying path lengths. Path certainty measures how good a path is for the similarity computations.

DEFINITION 5 (PATH CERTAINTY). *Given any meta-path ($p = i_1 \leftrightarrow i_2 \ldots \leftrightarrow i_k$), we compute the path certainty (c_p) of the meta-path p as the product of the normalized weighted significance between each consecutive pair of items in the path p. Formally, we define the path certainty as follows.*

$$c_p = \prod_{t=1}^{t=k-1} \widehat{S}_{i_t,i_{t+1}}$$

It is important to note that the product of the normalized weighted significance values inherently incorporates the path length in our path certainty metric. Hence, shorter paths have higher weights compared to longer ones. Finally, we define our X-SIM metric as follows.

DEFINITION 6 (X-SIM). *Let $P(i,j)$ denote the set of all meta-paths between items i and j. We define the* X-SIM *for the item pair (i,j) as the path similarity weighted by the path certainty for all paths in $P(i,j)$. Formally, we define* X-SIM *for any given pair of items i and j as follows.*

$$\text{X-SIM}(i,j) = \frac{\sum_{p\in P(i,j)} c_p \cdot s_p}{\sum_{p\in P(i,j)} c_p}$$

Here, X-SIM(i,j) denotes the meta-path-based heterogeneous similarity between any two items i and j. X-SIM is then utilized to build the artificial profiles for users (*AlterEgos*). Note that a trivial transitive closure over similarities would not take into account the above-mentioned factors, which would in turn impact the heterogeneous similarities and consequently the recommendation quality.

4. RECOMMENDATION

We show in this section how to leverage our X-SIM metric to generate artificial (*AlterEgo*) profiles of users in domains where these users might not have any activity yet. For pedagogical reasons, we first present the non-private (NX-MAP) scheme, and then the extensions needed for the private (X-MAP) one.

4.1 Similarity Computation Phase

In this phase, X-MAP treats both the source and target domains as a single aggregated domain in order to compute pairwise item similarities, called *baseline* similarities. Basically, X-MAP computes the adjusted cosine similarities between the items in $I^S \cup I^T$ based on the preferences of the users in $\mathcal{U}^S \cup \mathcal{U}^T$ for these items. We distinguish the following two types of similarities:

(a) *Homogeneous similarities* are computed between items in the same domain. Such similarities are used for intra-domain extensions in § 5.

(b) *Heterogeneous similarities* are computed between items in different domains. Such similarities are used for cross-domain extensions in § 5.

4.2 X-SIM Computation Phase

After the computation of the baseline item-item similarities, X-MAP uses the X-SIM metric within a single domain to extend the connections between the bridge items of a

[5] This concept is analogous to statistical significance used in hypothesis testing.

domain and other items within the same domain. Then, X-MAP uses the X-SIM metric to extend the similarities between items across domains (we come back to this in more details in § 5). After the heterogeneous similarity extension, each item in source domain ($\mathcal{D}^S$) has a corresponding set of items in target domain ($\mathcal{D}^T$) with quantified (positive or negative) X-SIM values.

4.3 AlterEgo Generation Phase

In this phase, the profile of Alice (in $\mathcal{D}^S$) is mapped to her AlterEgo profile (in $\mathcal{D}^T$) as shown in Figure 3. We first present the non-private case, and then discuss the private one.

NX-MAP AlterEgo generation. The non-private mapping is performed in two steps.

Replacement selection. In this step, for every item i in $\mathcal{D}^S$, we determine the replacement item j in $\mathcal{D}^T$. Here, j is the heterogeneous item which is most similar to i based on the heterogeneous similarities computed using X-SIM.

AlterEgo profile construction. We then replace every item rated by Alice in $\mathcal{D}^S$ with the most similar item in $\mathcal{D}^T$ computed in the previous step. This item replacement induces the AlterEgo profile [6] of Alice in the target domain as shown in Figure 3.

This AlterEgo profile of Alice is the mapped profile of Alice from the source domain to the target domain. Note that the AlterEgo profiles could be incrementally updated to avoid re-computations in X-MAP.

Figure 3: Alice's AlterEgo profile (in target domain) mapped from her original profile (in source domain).

X-MAP AlterEgo generation. We now explain how we achieve the differentially private mapping.

Private replacement selection. We apply an obfuscation mechanism, depending on the meta-path-based heterogeneous similarities, to design our differentially private replacement selection technique (Algorithm 3). Note that standard differentially private techniques consisting, for example, in adding noise based on Laplace or Gaussian distributions would not work here for they would not build a profile consisting of items in the target domain. The following theorem conveys our resulting privacy guarantee.

THEOREM 1. *Given any item t_i, we denote the global sensitivity of* X-SIM *by GS and the similarity between t_i and any arbitrary item t_j by* X-SIM(t_i, t_j). *Our Private Replacement Selection (PRS) mechanism, which outputs t_j as the replacement with a probability proportional to $exp(\frac{\epsilon \cdot \text{X-SIM}(t_i,t_j)}{2 \cdot GS})$, ensures ϵ-differential privacy.*

Due to space constraints, the full proof is provided in our companion technical report [1] for interested readers.

[6] If Alice has rated a few items in $\mathcal{D}^T$, then the mapped profile is appended to her original profile in $\mathcal{D}^T$ to build her AlterEgo profile.

