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 are widely implemented nowadays by major e-commerce players like Netflix, Amazon or Last.fm. They typically make use of effective collaborative filtering schemes to select the most relevant items to suggest. However, most recommenders today are *homogeneous*: they focus on a specific domain (e.g., either movies or books). In short, Alice will only get recommended a movie if she has been rating only movies. Clearly, the multiplicity of web applications is calling for *heterogeneous* recommendations. Ideally, Alice should be able to get recommendations for books even if she has only rated movies.

In this paper, we present X-Map, a heterogeneous recommender addressing such issues. X-Map relies on X-Sim, a novel meta path-based similarity where a *meta path* is a path consisting of heterogeneous items (like movies, books). X-Sim computes the transitive closure of inter-item similarities over several domains based on users who rated across these domains. X-Sim is then used to generate a user profile (called *AlterEgo*) for a domain where the user might not have any activity yet. Furthermore, information aggregation across multiple domains also increases the potential privacy risk for users. X-Map employs differential privacy to address such privacy concerns.

We present a comprehensive experimental evaluation of X-Map using real traces from Amazon. Our evaluation shows that, in terms of recommendation quality, X-Map outperforms alternative heterogeneous recommenders, and in terms of throughput, X-Map scales linearly with increasing number of machines.

1. INTRODUCTION

The vast amount of information available in the Internet calls for *personalization* schemes [6], such as *recommenders*, that aid the users in their web navigation activities. Recommenders seek to suggest *relevant* items to users based on their interests, by typically relying on *Collaborative Filtering* (CF) algorithms [27] to leverage similarities between users. Such similarities are computed using the past rating histories of users. The axiom here is that *if Alice and Bob have liked the same items in the past, they are likely to be interested in the same items in the future.*

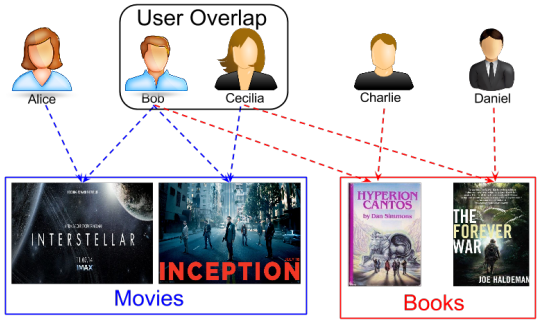


Figure 1: User overlap (Bob, Cecilia) with heterogeneous personalization.

1.1 Motivation

However, although widely used today, recommenders are mainly applied in a *homogeneous* sense. Movie recommenders like IMDb or Netflix, news recommenders like Google News or Yahoo News, as well as music recommenders like Last.fm and Spotify, each focuses on a specific domain. In short, you will be recommended books if you rated books, and you will be recommended movies if you rated movies. Given the growing multiplicity of web applications, as well as the amount of information available in the Internet, homogeneity is a major limitation. In short, 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 and news.

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 domains, can be leveraged to improve personalization, not only for users who are new to a given domain (*cold-start* situation), but also when the data is *sparse* [1] (e.g, very few ratings per user). In a heterogeneous personalization scheme, if a user, say Alice, liked *Interstellar*, then the system could recommend her books such as *The Forever War* by Joe Haldeman. The intuition is expressed in the scenario depicted in Figure 1 where five users rated *at most* one book. First, we make the following observations.

(1) Alice and Bob are like-minded as both like *Interstellar*.
(2) Bob and Cecilia are like-minded as both like *Inception*.
(3) Alice and Cecilia have no common interests.
(4) Alice and Cecilia are only related through Bob.

Based on the observations (1), (2) and (4), *The Forever War* could be recommended to Alice. Note that if we consider both domains as a single aggregated domain, this heterogeneous recommendation would be less likely due to the observation (3).

Heterogeneous personalization improves the quality of recommendation but it also raises several privacy concerns. For example, the new connection between Alice and Cecilia provides the opportunity for a curious user, say Alice, to discover the people (like Bob or Cecilia) who form a bridge between these two domains. She can actually probe to determine the item(s) that allows her to get this recommendation. The probing is done by masquerading as another user and incrementally rating items until the recommendation is made. By itself, her knowledge of such information poses no risk, but when used in conjunction with other information it fills a gap in knowledge that could be key to identifying the other user(s). This is similar to the privacy risk inherent in allowing statistical database queries where it is possible to make inferences from combinations of queries. Such privacy risks are amplified when the domains are owned by different companies (like IMDb for movies and Goodreads for books) which could lead to serious privacy violations.

1.2 Challenges

Building a simple heterogeneous recommender is straightforward by integrating ratings from both domains as a single domain. Following the aforementioned intuition (observation (3)), existing homogeneous personalization schemes [27–29], if applied as such over these aggregated ratings, lead to substandard recommendation quality due to decrease in overall *rating density*[1] as shown in Table 1. Hence, building an *efficient* heterogeneous recommender is not trivial. Furthermore, making it privacy-aware is even more challenging as we discuss below.

Books	Movies	Books+Movies
0.0204 %	0.0569 %	**0.0147 %**

Table 1: Densities for two domains in Amazon [24].

Similarity. Given some user overlap across domains, a major technical challenge is to compute the *similarity* between users in different domains. One might consider a graph-based similarity computation [11] 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 *path* is a sequence of adjacent vertices connected by edges in this graph. Although at first glance appealing, such an approach however ignores two factors crucial to collaborative filtering: *significance weighting* [15] and *path length* [26]. Significance weighting for an item-pair (i,j) reflects the importance of the number of users who either like both items or dislike both. For equal similarities, item-pairs with higher significance weights are more crucial. Paths with shorter length provide better recommendations due to the stronger ties between item-pairs on a path [26]. Homogeneous similarities have been improved based on the notion of significance weighting [15] as well as path length [26,33]. However, both factors have not been considered jointly which is crucial for accurate similarity computations in heterogeneous recommenders due to the decrease in rating density (as we show in § 3).

Scalability. Another technical challenge underlying the development of an effective heterogeneous recommender is *scalability*. Assuming m items in the recommender database, the time-complexity for a naive graph-based model is $O(m^2)$: this rapidly becomes a bottleneck with millions of items. For heterogeneous recommenders, the number of computations increases many-fold leading to higher computation time compared to homogeneous recommenders.