Algorithm 3 *Private Replacement Selection Algorithm: PRS(t_i,$I(t_i)$,X-Sim$(I(t_i)))$ where $I(t_i)$ is the set of items in the target domain with* X-SIM *values.*

Input: $\epsilon, t_i, I(t_i)$, *X-Sim*$(I(t_i))$ ▷ ϵ : Privacy parameter
1: Global sensitivity for X-SIM:
2: GS = |X-SIM$_{max}$ − X-SIM$_{min}$| = 2
3: **for** item t_j in $I(t_i)$ **do**
4: Allocate probability as:

$$\frac{exp(\frac{\epsilon \cdot \text{X-SIM}(t_i,t_j)}{2 \cdot GS})}{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot \text{X-SIM}(t_i,t_k)}{2 \cdot GS})}$$

5: **end for**
6: Sample an element t from $I(t_i)$ according to their probability.
7: **return:** t; ▷ ϵ-differentially private replacement for t_i

AlterEgo profile construction. In this step, we replace every item rated by Alice in $\mathcal{D}^S$ with the item in $\mathcal{D}^T$ returned by the PRS mechanism in the previous step. This item replacement scheme produces a private AlterEgo profile of Alice in the target domain.

Note that this private AlterEgo profile protects the privacy of the straddlers, users who rated in both the domains, as the ratings of these users are used to compute the heterogeneous similarities leaving their privacy at risk [27]. In addition, if the application domains are typically owned by different companies like Netflix and Last.fm, then this mechanism aids the exchange of AlterEgo profiles while preventing curious or malicious users to infer the preferences of others.

4.4 Recommendation Phase

We now present the main steps of our recommendation scheme. Again, we first explain the non-private case followed by the private one.

NX-MAP recommendation. The AlterEgo profile of Alice is used along with the original profiles in the target domain to compute the top-k similar users for Alice and then compute recommendations for Alice leveraging the profiles of the k most similar users from the target domain as shown in Algorithm 1. The item-based version of X-MAP utilizes this AlterEgo profile and computes the recommendations as demonstrated in Algorithm 2.

Furthermore, the AlterEgo profile in the target domain also retains the temporal behaviour [12] of the user in the source domain due to the item-to-item mapping. We incorporate this temporal behaviour in the item-based version of X-MAP by adding a time-based weight to the ratings to improve the recommendation quality further. The predictions (Equation 4), weighted by the time-based parameter (α), for the ratings are computed as follows.

$$Pred[i](t) = \bar{r}_i + \frac{\sum_{j \in N_i} \tau(i,j) \cdot (r_{A,j} - \bar{r}_j) \cdot e^{-\alpha(t - t_{A,j})}}{\sum_{j \in N_i} |\tau(i,j)| \cdot e^{-\alpha(t - t_{A,j})}} \quad (7)$$

Note that the prediction has a time-based relevance factor ($e^{-\alpha(t-t_{A,j})}$) with a decaying rate controlled by the parameter α to determine each rating's weight for the prediction computation. Here, $t_{A,j}$ denotes the *timestep* [7] when user A rated the item j. This specific time-based CF technique is applicable to the item-based CF approach as the prediction computation (Equation 7) for a user A is dependent only on her previous ratings for similar items and thereby leverages

[7] The timestep is a logical time corresponding to the actual timestamp of an event.

time as observed by A.

X-Map recommendation. The private AlterEgo profile of Alice is used along with the original profiles in the target domain to compute the recommendations for Alice. To demonstrate the adaptability of our heterogeneous recommender, the recommendation step is integrated with a differentially private approach, inspired by [39, 40], to protect the privacy of users in the target domain against other curious users. We implemented both item-based and user-based versions of X-Map. The item-based recommendation mechanism is demonstrated in Algorithm 5 which utilizes the PNSA mechanism (Algorithm 4). We first present our similarity-based sensitivity, required for the private approach, along with its correctness proof sketch. [8]

DEFINITION 7 (Local Sensitivity). *For any given function $f : \mathcal{R} \rightarrow \mathcal{R}$ and a dataset D, the Local Sensitivity of f is defined as $LS_f = \max_{D'} \|f(D) - f(D')\|_1$, where D and D' are neighboring datasets which differ at one user profile.*

We define a rating vector $r_{t_i} = [r_{t_{ai}}, ..., r_{t_{xi}}, r_{t_{yi}}]$ which consists of all the ratings for an item $t_i \in D$. Similarly, we define a rating vector r'_{t_i} for $t_i \in D'$. Since we use adjusted-cosine for X-Sim, a rating $r_{t_{xi}}$ is the result after subtracting the average rating of user x ($\bar{r}_x$) from the actual rating provided by x for an item i. The similarity-based sensitivity is formulated as follows.

THEOREM 2 (Similarity-based sensitivity). *Given any score function $q : \mathcal{R} \rightarrow R$ and a dataset D, we formulate the similarity-based sensitivity corresponding to a score function $q_i(I, t_j)$ for a pair of items t_i and t_j as:*

$$SS(t_i, t_j) = max\{max_{u_x \in \mathcal{U}_{ij}}(\frac{r_{t_{xi}} \times r_{t_{xj}}}{\| r'_{t_i} \| \times \| r'_{t_j} \|}),$$
$$max_{u_x \in \mathcal{U}_{ij}}(\frac{r_{t_i} \cdot r_{t_j}}{\| r'_{t_i} \| \times \| r'_{t_j} \|} - \frac{r_{t_i} \cdot r_{t_j}}{\| r_{t_i} \| \times \| r_{t_j} \|})\}$$

The full proof is provided in our companion technical report [1] for interested readers.

We use the notion of *truncated similarity* [39, 40] (Step 7 in Algorithm 4) along with our similarity-based sensitivity to enhance the quality of selected neighbors. The two theorems which prove that this truncated similarity along with our similarity-based sensitivity can enhance the quality of neighbors are as follows.

THEOREM 3. *Given any item t_i, we denote its k neighbors by $N_k(t_i)$, the maximal length of all the rating vector pairs by $|v|$, the minimal similarity among the items in $N_k(t_i)$ by $Sim_k(t_i)$ and the maximal similarity-based sensitivity between t_i and other items by SS. Then, for a small constant $0 < \rho < 1$, the similarity of all the items in $N_k(t_i)$ are larger than $(Sim_k(t_i) - w)$ with a probability at least $1 - \rho$, where $w = min(Sim_k(t_i), \frac{4k \times SS}{\epsilon'} \times ln\frac{k \times (|v|-k)}{\rho})$.*

Intuitively, Theorem 3 implies that the selected neighbors have similarities greater than some threshold value $(Sim_k(t_i) - w)$ with a high probability $(1 - \rho)$.