Privacy. As we pointed out before, *privacy* is also another technical challenge for heterogeneous personalization. The advent of *heterogeneous* applications increases the risk of leaking information about users. With heterogeneous applications potentially involving multiple companies, privacy is crucial: *user profile should not be revealed in clear*. In addition, privacy-preserving mechanisms to prevent curious users from making inferences about someone else's input based on their recommendations are needed. The situation is even more severe with *straddlers* (like Bob and Cecilia in Figure 1), users who connect multiple domains. Ramakrishnan et al. showed that such straddlers are at a privacy risk and information about their preference, towards items, could be used in conjunction with other data sources to uncover identities and reveal personal details [26].

1.3 Contributions

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

- X-Map leverages a novel similarity metric, X-Sim, which computes a meta path-based transitive closure of item-item similarities across several domains while supporting the two crucial factors: *significance weighting* and *path length*. X-Sim is leveraged to compute artificial profiles of users even in domains where they have *no* or *very little* activity yet.
- We capture these artificial profiles through our notion of *AlterEgo*. X-Map generates an *AlterEgo* profile (of Alice) in the target domain leveraging an item-to-item mapping from a source domain (e.g., movies) to a target domain (e.g., books). We present a private technique (in X-Map) as well as a non-private one (in NX-Map) for generating the AlterEgo profiles. Note that NX-Map illustrates the effectiveness of X-Sim and *AlterEgo* profiles for heterogeneous recommendations without the additional privacy overhead and could be used for applications with heterogeneous intra-company services like *Amazon* or *Google Play*. We demonstrate the flexibility of X-Map to integrate auxiliary recommendation features in the target domain by incorporating a differentially private approach inspired by Zhu et al. [39, 40] to provide the final recommendations. NX-Map significantly outperforms the quality of alternative heterogeneous systems [3,4,11,21,28] whereas X-Map (even with the additional privacy overhead) still provides better quality compared to the non-private alternatives.

[1] Rating density is defined as the fraction of collected ratings over all the possible ratings.

- We implement X-MAP on top of Apache Spark to achieve scalability and fault tolerance. We show that X-MAP scales almost linearly with an increasing number of machines (a major requirement for the deployment of a recommender in a practical environment).
- We deployed an online recommendation platform, leveraging X-SIM on a database of around 660K items, to recommend books and movies to users based on their *searched* product at[2]:

  http://x-map.work/

  We observe that it can recommend books like *The Da Vinci Code* when the search query is *Angels & Demons* (2009) movie.

1.4 Roadmap

The rest of the paper is structured as follows. We provide some preliminaries on collaborative filtering before formulating the heterogeneity problem for recommenders in § 2. Next, we introduce our X-SIM metric in § 3. We distinguish the private recommendation scheme underlying X-MAP from the non-private one (NX-MAP) in § 4. Then, we present our scalable implementation of X-MAP in § 5. We present our experimental results in § 6 and then review the related work in § 7. Finally, we conclude our paper in § 8. For space limitations, we defer the detailed proofs to our companion technical report [] for interested readers.

2. PRELIMINARIES

In this section, we present some preliminaries on collaborative filtering before formulating the heterogeneous recommendation problem. Table 2 summarizes some notations we use in this paper.

2.1 Collaborative Filtering

Collaborative Filtering (CF) algorithms fall mainly in two categories: *memory-based* [7, 27, 32] and *model-based* [16, 18, 37]. Memory-based algorithms compute the *top-k* like-minded users for a given user (Alice), denoted as the *neighbors* of Alice, from the training database and make recommendations to Alice 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 have two advantages over model-based ones. First, new information, like recent ratings by neighbors, can be easily integrated in the recommender. Second, item contents or features need not be considered in the prediction, leading to less computations.

Among memory-based CF schemes, *neighbor-based CF*, namely *k nearest neighbor* (kNN) algorithms are more popular and widely used in practice [28]. A user-based CF scheme, depicted in Algorithm 1, computes the k neighbors of Alice with highest user-to-user similarities (Phase 1 in Algorithm 1) and then these neighbors' profiles are leveraged to compute the best recommendations for Alice (Phase 2 in Algorithm 1). An item-based CF scheme, 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). X-MAP currently supports both user-based and item-based CF schemes for providing recommendations to its users.

[2]Currently, we support only Chrome, Safari and Firefox browsers.

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_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
Y_i	the set of users who rated i
$\tau(u,v)$	the Pearson correlation between u and v

Table 2: Notations

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: $N_A = \tau_A.sortByValue(ascending = false)$;
7: **return:** $N_A[:k]$;

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

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

2.2 Problem Formulation

Without loss of generality, we assume 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 $\mathcal{U}^T$ overlaps, but $\mathcal{I}^S$ and $\mathcal{I}^T$ have no common items. This captures the most common heterogeneous personalization scenario in web companies nowadays, e.g. Amazon, E-bay. We now define the recommendation problem which we study in this paper as follows.

DEFINITION 1. *Given the source domain $\mathcal{D}^S$ and the target domain $\mathcal{D}^T$, the heterogeneous personalization problem consists in recommending items in $\mathcal{I}^T$ to users in $\mathcal{U}^S$*

[3]Normally, M_D is a sparse matrix. Hence, if u has not rated i, then we use the average rating of i as $r_{u,i}$ to complete M_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: $N_j = \tau_j.sortByValue(ascending = false)$;
7: **return:** $N_j[:k]$;

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

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

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 $\mathcal{I}^S \cup \mathcal{I}^T$. More precisely, we aim to recommend items in $\mathcal{I}^T$ to a user who rated a few items (sparsity) or no items (cold-start) in $\mathcal{I}^T$.

The scenario, underlying Figure 1, 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 rated items in $\mathcal{D}^S$ (e.g., movies).

3. X-SIM

In this section, we present X-SIM, our novel similarity computation algorithm designed for heterogeneous recommendation.

Overview. Consider a similarity graph G of which the vertices are the items and the edges are weighted by the similarities computed using the standard adjusted cosine similarity [28]. Typically, this metric can only identify a small number of item similarities between different domains. Figure 2 conveys this very fact that we can compute around 0.3 million heterogeneous similarities[4] using this standard item-item adjusted cosine similarity (ADCOS). Being sparse with few edges, G induces very few heterogeneous connections and results in substandard recommendation quality. Clearly, this standard similarity computation is not suitable for *heterogeneous* recommenders.