THEOREM 4. *Given any item t_i, for a small constant $0 < \rho < 1$, all items with similarities greater than $(Sim_k(t_i) + w)$ are present in $N_k(t_i)$ with a probability at least $1 - \rho$ where $w = min(Sim_k(t_i), \frac{4k \times SS}{\epsilon'} \times ln\frac{k \times (|v|-k)}{\rho})$.*

[8] Our similarity-based sensitivity is slightly different from the recommendation-aware one presented in [39, 40].

Intuitively, Theorem 4 implies that the items with similarities greater than some threshold value $(Sim_k(t_i) + w)$ are selected as neighbors with a high probability $(1 - \rho)$.

Both theorems prove that the truncated similarity along with our similarity-based sensitivity provides neighbors of good quality while providing $\epsilon'/2$-differential privacy. The predictions are computed leveraging the PNCF mechanism (Algorithm 5) which adds Laplacian noise to provide $\epsilon'/2$-differential privacy. By the composition property of differential privacy, PNSA and PNCF together provide ϵ'-differential privacy. The item-based version of X-Map includes the additional feature of *temporally relevant* predictions to boost the recommendation quality traded for privacy.

We provide here only two illustrations (temporal dynamics and differential privacy) of the adaptability of our heterogeneous recommender due to space constraints. Since the AlterEgo profile could be considered as an actual profile in the target domain, thereby any homogeneous recommendation algorithm [2] like Matrix Factorization techniques, can be applied in the target domain to generate the recommendations. We provide a demonstration regarding how to use Spark-MLlib's matrix factorization technique with X-Map in our GitHub repository [1].

Algorithm 4 *Private Neighbor Selection : PNSA(t_i,I,Sim(t_i)) where I is the set of all items.*

Input: $\epsilon', w, t_i, I, Sim(t_i)$, k ▷ ϵ' : Privacy

1: $C_1 = [t_j | s(t_i, t_j) \geq Sim_k(t_i) - w, t_j \in I]$
2: $C_0 = [t_j | s(t_i, t_j) < Sim_k(t_i) - w, t_j \in I]$
3: $w = min(Sim_k(t_i), \frac{4k \times SS}{\epsilon'} \times ln\frac{k \times (|v|-k)}{\rho})$
4: **for** N=1:k **do**
5: **for** item t_j in I **do**
6: $SS(t_i, t_j) = max\{max_{u_x \in \mathcal{U}_{ij}}(\frac{r_{t_{xi}} \times r_{t_{xj}}}{\| r'_{t_i} \| \times \| r'_{t_j} \|}),$ $max_{u_x \in \mathcal{U}_{ij}}(\frac{r_{t_i} \cdot r_{t_j}}{\| r'_{t_i} \| \times \| r'_{t_j} \|} - \frac{r_{t_i} \cdot r_{t_j}}{\| r_{t_i} \| \times \| r_{t_j} \|})\}$
7: $\widehat{Sim}(t_i, t_j) = max(Sim(t_i, t_j), Sim_k(t_i) - w)$
8: Allocate probability as: ▷ $\frac{\epsilon'}{2k}$-Privacy
$$\frac{exp(\frac{\epsilon' \cdot \widehat{Sim}(t_i,t_j)}{2k \times 2SS(t_i,t_j)})}{\sum_{l \in C_1} exp(\frac{\epsilon' \cdot \widehat{Sim}(t_i,t_l)}{2k \times 2SS(t_i,t_l)}) + \sum_{l \in C_0} exp(\frac{\epsilon' \cdot \widehat{Sim}(t_i,t_l)}{2k \times 2SS(t_i,t_l)})}$$
9: **end for**
10: Sample an element t from C_1 and C_0 without replacement according to their probability.
11: $N_k(t_i) = N_k(t_i) \cup t$
12: **end for**
13: **return:** $N_k(t_i)$;

5. IMPLEMENTATION

In this section, we describe our implementation of X-Map. Figure 4 outlines the four main components of our implementation: *baseliner*, *extender*, *generator* and *recommender*. We describe each of these components along with their functionality.

5.1 Baseliner

This component computes the baseline similarities leveraging the adjusted cosine similarity (Equation 3) between the items in the two domains. The baseliner splits the item-pairs based on whether both items belong to the same domain or not. If both items are from the same domain, then the item-pair similarities will be delivered as *homogeneous*

Algorithm 5 *Private Recommendation: PNCF(P_A, I) where P_A denotes the AlterEgo profile of Alice, and I denotes the set of all items.*

var P; ▷ Dictionary with predictions for Alice
var τ; ▷ User similarities
var $\bar{r}$; ▷ Average rating for each items
var ϵ' ▷ Degree of privacy
var N_k ▷ Private neighbors using PNSA
for t_i : *item* in P_A **do**
  $N_k(t_i) = PNSA(t_i, I, Sim(t_i))$
  for t_j : *item* in $N_k(t_i)$ **do**
    $P[t_j] = \bar{r}_{t_j} + \frac{\sum_{t_k \in N_k(t_j)} (\tau(t_k,t_j) + Lap(\frac{SS(t_k,t_j)}{\epsilon'/2})) \cdot (r_{A,t_k} - \bar{r}_{t_k})}{\sum_{t_k \in N_k(t_j)} |\tau(t_k,t_j) + Lap(\frac{SS(t_k,t_j)}{\epsilon'/2})|}$
  end for
end for
$R_A = P.sortByValue(ascending = false)$;
return: $R_A[:N]$; ▷ Top-N recommendations for Alice

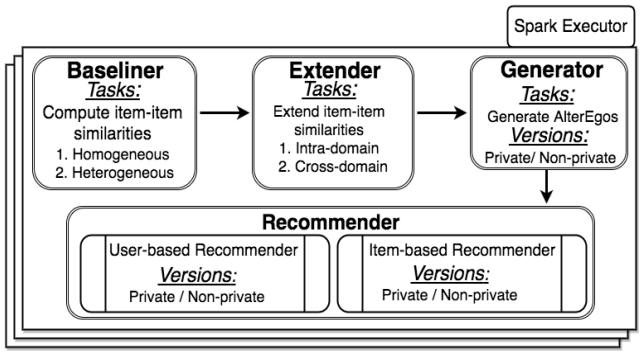


Figure 4: The components of X-MAP: *Baseliner, Extender, Generator, Recommender.*

similarities. If one of the items belongs to a different domain, then the item-pair similarities will be delivered as *heterogeneous similarities*. The baseline heterogeneous similarities are computed based on the user overlap. [9]

5.2 Extender

This component extends the baseline similarities both within a domain and across domains. The items in each domain are divided into three layers based on our layer-based pruning technique as shown in Figure 2. For every item in a specific layer, the extender computes the top-k similar items for the neighboring layers. For instance, for the items in the BB-layer of $\mathcal{D}^S$, the extender computes the top-k similar ones from items in the BB-layer in $\mathcal{D}^T$ and also the top-k similar ones from the items in the NB-layer in $\mathcal{D}^S$.

Intra-domain extension. In this step, the extender computes the X-SIM similarities for the items in the NN-layer in $\mathcal{D}^S$ and the items in the BB-layer of $\mathcal{D}^S$ via the NB-layer items of $\mathcal{D}^S$. Similar computations are performed for domain $\mathcal{D}^T$.

Cross-domain extension. After the previous step, the extender updates the NB and NN layers in both domains based on the new connections (top-k). Then, it updates the connections between the items in NB and BB layers in one domain and the items in NB and BB layers in the other one.

At the end of the execution, the extender outputs, for every item t_i in $\mathcal{D}^S$, a set of items $I(t_i)$ in $\mathcal{D}^T$ with some quantified (positive or negative) X-SIM values with t_i.

[9] These are the baseline similarities without any extension or enhancements.

5.3 Generator

The generator performs the following computational steps.

Item mapping. The Generator maps every item in one domain (say $\mathcal{D}^S$) to its most similar item (for NX-MAP) or its private replacement (for X-MAP) in the other domain ($\mathcal{D}^T$). After, the completion of this step, every item in one domain has a replacement item in the other domain. [10]

Mapped user profiles. The Generator here creates an artificial profile (AlterEgo) of a user in a target domain $\mathcal{D}^T$ from her actual profile in the source domain $\mathcal{D}^S$ by replacing each item in her profile in $\mathcal{D}^S$ with its replacement in $\mathcal{D}^T$ as shown in Figure 3. Finally, after this step, the Generator outputs the AlterEgo profile of a user in the target domain where she might have little or no activity yet.

5.4 Recommender

This component utilizes the artificial AlterEgo profile created by the Generator to perform the recommendation computation. It can implement any general recommendation algorithm for its underlying recommendation computation. In this paper, we implemented user-based and item-based CF schemes. For NX-MAP, the recommender uses Algorithm 1 (user-based CF) or Algorithm 2 (item-based CF) in the target domain. For X-MAP, the recommender also uses the PNSA algorithm along with the PNCF algorithm to generate recommendations either in a user-based manner or in an item-based manner. Additionally, for both NX-MAP and X-MAP, the item-based CF recommender leverages the temporal relevance to boost the recommendation quality. It is important to note that X-MAP runs periodically in an offline manner to update the predicted ratings. The top-10 items (sorted by the predicted ratings), not-yet-seen by the current user, would be recommended to users in X-MAP.

6. EXPERIMENTAL EVALUATION

In this section, we report on our empirical evaluation of X-MAP on a cluster computing framework, namely Spark [38], with real world traces from Amazon [25] to analyze its prediction accuracy, privacy and scalability. We choose Spark as our cluster computing framework since the underlying data processing framework to support X-MAP must be scalable and fault-tolerant.

6.1 Experimental Setup

Experimental platform. We perform all the experiments on a cluster of 20 machines. Each machine is an Intel Xeon CPU E5520 @2.26GHz with 32 GB memory. The machines are connected through a 2xGigabit Ethernet (Broadcom NetXtremeII BCM5716).

Datasets. We now provide an overview of the datasets used in our experiments.

Amazon. We use two sets of real traces from Amazon datasets [25]: *movies* and *books*. We use the ratings for the period from 2011 till 2013. The movies dataset consists of 1,671,662 ratings from 473,764 users for 128,402 movies whereas the books dataset consists of 2,708,839 ratings from 725,846 users for 403,234 books. The ratings vary from 1 to 5 with an increment of 1. The overlapping users in these two datasets are those Amazon users who are present in both

[10] We could also choose a set of replacements for any item, using X-SIM, in the target domain to have more diversity.

datasets and are ascertained using their Amazon customer-ids. The number of such overlapping users from both the domains is 78,201.

Movielens. We use the Movielens dataset (ML-20M) for evaluating performance of X-MAP within a single domain. This dataset consists of 20,000,263 ratings from 138,493 users for 27,278 movies. In this dataset, the ratings also vary from 1 to 5 with an increment of 1.

Evaluation metrics. We evaluate X-MAP along three complementary metrics: (1) the recommendation *quality* as perceived by the users in terms of prediction *accuracy*, (2) the degree of *privacy* provided to the end-users in terms of the privacy parameters (ϵ, ϵ'), and (3) the *scalability* in terms of *speedup* achieved in X-MAP when increasing the number of machines in the cluster.