In X-MAP, we use a novel meta path-based algorithm to compute an extended similarity graph. We compute the similarity between items even if they are not directly connected, and only related through some meta paths. Figure 2 shows that the number of detected heterogeneous similarities is nearly an order of magnitude higher by using our meta path-based approach instead of ADCOS. (These freshly detected heterogeneous similarities lead to better recommendation quality as we show later in § 6.4.)

[4]Heterogeneous similarity denotes the similarity computed between items in different domains, e.g between a book and a movie.

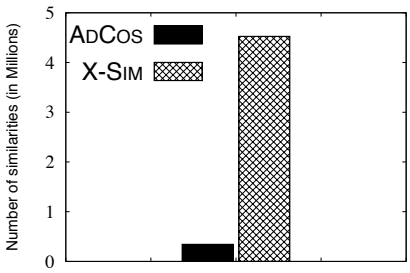


Figure 2: The number (millions) of inter-item similarities computed for the Amazon dataset using adjusted cosine similarity and X-SIM.

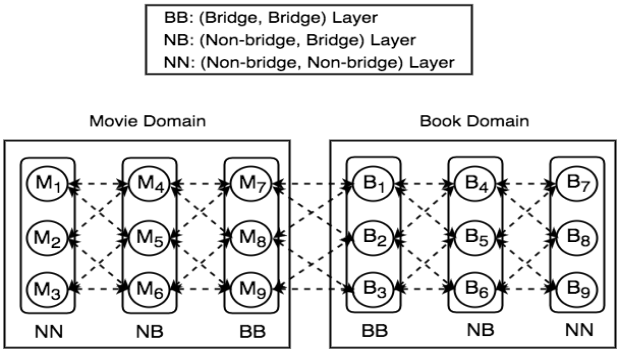


Figure 3: The layer-based pruning technique used in X-MAP.

Baseline Similarity Graph. Our first step is to build the baseline similarity graph. There are several known methods for computing item-item similarities, e.g. Cosine, Pearson, Adjusted-cosine [28]. Though X-SIM can support any similarity metric as its baseline, we use adjusted-cosine which is shown to be more effective than the others [28]:

$$s_{ac}(i,j) = \frac{\sum_{u \in Y_i \cap Y_j}(r_{u,i} - \bar{r_u})(r_{u,j} - \bar{r_u})}{\sqrt{\sum_{u \in Y_i}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{u \in Y_j}(r_{u,j} - \bar{r_u})^2}} \quad (5)$$

In this first step, we compute the (baseline) similarities by integrating both S and T as a single domain. We denote by G_{ac} the resulting similarity graph.[5] We address the sparsity issue of G_{ac} (Figure 2) by considering meta paths connecting both domains and then by extending G_{ac}. Clearly, a brute-force scheme considering all possible paths would be inefficient and not scalable. Assuming m items in the database, the time complexity of such a brute-force scheme (computing path similarity for every pair of items) is $O(m^2)$, which is not suitable for big datasets like the Amazon one. X-MAP uses a layer-based pruning technique to consider only $O(k)$ paths for each item where $k \lll m$. This pruning technique reduces the time complexity to $O(km) \simeq O(m)$.

Layer-based Pruning. Based on the baseline similarity graph, any item i in a domain D which connects to some item j in another domain D' is called a *bridge item*. Both i and j are bridge items in this case. Any item that is not a bridge item is called a *non-bridge item*.

[5]Here ac denotes adjusted cosine.

X-MAP's pruning technique partitions the items from S and T into six layers, as depicted in Figure 3. In turn, the items in each domain, say D, are divided into three layers (Figure 3).

- *BB-layer.* The (Bridge, Bridge)-layer consists of the bridge items of D connected to the bridge items of another domain.
- *NB-layer.* The (Non-bridge, Bridge)-layer consists of the non-bridge items of D which are connected to bridge items of D.
- *NN-layer.* The (Non-bridge, Non-bridge)-layer consists of the non-bridge items of D which are not connected to other bridge items.

X-MAP then considers only the paths crossing different layers denoted by *meta paths*. Since a k-nearest neighbor method is used for the recommendation, X-MAP chooses, for each item i, the top-k items from every neighboring layer. Let G be the resulting top-k similarity graph, which X-MAP treats as an undirected graph, i.e. if i is one of j's top-k neighbors or j is one of i's top-k neighbors then i and j are connected in G. Each node in G has thus a degree at most $2k$. We describe the meta path selection via layers in more details in § 5.

X-SIM. 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 it. 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 $n(Y_i)$ the cardinality of the set Y_i.

DEFINITION 2 (WEIGHTED SIGNIFICANCE). *Given a pair of items i and j, we define weighted significance ($S_{i,j}$) as the number of users who mutually agree or disagree on their preference for this pair. More precisely, we define the weighted significance ($S_{i,j}$) between i and j as follows.*

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

Intuitively, 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).

DEFINITION 3 (META PATH). *Given G and its six corresponding layers of items, we define a meta path as the path that contains at most one item from each layer.*

For every meta path $p = i_1 \leftrightarrow i_2 \ldots \leftrightarrow i_k$, we compute the path 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 connecting i and j. Due to the different lengths and similarities for paths, we have to give different weights to different paths. Shorter paths produce better similarities in recommenders [26, 33] and hence are more preferred over longer ones. We now explain the strategy behind assigning these weights and thereby computing the X-SIM values.

DEFINITION 4 (FRACTIONAL WEIGHTED SIGNIFICANCE). *Given a pair of items i and j, we define fractional weighted significance ($F_{i,j}$) between i and j as their significance value weighted by the inverse of number of users rating either i or j. More precisely, we denote fractional weighted significance as follows.*

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

Next, we define the notion of *path certainty* (c_p) of a meta path to take into account the factor of varying path lengths.

DEFINITION 5 (PATH CERTAINTY). *Given a 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 fractional weighted significance between each consecutive pair of items in the path p. More precisely, we define the path certainty as follows.*

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

Finally, we define our X-SIM metric as follows.