Accuracy. We evaluate the accuracy of the predictions in terms of Mean Absolute Error (MAE). MAE computes the average absolute deviation between a predicted rating and the user's true rating. MAE is a standard metric used to evaluate state-of-the-art recommenders [16, 31]. We assume that the predicted rating for an item i is denoted by p_i and the actual rating is denoted by r_i in the test dataset. Then, the MAE for a test dataset, with $\mathcal{N}$ ratings, is computed as: $\frac{\sum_{i=1}^{\mathcal{N}} |p_i - r_i|}{\mathcal{N}}$. Given that r_{min} and r_{max} denotes the minimum and maximum ratings respectively, the following inequality always holds: $0 < MAE < (r_{max} - r_{min})$. The lower the MAE, the more accurate the predictions.

Privacy. Our differential privacy guarantees are parameterized as follows: ϵ for the PRS technique (Algorithm 3) used for AlterEgo generation and ϵ' for the PNCF (Algorithm 5) used for the private recommendation generation in X-MAP. According to the privacy literature [13, 39, 40], $\epsilon = 1$ or less would be suitable for privacy preserving purposes.

Speedup. We evaluate the speedup in terms of the time required for sequential execution (T_1) and the time required for parallel execution with p processors (T_p). Amdahl's law models the performance of speedup (S_p) as follows.

$$S_p = \frac{T_1}{T_p}$$

Due to the considerable amount of computations for heterogeneous recommendation on the Amazon dataset, we compare the speedup on p processors with respect to a minimum of 5 processors (T_5) instead of a sequential execution (T_1).

Competitors. We now present the recommenders against which we compare X-MAP. Existing recommendation schemes over several domains can be classified as follows.

Linked-domain personalization. The goal here is to recommend items in the target domain ($\mathcal{D}^T$) by exploring rating preferences aggregated from both source and target domains, i.e, to recommend items in $\mathcal{I}^T$ to users in $\mathcal{U}^S$ based on the preferences of users in $\mathcal{U}^S \cup \mathcal{U}^T$ for items in $\mathcal{I}^S \cup \mathcal{I}^T$. In this approach, ratings from multiple domains are aggregated into a single domain. Then, a traditional CF mechanism can be applied over this aggregated single domain [11, 29]. ITEM-BASED-KNN is a linked-domain personalization approach [11, 29] where we use item-based collaborative filtering over the aggregated ratings from both the domains.

Heterogeneous recommendation. The goal here is to recommend items in $\mathcal{I}^T$ to users in $\mathcal{U}^S$ based on the preferences of $\mathcal{U}^S$ for $\mathcal{I}^S$, $\mathcal{U}^T$ for $\mathcal{I}^T$ and $\mathcal{U}^S \cap \mathcal{U}^T$ for $\mathcal{I}^S \cup \mathcal{I}^T$. In this approach, the user similarities are first computed in both source and target domains. These domain-related similarities are then aggregated into the overall heterogeneous similarities. Finally, the k-nearest neighbors, used for recommendation computations, are selected based on these heterogeneous similarities [6]. In the REMOTEUSER approach [6], the user similarities in source domain are used to compute the k nearest neighbors for users who have not rated in the target domain. Finally, user-based collaborative filtering is performed.

Baseline prediction. For a sparse dataset, the baseline is provided by item-based average ratings [5] or user-based average ratings [22]. The goal here is to predict based on the average ratings provided by users in $\mathcal{U}^S \cup \mathcal{U}^T$ for items in $\mathcal{I}^S \cup \mathcal{I}^T$. One of the most basic prediction schemes is the ITEMAVERAGE scheme where we predict that each item will be rated as the average over all users who rated that item [5]. Note that though this technique gives a very good estimate of the actual rating but it is not personalized due to same predictions for all the users.

We compare X-MAP with these three other systems namely: ITEM-BASED-KNN, REMOTEUSER and ITEMAVERAGE.

Evaluation scheme. We partition the set of common users who rated both movies and books into *training* and *test* sets. For the test users, we hide their profile in the target domain (say books) and use their profile in the source domain (movies) to predict books for them. This strategy evaluates the accuracy of the predictions if the user did not rate any item in the target domain. Hence, we can evaluate the performance of X-MAP in the scenario where the test users did not rate any item in the target domain (cold-start). Additionally, if we hide part of the user profile in the target domain, then we can evaluate how X-MAP handles the scenario where the test users rated very few items in the target domain (sparsity). Furthermore, we denote the item-based variant of X-MAP as X-MAP-IB and the user-based variant as X-MAP-UB. Similarly for NX-MAP, we denote the item-based variant of NX-MAP as NX-MAP-IB and the user-based variant as NX-MAP-UB.

6.2 Temporal Dynamics

In this section, we observe the temporal effect of users, retained by the AlterEgos across domains, in X-MAP. We leverage the item-based recommender, and tune the temporal parameter α accordingly. Figure 5 demonstrates this temporal relevance effect where α varies between 0 (no temporal effect) to 0.2. Note that an item-based CF approach computes the predictions leveraging the target user's very few observed ratings on the nearest neighbors and given the very limited size of this set of ratings, any further amplification of α impacts the predictions negatively as it reduces the contribution of old ratings furthermore. We provide the optimally tuned parameter (α_o) for our experiments, shown in Figure 5, to achieve optimal recommendation quality.

6.3 Privacy

In this section, we tune the privacy parameters (ϵ, ϵ') for X-MAP. Figures 6 and 7 demonstrate the effect of tuning the privacy parameters on the prediction quality in terms of MAE. We observe that the recommendation quality improves (lower MAE) as we decrease the degree of privacy (higher ϵ, ϵ'). It is important to note that X-MAP inherently transforms to NX-MAP as the privacy parameters increase furthermore (lower privacy guarantees). For the following experiments, we select the privacy parameters as follows.