DEFINITION 6 (X-SIM). *Let $P(i,j)$ denote the set of all meta paths connecting 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 $P(i,j)$. More precisely, 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 heterogeneous similarity between any two items i and j. Note that a trivial transitive closure over similarities would not take into account these factors, which would in turn impact the heterogeneous similarities and 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 the users might not have any activity yet. We present both a privacy-preserving (X-MAP) and a non-privacy-preserving (NX-MAP) schemes.[6]

4.1 Similarity Computation Phase

We call the similarities computed in this phase as the *baseline* similarities. For this phase, X-MAP treats both the source and target domains as a single domain and computes all pairwise item 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 $U^S \cup U^T$ for these items. We distinguish two types of similarities which are as follows.

Homogeneous similarities. These similarities are computed between items in the same domain. For instance, the similarity between a book and another one is homogeneous. Such similarities are used for intra-domain extension in § 5.

[6]Recall that NX-MAP is used to demonstrate the effectiveness of X-SIM without the additional privacy overhead which can be useful for intra-company heterogeneous services like *Amazon* or *Google Play*.

Heterogeneous similarities. These similarities are computed between items in different domains. For instance, the similarity between a book and a movie is heterogeneous. Such similarities are used for cross-domain extension in § 5.

This phase remains the same for both X-MAP as well as NX-MAP.

4.2 X-SIM Computation Phase

After the computation of the baseline item-item similarities, X-MAP uses the X-SIM metric at first within a single domain to extend and improve the connections between the bridge items of a domain and other items within the same domain. Next, X-MAP uses the X-SIM metric to extend the similarities between items across the domains (which we explain in more details in § 5). At the completion of the heterogeneous similarity extension, for each item in source domain (D^S), there exists a corresponding set of items in target domain (D^T) with quantified (positive or negative) X-SIM values. This phase is the same for both X-MAP as well as NX-MAP.

4.3 AlterEgo Generation Phase

In this phase, the profile of Alice (in D^S) is mapped to her AlterEgo profile (in D^T) as shown in Figure 4. For pedagogical reasons, we first present the non-private case followed by the private one.

NX-MAP AlterEgo generation. The non-private mapping is performed in two steps which are as follows.

Similar item computation. In this step, for every item i in D^S, we determine the replacement item j in D^T. Here, j is the 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 D^S with the most similar item in D^T computed in the previous step. This item replacement induces the AlterEgo profile[7] of Alice in the target domain as shown in Figure 4.

This AlterEgo profile of Alice is the mapped profile of Alice from the source domain to the target domain.

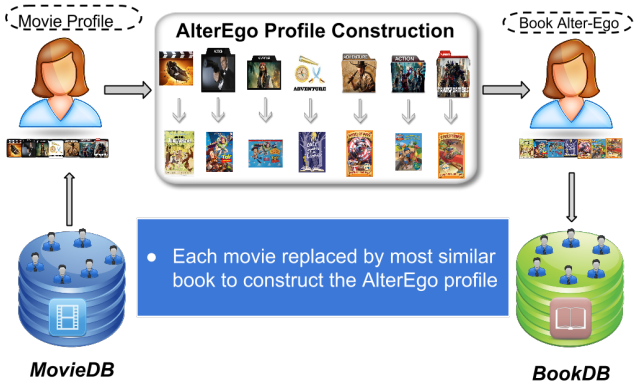


Figure 4: Alice's AlterEgo profile (in target domain) mapped from her original profile (in source domain).

X-MAP AlterEgo generation. The private mapping is performed in two steps which are as follows.

[7]If Alice has rated a few items in D^T then the mapped profile is appended to her original profile in D^T to build her AlterEgo profile.

Private replacement selection. We consider Differential privacy [13] which is a technique for releasing statistical information about a database without revealing information about its individual entries. More specifically, differential privacy is defined as follows.

DEFINITION 7 (DIFFERENTIAL PRIVACY [13]). *A randomized function $\mathcal{R}$ provides ϵ-differential privacy if for all datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, differing on at most one user profile, and all $t \in Range(\mathcal{R})$, the following inequality always holds:*

$$\frac{Pr[\mathcal{R}(\mathcal{D}_1) = t]}{Pr[\mathcal{R}(\mathcal{D}_2) = t]} \leq exp(\epsilon)$$

We leverage an exponential mechanism to design our differentially private replacement selection technique (Algorithm 3). The following theorem conveys our resulting privacy guarantee.

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

PROOF SKETCH. We now present a sketch of our proof. Consider two datasets D and D' which differ at one user, say u. We denote X-SIM (t_i,t_j) in dataset D by $q(D, t_i, t_j)$ and the set of items in target domain, with quantified X-SIM values, by $I(t_i)$. Furthermore, $q(D, I(t_i))$ denotes the set of X-SIM values between t_i and each item in $I(t_i)$. The global sensitivity (GS) is defined as $max_{D'}||q(D, t_i, t_j) - q(D', t_i, t_j)||_1$. Our PRS mechanism outputs an item t_j as a private replacement for t_i. Then, we have the following:

$$\frac{Pr[PRS(t_i, I(t_i), q(D, I(t_i))) = t_j]}{Pr[PRS(t_i, I(t_i), q(D', I(t_i))) = t_j]}$$

$$= \frac{exp(\frac{\epsilon \cdot q(D,t_i,t_j)}{2 \cdot GS})}{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot q(D,t_i,t_k)}{2 \cdot GS})} \div \frac{exp(\frac{\epsilon \cdot q(D',t_i,t_j)}{2 \cdot GS})}{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot q(D',t_i,t_k)}{2 \cdot GS})}$$

$$= \underbrace{\frac{exp(\frac{\epsilon \cdot q(D,t_i,t_j)}{2 \cdot GS})}{exp(\frac{\epsilon \cdot q(D',t_i,t_j)}{2 \cdot GS})}}_{\text{P}} \cdot \underbrace{\frac{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot q(D',t_i,t_k)}{2 \cdot GS})}{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot q(D,t_i,t_k)}{2 \cdot GS})}}_{\text{Q}}$$

$$P = exp(\frac{\epsilon \cdot (q(D, t_i, t_j) - q(D', t_i, t_j))}{2 \cdot GS}) \leq exp(\frac{\epsilon}{2})$$

$$Q = \frac{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot q(D',t_i,t_k)}{2 \cdot GS})}{\sum_{t_k \in I(t_i)} exp(\frac{\epsilon \cdot q(D,t_i,t_k)}{2 \cdot GS})} \leq exp(\frac{\epsilon}{2})$$

Therefore, we have:

$$\frac{Pr[PRS(t_i, I(t_i), q(D, I(t_i))) = t_j]}{Pr[PRS(t_i, I(t_i), q(D', I(t_i))) = t_j]} \leq exp(\epsilon)$$

Hence, PRS provides ϵ-differential privacy. □

AlterEgo profile construction. In this step, we replace every item rated by Alice in D^S with the item in D^T returned by the private mechanism in the previous step. This item replacement leads to an 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 [26]. In

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

addition, if the domains are typically two different applications 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 about the preferences of others.