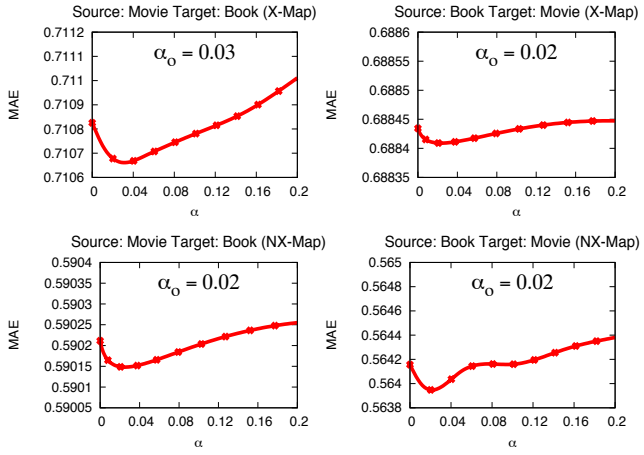


Figure 5: Temporal relevance (X-MAP, NX-MAP).

For X-MAP-IB, we select $\epsilon = 0.3$ and $\epsilon' = 0.8$. For X-MAP-UB, we select $\epsilon = 0.6$ and $\epsilon' = 0.3$. [11]

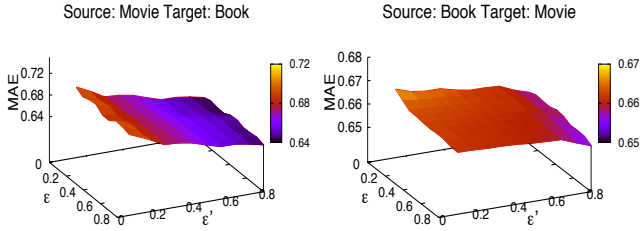


Figure 6: Privacy-quality trade-off in X-MAP-IB.

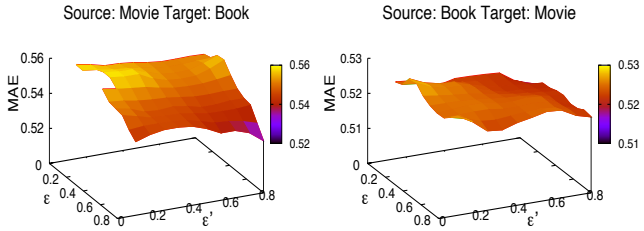


Figure 7: Privacy-quality trade-off in X-MAP-UB.

6.4 Accuracy

We now compare the accuracy of the predictions of X-MAP and NX-MAP with the competitors.

Impact of top-k neighbors. First, we evaluate the quality in terms of MAE when the size of k (neighbors in Equation 7) is varied. Figure 8(a) demonstrates that X-MAP-UB and NX-MAP-UB outperform the competitors by a significant margin of 30% where the source domain is book and the target domain is movie. Also, Figure 8(b) shows that X-MAP performs better than the non-private competitors whereas NX-MAP again outperforms the competitors by a margin of 18% where the source domain is movie and the target domain is book. A higher number of neighbors induces more connections across the domains (Figure 2) and hence enables X-MAP to explore better meta-paths between items. Moreover, better meta-paths lead to better meta-path based similarities and thereby superior recommendation quality. We consider k as 50 for all further experiments.

[11] These parameters are selected from a range of possible values providing quality close to the optimal one as observed from Figures 6 and 7.

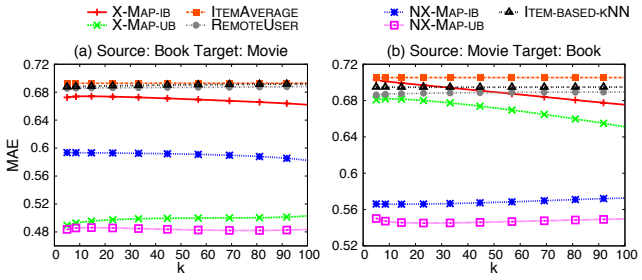


Figure 8: MAE comparison with varying k.

Impact of overlap. We now evaluate how X-MAP and NX-MAP perform when the number of users in the overlap increases. Intuitively, a good approach should provide better accuracy as more and more users connect the domains. These increasing connections improve the baseline heterogeneous similarities which are then leveraged by X-SIM to generate better meta-path based similarities across the domains. Figure 9 shows that the prediction error of X-MAP decreases as there are more users connecting the domains. This observation demonstrates that the quality of the AlterEgo profiles improves when the overlap size increases. Furthermore, we observe in Figure 9(a) that the user-based models show more improvement than the item-based ones. This behaviour occurs as the item similarities are more static than the user similarities [19].

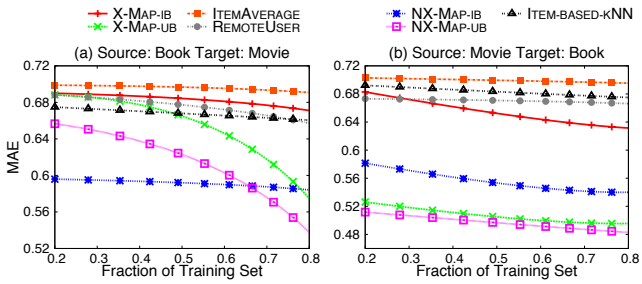


Figure 9: MAE comparison (Overlap size). [12]

Impact of sparsity. We now evaluate how X-MAP performs when the size of the training profile of a user, in the target domain, increases from a minimum of 0 (cold-start situation) to a maximum of 6 (low sparsity), in addition to her profile in the source domain [13]. This experiment also highlights the performance of X-MAP when the sparsity of the dataset decreases. Additionally, we evaluate the accuracy improvement of X-MAP over a single domain solution, *item-based kNN in the target domain* denoted by KNN-SD, as well as over a heterogeneous solution, *item-based kNN in the aggregated domain* denoted by KNN-CD. Figure 10 demonstrates that KNN-SD and KNN-CD are outperformed by NX-MAP and X-MAP. Furthermore, we observe a relatively fast improvement for our non-private item-based technique (NX-MAP-IB) due to the improvement in item similarities with lower sparsity.