4.4 Recommendation Phase

We now present the recommendation computation steps. 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 using Algorithm 1. The item-based version of X-Map leverages this AlterEgo profile and computes the recommendations as demonstrated in Algorithm 2. For the item-based version of X-Map, we improve the recommendation quality further by adding a time-based weight to the ratings. The predictions (Equation 4), weighted by the time-based parameter (α), for the ratings are computed as follows.

$$Pred[i](t) = \bar{r}_i + \frac{\sum_{j \in N_i} \tau(i,j) \cdot (r_{A,j} - \bar{r}_j) \cdot e^{-\alpha(t - t_{A,j})}}{\sum_{j \in N_i} |\tau(i,j)| \cdot e^{-\alpha(t - t_{A,j})}} \tag{6}$$

Note that the prediction has a time-based relevance factor ($e^{-\alpha(t-t_{A,j})}$) with decaying rate controlled by parameter α to determine each rating's weight for the prediction computation. Here, $t_{A,j}$ denotes the *timestep*[8] when user A rated item j. The time-based CF feature is applicable to the item-based CF approach as the prediction computation (Equation 6) for a user A is dependent only on her previous ratings for similar items and thereby leverages 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 flexibility of our heterogeneous recommender, the recommendation step is integrated with a differential private approach, inspired by Zhu et al. [39, 40], to protect 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 leverages the PNSA mechanism (Algorithm 4). We now present our modified recommendation-aware sensitivity along with its correctness proof sketch.[9]

Definition 8 (Local Sensitivity). *For a given function $f : \mathcal{R} \rightarrow R$ 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 the dataset D. Similarly, we have a rating vector r'_{t_i} for t_i in the neighboring dataset 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 provide by x for an item i. The recommendation-aware sensitivity is defined as follows.

Theorem 2 (Recommendation-aware sensitivity). *Given a score function $q : \mathcal{R} \rightarrow R$ and a dataset D, we define the recommendation-aware sensitivity corresponding to a score function $q_i(I, t_j)$ for a pair of items t_i and t_j as:*

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

Proof sketch. We now provide a proof sketch of the recommendation-aware sensitivity which is measured by the maximal change in the similarity between two items when deleting a user's rating. The score function (q) for a pair of items t_i and t_j is defined as their similarity value ($s(t_i, t_j)$). First, we have:

$$RS(t_i,t_j) = max \parallel s(t_i,t_j) - s'(t_i,t_j) \parallel_1$$

After inserting the similarity values for $s(t_i, t_j)$, we have:

$$s(t_i,t_j) - s'(t_i,t_j) = \frac{r_{t_i} \cdot r_{t_j}}{\| r_{t_i} \| \times \| r_{t_j} \|} - \frac{r'_{t_i} \cdot r'_{t_j}}{\| r'_{t_i} \| \times \| r'_{t_j} \|}$$
$$= \frac{r_{t_i} \cdot r_{t_j} \times \| r'_{t_i} \| \times \| r'_{t_j} \| - r'_{t_i} \cdot r'_{t_j} \times \| r_{t_i} \| \times \| r_{t_j} \|}{\| r_{t_i} \| \times \| r_{t_j} \| \times \| r'_{t_i} \| \times \| r'_{t_j} \|} = \frac{P}{Q}$$

We assume that the profile of a user x, in D, is not present in D'. This user rated both t_i and t_j. Note that if this user rated one of these items or none, then the similarity value does not depend on the presence or absence of this user in the dataset. We have: $\| r'_{t_i} \| \times \| r'_{t_j} \| \leq \| r_{t_i} \| \times \| r_{t_j} \|$.

We have P= $(r_{t_i} \cdot r_{t_j} \times \| r'_{t_i} \| \times \| r'_{t_j} \| - r'_{t_i} \cdot r'_{t_j} \times \| r_{t_i} \| \times \| r_{t_j} \|)$ and Q=$(\| r_{t_i} \| \times \| r_{t_j} \| \times \| r'_{t_i} \| \times \| r'_{t_j} \|)$. Since $Q \geq 0$, we have two conditions depending on whether $P \geq 0$ or $P \leq 0$.

If $P \geq 0$, then

$$\| s(t_i,t_j) - s'(t_i,t_j) \|_1 \leq \frac{(r_{t_i} \cdot r_{t_j} - r'_{t_i} \cdot r'_{t_j})}{\| r'_{t_i} \| \times \| r'_{t_j} \|} = \frac{r_{t_{xi}} \times r_{t_{xj}}}{\| r'_{t_i} \| \times \| r'_{t_j} \|}.$$

If $P \leq 0$, then

$$\| s(t_i,t_j) - s'(t_i,t_j) \|_1 \leq \frac{r_{t_i} \cdot r_{t_j}}{\| r'_{t_i} \| \times \| r'_{t_j} \|} - \frac{r_{t_i} \cdot r_{t_j}}{\| r_{t_i} \| \times \| r_{t_j} \|}.$$

Hence, we have the recommendation-aware sensitivity as:

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

□

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

[9] Our notion of recommendation-aware sensitivity is slightly different from the one presented in [39, 40].

Similar to [39, 40], we use a *truncated similarity* (Step 7 in Algorithm 4) along with our revised recommendation-aware sensitivity to enhance the quality of selected neighbors. The two theorems which prove that this truncated similarity along with our recommendation-aware sensitivity can enhance the quality of neighbors are as follows. (The proofs have been omitted due to lack of space.)

THEOREM 3. *Given an 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 recommendation-aware sensitivity between t_i and other items by RS. 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 $w = min(Sim_k(t_i), \frac{4k \times RS}{\epsilon'} \times ln\frac{k \times (|v|-k)}{\rho})$.*

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 an 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 $w = min(Sim_k(t_i), \frac{4k \times RS}{\epsilon'} \times ln\frac{k \times (|v|-k)}{\rho})$.*

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 recommendation-aware sensitivity provides good quality of neighbors while providing $\epsilon'/2$-differential privacy. The predictions are finally computed leveraging the PNCF mechanism (Algorithm 5) which adds Laplacian noise to provide $\epsilon'/2$-differential privacy. By additive property of differential privacy, PNSA and PNCF together provides ϵ'-differential privacy. The item-based version of X-MAP includes the additional feature of *time-based relevant* predictions to boost the recommendation quality traded for privacy.

Here we provide only one demonstration of the flexibility of our heterogeneous recommender due to lack of space. Since the AlterEgo profile can be considered as an actual profile in the target domain, thereby any homogeneous recommendation algorithm [1] like Matrix Factorization techniques, can be applied to the target domain to generate the recommendations.

5. IMPLEMENTATION

In this section, we describe our implementation of X-MAP (as well as NX-MAP). Figure 5 outlines the four main components of our implementation: *baseliner*, *extender*, *generator* and *recommender*. We describe each of these components along with their functionality. Note that only the baseliner component remains same for X-MAP and NX-MAP.

5.1 Baseliner

The Baseliner computes the baseline similarities leveraging the adjusted cosine similarity (Equation 3) between the items in the two domains. It splits the item-pairs based on whether both the items belong to the same domain or not. If both items are from same domain, then the item-pair similarities will be delivered as *homogeneous 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.[10]

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: $w = min(Sim_k(t_i), \frac{4k \times RS}{\epsilon'} \times ln\frac{k \times (|v|-k)}{\rho})$
4: **for** N=1:k **do**
5: **for** item t_j in I **do**
6: $RS(t_i, t_j) = max\{max_{u_x \in U_{ij}}(\frac{r_{t_{xi}} \times r_{t_{xj}}}{\| r'_{t_i} \| \times \| r'_{t_j} \|}),$
$max_{u_x \in U_{ij}}(\frac{r_{t_i} \cdot r_{t_j}}{\| r'_{t_i} \| \times \| r'_{t_j} \|} - \frac{r_{t_i} \cdot r_{t_j}}{\| r_{t_i} \| \times \| r_{t_j} \|})\}$
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
$$\frac{exp(\frac{\epsilon' \cdot \widehat{Sim}(t_i,t_j)}{2k \times 2RS(t_i,t_j)})}{\sum_{l \in C_1} exp(\frac{\epsilon' \cdot \widehat{Sim}(t_i,t_l)}{2k \times 2RS(t_i,t_l)}) + \sum_{l \in C_0} exp(\frac{\epsilon' \cdot \widehat{Sim}(t_i,t_l)}{2k \times 2RS(t_i,t_l)})}$$
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 $N_k(t_i)$;

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.*

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

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 3. For every specific layer of items, the extender computes the top-k for the neighboring layers. For instance, for the items in the BB-layer of D^S, the extender computes the top-k from items in the BB-layer in D^T and also the top-k from the items in the NB-layer in D^S.

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

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

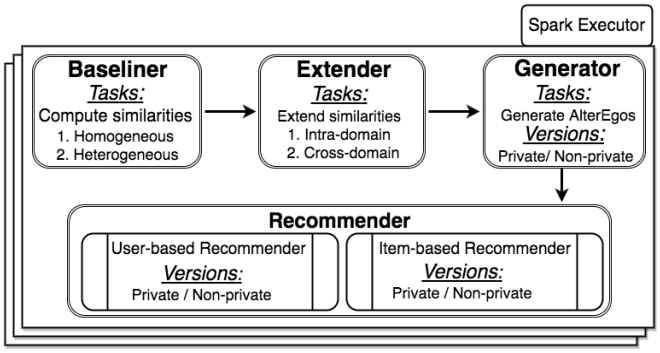


Figure 5: The components of X-MAP: *Baseliner, Extender, Generator, Recommender.*

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 NB and BB layer items in one domain to the NB and BB category items in the other one.

At the end of the execution of the extender, for every item t_i in D_S, we get a set of items $I(t_i)$ in D_T with some quantified (positive or negative) X-SIM values with t_i.

5.3 Generator

The generator performs the following two computational steps.

Item mapping. The Generator maps every item in one domain (say D^S) to its most similar item (for NX-MAP) or its private replacement (for X-MAP) in the other domain (D^T). After, the completion of this step, every item in one domain has a replacement item in the other domain.[11]

Mapped user profiles. The Generator here creates an artificial profile (AlterEgo) of a user in a target domain D_T from her actual profile in the source domain D_S by replacing each item in her profile in D^S with its replacement in D^T as shown in Figure 4. 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 leverages this artificial AlterEgo profile created by the 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.TALK ABOUT MF!

6. EXPERIMENTAL EVALUATION

[11]Note that to have more diversity, we could also choose a set of replacements for an item in the target domain, based on the X-SIM metric. This however is out of the scope of this paper.

In this section, we report on our empirical evaluation of X-MAP on a cluster computing framework namely Spark with real world traces from Amazon [24] to analyze its prediction accuracy, privacy and scalability. We choose Spark as our cluster computing framework since the underlying 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 consisting of 20 machines. Each machine is an Intel Xeon CPU E5520 @2.26GHz with 32 GB memory and 600 GB disk storage. The machines are connected through a 2xGigabit Ethernet (Broadcom NetXtremeII BCM5716).

Datasets. We use two sets of real traces from Amazon datasets [24]: *movies* and *books*. We use the ratings for the period from 2011 till 2013. The movies dataset consists of 743,735 ratings from 777,729 users for 168,105 movies where each user rated at least 10 movies. The books dataset consists of 1,424,135 ratings from 1,232,135 users for 567,731 books where each user rated at least 10 books. The ratings vary from 1 to 5 with an increment of 1 between the possible ratings. The overlapping users in these two datasets are those Amazon users who are present in both and are ascertained using their Amazon user-ids.

Evaluation metrics. We evaluate X-MAP along three complementary metrics: the recommendation *quality* as perceived by the users in terms of prediction *accuracy*, the degree of *privacy* provided to the end-users in terms of the privacy parameters (ϵ, ϵ') and the *scalability* in terms of *speedup* achieved in X-MAP by 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 [7, 15, 30]. 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 by the privacy parameters: ϵ for the Private Replacement Selection technique used for AlterEgo generation and ϵ' for the PNCF used for the private recommendation generation in X-MAP. According to the privacy literature [12, 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 [20] models the performance of speedup (S_p) as follows.