6.5 Homogeneity

We now evaluate the ability of X-MAP to be applied to a homogeneous setting consisting of a single domain. Depend-

[12] Training set size denotes overlap size.

[13] We consider only those users who rated at least 10 products in each domain.

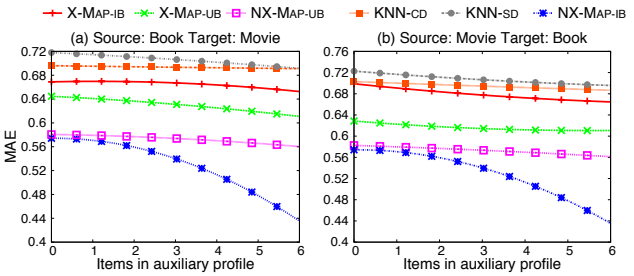


Figure 10: MAE comparison based on profile size.

ing on the structural property of the data (e.g., genres), any domain could be partitioned into multiple sub-domains. For this experiment, we use the ML-20M dataset which consists of 19 different genres. We partition this dataset into two sub-domains D_1 and D_2 by sorting the genres based on the movie counts per genre and allocating alternate sorted genres to the sub-domains as shown in Table 2. Note that a movie can have multiple genres. If a movie m belongs to both the sub-domains, we add it to the sub-domain which has the most number of genres overlapping with m's set of genres and to any of the two sub-domains in case of equal overlap with both sub-domains. Sub-domain D_1 consists of $15,119$ movies with $138,492$ users whereas sub-domain D_2 consists of $11,383$ movies with $138,483$ users.

Table 2: Sub-domains (D_1 and D_2) based on genres in Movielens 20M dataset.

D_1		D_2	
Genres	**Movie counts**	**Genres**	**Movie counts**
Drama	13344	Comedy	8374
Thriller	4178	Romance	4127
Action	3520	Crime	2939
Horror	2611	Documentary	2471
Adventure	2329	Sci-Fi	1743
Mystery	1514	Fantasy	1412
War	1194	Children	1139
Musical	1036	Animation	1027
Western	676	Film-Noir	330
Other	196	–	–

We compare X-MAP and NX-MAP with Alternating Least Square from MLLIB (MLLIB-ALS). We observe from Table 3 that NX-MAP significantly outperforms MLLIB-ALS whereas X-MAP, even with the additional privacy overhead, almost retains the quality of non-private MLLIB-ALS.

Table 3: MAE comparison (homogeneous setting on ML-20M dataset).

	NX-MAP	X-MAP	MLLIB-ALS
MAE	**0.6027**	0.6830	0.6729

6.6 Scalability

In this section, we evaluate the scalability of X-MAP in terms of the speedup achieved with an increasing number of computational nodes. We also compare our scalability with a state-of-the-art homogeneous recommender leveraging Spark to implement *Alternating-Least-Squares* based matrix factorization (MLLIB-ALS). For the ALS recommender, we use the aggregated ratings over both the domains (linked-domain personalization). Figure 11 demonstrates the near-linear speedup of X-MAP. Additionally, we see that X-MAP outperforms the scalability achieved by MLLIB-ALS. Note that X-MAP is periodically executed offline and the computation time for the recommendations, corresponding to all the users in the test set, is around 810 seconds on 20 nodes.

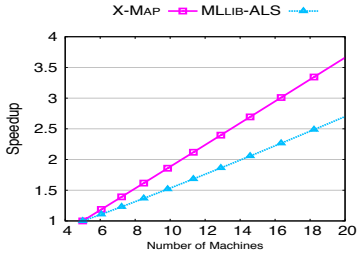


Figure 11: Scalability of X-MAP.

6.7 Online Deployment

We deployed an online recommendation platform (http://x-map.work/) leveraging X-SIM and made it available to users. We observe that this recommender indeed provides book recommendations like *Shutter Island: A Novel* when the user queries for the movie *Inception*. Besides, it also recommends the *Shutter Island* movie as a homogeneous recommendation. We observe similar results for multiple other queries.

7. RELATED WORK

Heterogeneous trends. Research on heterogeneous recommendation is relatively new. There are, however, a few approaches to tackle the problem which we discuss below.

Smart User Models. González et al. introduced the notion of Smart User Models (SUMs) [15]. The idea is to aggregate heterogeneous information to build user profiles that are applicable across different domains. SUMs rely on users' emotional context which are, however, difficult to capture. Additionally, it has been shown that users' ratings vary frequently with time depending on their emotions [4].

Web Monitoring. Hyung et al. designed a web agent which profiles user preferences across multiple domains and leverages this information for personalized web support [21]. Tuffield et al. proposed Semantic Logger, a meta-data acquisition web agent that collects and stores any information (from emails, URLs, tags) accessed by the users [36]. However, web agents are considered a threat to users' privacy as users' data over different e-commerce applications are stored in a central database administered by the web agent.

Cross-domain Mediation. Berkovsky et al. [6] proposed the idea of cross-domain mediation to compute recommendations by aggregating data from several recommenders. We showed empirically that X-MAP outperforms cross-domain mediation in Figures 8 and 9.

In contrast, X-MAP introduces a new trend in heterogeneous personalization in the sense that the user profile from a source domain is leveraged to generate an *artificial* AlterEgo profile in a target domain. The AlterEgo profiles can even be exchanged between e-commerce companies like Netflix, Last.fm thanks to the privacy guarantee in X-MAP.

Merging preferences. One could also view the heterogeneous recommendation problem as that of merging single-domain user preferences. Through this viewpoint, several approaches can be considered which we discuss below.

Rating aggregation. This approach is based on aggregating user ratings over several domains into a single multi-domain rating matrix [6, 7]. Berkovsky et al. showed that

this approach can tackle cold-start problems in collaborative filtering [7]. We showed empirically that X-MAP easily outperforms such rating aggregation based approaches [6].

Common representation. This approach is based on a common representation of user preferences from multiple domains either in the form of a *social tag* [35] or *semantic relationships between domains* [23]. Shi et al. developed a Tag-induced Cross-Domain Collaborative Filtering (TAGCDCF) to overcome cold-start problems in collaborative filtering [32]. TAGCDCF exploits shared tags to link different domains. They thus need additional tags to bridge the domains. X-MAP can bridge the domains based on the ratings provided by users using its novel X-SIM measure without requiring any such additional information which is difficult to collect in practice.

Linked preferences. This approach is based on linking users' preferences in several domains [11]. We showed empirically that X-MAP outperforms such linked preference based approaches [11] in Figures 8 and 9.

Domain-independent features. This approach is based on mapping user preferences to domain-independent features like *personality types* [8] or *user-item interactions* [24]. This approach again requires additional information like *personality scores* which might not be available for all users.

8. CONCLUDING REMARKS

We presented X-MAP, a scalable and private heterogeneous recommender. X-MAP leverages a novel similarity metric X-SIM, identifying similar items across domains based on meta-paths, to generate AlterEgo profiles of users in domains where these users might not have any activity yet. We demonstrated that X-MAP performs better in terms of recommendation quality than alternative heterogeneous recommenders [5, 6, 11]. (Although, not surprisingly, there is a trade-off between quality and privacy.)

Acknowledgement. This work was funded by Web-Alter-Ego Google Focused Award, and ERC grant for Adversary Oriented Computing under grant agreement number 339539.

9. REFERENCES

[1] X-Map GitHub repository. https://github.com/LPD-EPFL-ML/X-MAP.
[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In *TKDE*, pages 734–749, 2005.
[3] K. Ali and W. Van Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *SIGKDD*, pages 394–401, 2004.
[4] X. Amatriain, J. M. Pujol, and N. Oliver. I like it... i like it not: Evaluating user ratings noise in recommender systems. In *UMAP*, pages 247–258. 2009.
[5] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, 2009.
[6] S. Berkovsky, T. Kuflik, and F. Ricci. Cross-domain mediation in collaborative filtering. In *User Modeling*. 2007.
[7] S. Berkovsky, T. Kuflik, and F. Ricci. Distributed collaborative filtering with domain specialization. In *RecSys*, 2007.
[8] I. Cantador, I. Fernández-Tobías, A. Bellogín, M. Kosinski, and D. Stillwell. Relating personality types with user preferences in multiple entertainment domains. In *UMAP Workshops*, 2013.
[9] F. Carmagnola and F. Cena. User identification for cross-system personalisation. *Information Sciences*, 179(1):16–32, 2009.
[10] F. Carmagnola, F. Cena, and C. Gena. User model interoperability: a survey. In *UMUAI*, 2011.
[11] P. Cremonesi, A. Tripodi, and R. Turrin. Cross-domain recommender systems. In *ICDMW*, 2011.
[12] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM*, pages 485–492, 2005.
[13] C. Dwork. A firm foundation for private data analysis. *CACM*, 54(1):86–95, 2011.
[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.
[15] G. González, B. López, and J. L. de la Rosa. A multi-agent smart user model for cross-domain recommender systems. In *Beyond Personalization*, 2005.
[16] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.
[17] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, 1999.
[18] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *SIGMOD*, 2015.
[19] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.
[20] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. In *CIMCA*, 1999.
[21] H. J. Kook. Profiling multiple domains of user interests and using them for personalized web support. In *Advances in Intelligent Computing*, pages 512–520. Springer, 2005.
[22] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SIAM*, 2005.
[23] A. Loizou. *How to recommend music to film buffs*. PhD thesis, UNIVERSITY OF SOUTHAMPTON, 2009.
[24] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In *ECIR*. 2014.
[25] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.
[26] M. F. Rahman, W. Liu, S. Thirumuruganathan, N. Zhang, and G. Das. Privacy implications of database ranking. *PVLDB*, 8(10):1106–1117, 2015.
[27] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, (6):54–62, 2001.
[28] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW*, 1994.
[29] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.
[30] B. Shapira, L. Rokach, and S. Freilikhman. Facebook single and cross domain data for recommendation systems. In *UMUAI*, 2013.
[31] U. Shardanand and P. Maes. Social information filtering: algorithms for automating a word of mouth. In *SIGCHI*, 1995.
[32] Y. Shi, M. Larson, and A. Hanjalic. Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering. In *UMAP*, 2011.
[33] I. Soboroff and C. Nicholas. Collaborative filtering and the generalized vector space model. In *SIGIR*, 2000.
[34] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.
[35] M. N. Szomszor, I. Cantador, and H. Alani. Correlating user profiles from multiple folksonomies. In *Hypertext*, 2008.
[36] M. M. Tuffield, A. Loizou, and D. Dupplaw. The semantic logger: Supporting service building from personal context. In *CARPE*, 2006.
[37] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, 1998.
[38] M. Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, 2012.
[39] T. Zhu, G. Li, Y. Ren, W. Zhou, and P. Xiong. Differential privacy for neighborhood-based collaborative filtering. In *Asonam*, pages 752–759, 2013.
[40] T. Zhu, Y. Ren, W. Zhou, J. Rong, and P. Xiong. An effective privacy preserving algorithm for neighborhood-based collaborative filtering. *Future Generation Computer Systems*, 36:142–155, 2014.