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

Due to the considerable size of 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 is applied over this aggregated single domain [11, 28]. ITEM-BASED-KNN is a linked-domain personalization approach [11, 28] where we use item-based collaborative filtering over the aggregated ratings over both the domains.

Heterogeneous personalization. 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 [4]. In the REMOTEUSER approach [4], the user similarities in source domain are used to compute the k nearest neighbors for users who have not rated in target domain. Then user-based collaborative filtering is performed.

Baseline prediction. For sparse dataset, the baseline is provided by item-based average ratings [3] or user-based average ratings [21]. 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 scheme is the ITEMAVERAGE scheme where we predict that each item will be rated as the average over all users who rated that item [3]. Note that though this technique gives a very good prediction of the actual rating but its not personalized.

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. Similarity 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 Relevance

In this section, we observe the temporal relevance effect for X-MAP, leveraging the item-based recommender, and then tune the temporal relevance parameter α accordingly. Figure 6 demonstrates this effect compared to the approach without any temporal relevance effect ($\alpha = 0$). We vary α between 0.01 to 0.3. 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 predictions negatively as it reduces the contribution of old ratings furthermore. Hence, we set α for our experiments, based on Figure 6, to achieve optimal recommendation quality.

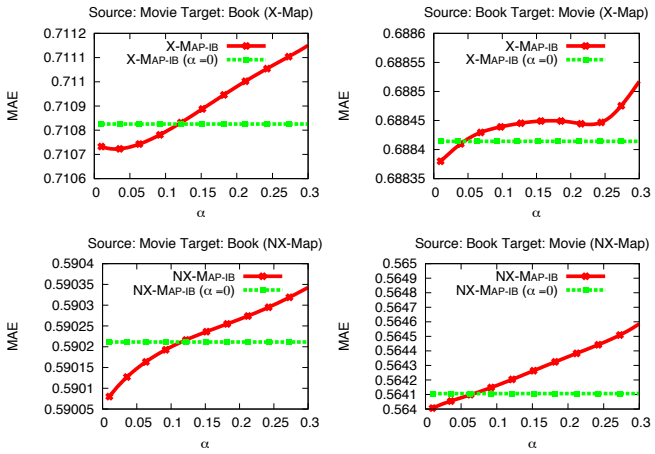
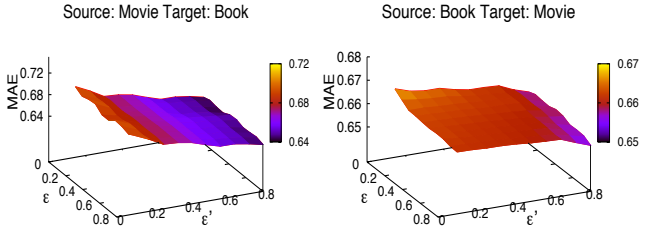


Figure 6: Temporal relevance (X-MAP, NX-MAP).

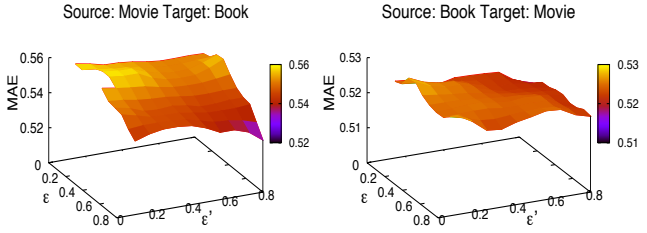
6.3 Privacy

In this section, we tune the privacy parameters (ϵ, ϵ') for X-MAP. Figures 7 and 8 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'). For the following experiments, we select the privacy parameters as follows. For X-MAP-UB, we select $\epsilon = 0.6$ and $\epsilon' = 0.3$. For X-MAP-IB, we select $\epsilon = 0.3$ and $\epsilon' = 0.8$.[12]



(a) (b)

Figure 7: Privacy trade-off (X-MAP-IB).



(a) (b)

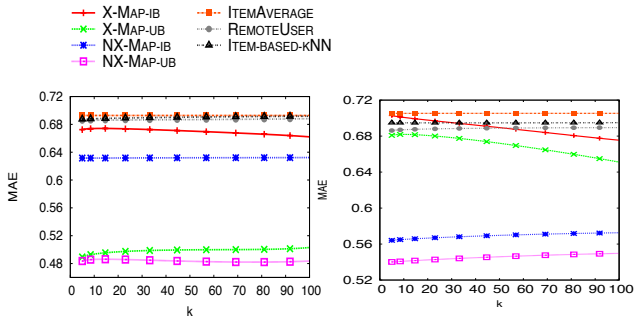
Figure 8: Privacy trade-off (X-MAP-UB).

[12]These parameters are selected from a range of possible values providing quality close to the optimal one as observed from Figures 7 and 8.

6.4 Accuracy

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

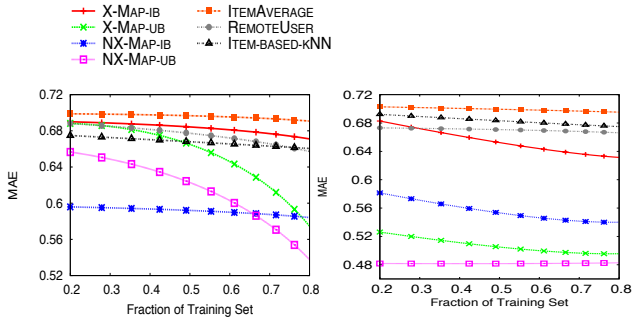
Impact of top-k neighbors. First, we evaluate the quality in terms of MAE when the size of k (neighbors in Equation 6) is varied. Figure 9(a) demonstrates that X-MAP-UB and NX-MAP-UB outperform the competitors by a significant margin of 30% where the source domain is book and the target domain is movie. Also, Figure 9(b) shows that X-MAP-UB and X-MAP-IB perform slightly 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. For all further experiments we consider k as 50.



(a) *Source:Book, Target:Movie* (b) *Source:Movie, Target:Book*

Figure 9: MAE comparison with varying *k*.

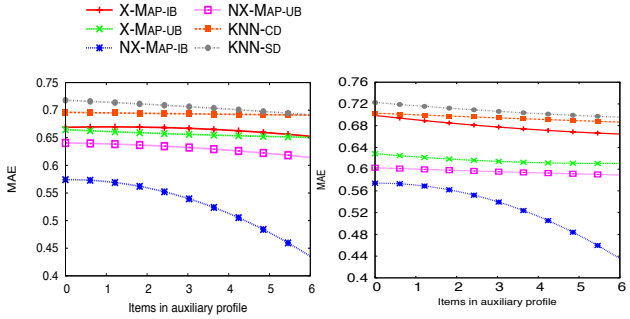
Impact of overlap. Here, we evaluate how X-MAP and NX-MAP perform when the number of users in the overlap increases. Intuitively, a good approach should provide better accuracy as more and more users connect the domains. These increasing connections improve the baseline heterogeneous similarities which are then leveraged by X-SIM. Figure 10 shows that the prediction error of X-MAP decreases as there are more users connecting the domains. Furthermore, we observe in Figure 10(a) that user-based models show more improvement than item-based ones. This behaviour occurs as item similarities are supposed to be more stable than user similarities [17].



(a) *Source:Book, Target:Movie* (b) *Source:Movie, Target:Book*

Figure 10: MAE comparison (Overlap size).[13]

Impact of sparsity. Next, we evaluate how X-MAP performs when the size of the test profile, in the target domain, increases from a minimum of 0 (cold-start situation) to a maximum of 6 (low sparsity). This experiment also demonstrates 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 target domain* denoted by KNN-SD, as well over a heterogeneous solution, *item-based kNN in aggregated domain* denoted by KNN-CD. Figure 11 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.



(a) *Source:Book, Target:Movie* (b) *Source:Movie, Target:Book*

Figure 11: MAE comparison based on profile size.

6.5 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 12 demonstrates the near-linear speedup of X-MAP. Additionally, we see that X-MAP outperforms the scalability achieved by MLLIB-ALS.

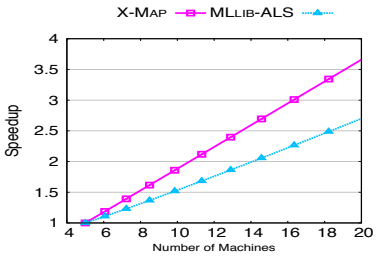


Figure 12: Scalability of X-MAP.

6.6 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 is able to recommend books 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. Hence, X-SIM is significantly efficient in achieving heterogeneous personalization in practise.

7. RELATED WORK

[13] Training set size denotes overlap size.

Heterogeneous trends. Research on heterogeneous personalization is relatively new. There has been 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) [14]. The idea is to aggregate heterogeneous information to build user profiles that are applicable across different domains. SUMs relies 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 [2].

Web Monitoring. Hyung et. al designed a web agent which profiles user interests across multiple domains and leverage this information for personalized web support [19]. 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 [4] proposed the idea of cross-domain mediation to compute recommendations by aggregating data over several recommenders. We showed empirically that X-Map outperforms cross-domain mediation in Figures 9 and 10.

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

Merging preferences. One could also view the problem as that of merging single-domain user preferences. Through this angle, several approaches can be considered.

Rating aggregation. This approach is based on aggregating user ratings over several domains into a single multi-domain rating matrix [4, 5, 38]. Berkovsky et. al showed that this approach can tackle cold-start problems in collaborative filtering [5]. We showed empirically that X-Map easily outperforms such rating aggregation based approaches [4].

Common representation. This approach is based on a common representation of user preferences from multiple domains either in the form of a *social tag* [34, 35] or *semantic relationships between domains* [22]. Shi et. al developed a Tag-induced Cross-Domain Collaborative Filtering (TagCDCF) to overcome cold-start problems in collaborative filtering [31]. 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 might be difficult to collect in practice.

Linked preferences. This approach is based on linking user preferences in several domains [11, 25]. We showed empirically that X-Map outperforms such linked preference based approaches [11] in Figures 9 and 10.

Domain-independent features. This approach is based on mapping user preferences to domain-independent features like *personality features* [8] or *user-item interaction features* [23]. 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, to generate AlterEgo profiles of users in a domain where they might not have any activity yet. We presented private as well as non-private versions of X-Map: both perform better in terms of recommendation quality than alternative heterogeneous recommenders [3, 4, 11, 21, 28]. (Although, not surprisingly, there is a trade-off between quality and privacy).

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In *TKDE*, pages 734–749, 2005.

[2] 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.

[3] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, 2009.

[4] S. Berkovsky, T. Kuflik, and F. Ricci. Cross-domain mediation in collaborative filtering. In *User Modeling*, pages 355–359. Springer, 2007.

[5] S. Berkovsky, T. Kuflik, and F. Ricci. Distributed collaborative filtering with domain specialization. In *RecSys*, 2007.

[6] M. Bonett. Personalization of web services: opportunities and challenges. In *Ariadne*, 2001.

[7] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[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] C. Dwork. A firm foundation for private data analysis. *CACM*, 54(1):86–95, 2011.

[13] 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.

[14] 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.

[15] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.

[16] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, 1999.

[17] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[18] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. In *CIMCA*, 1999.

[19] 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.

[20] S. Krishnaprasad. Uses and abuses of amdahl's law. *In Journal of Computing Sciences in Colleges*, 2001.

[21] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SIAM*, 2005.

[22] A. Loizou. *How to recommend music to film buffs*. PhD thesis, UNIVERSITY OF SOUTHAMPTON, 2009.

[23] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In *ECIR*. 2014.

[24] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

[25] M. Nakatsuji, Y. Fujiwara, A. Tanaka, T. Uchiyama, and T. Ishida. Recommendations over domain specific user graphs. In *ECAI*, 2010.

[26] 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.

[27] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW*, 1994.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[29] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[30] U. Shardanand and P. Maes. Social information filtering: algorithms for automating a word of mouth. In *SIGCHI*, 1995.

[31] Y. Shi, M. Larson, and A. Hanjalic. Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering. In *UMAP*, 2011.

[32] I. Soboroff and C. Nicholas. Collaborative filtering and the generalized vector space model (poster session). In *SIGIR*, 2000.

[33] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. 2011.

[34] M. Szomszor, H. Alani, I. Cantador, K. O'Hara, and N. Shadbolt. *Semantic modelling of user interests based on cross-folksonomy analysis*. Springer, 2008.

[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] P. Winoto and T. Tang. If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations. *New Generation Computing*, 26(3):209–225, 2008.

[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.