TALMUD – Transfer Learning for Multiple Domains

Orly Moreno, Bracha Shapira, Lior Rokach, Guy Shani
Department of Information Systems Engineering
Ben-Gurion University of the Negev
P.O.B. 653, Beer-Sheva, 84105. Israel
Deutsche Telekom Laboratories at Ben-Gurion University
Beer-Sheva, Israel
{orlymore, bshapira, liorrk, shanigu}@bgu.ac.il

ABSTRACT

Most collaborative Recommender Systems (RS) operate in a single domain (such as movies, books, etc.) and are capable of providing recommendations based on historical usage data which is collected in the specific domain only. Cross-domain recommenders address the sparsity problem by using Machine Learning (ML) techniques to transfer knowledge from a dense domain into a sparse target domain. In this paper we propose a transfer learning technique that extracts knowledge from multiple domains containing rich data (e.g., movies and music) and generates recommendations for a sparse target domain (e.g., games). Our method learns the relatedness between the different source domains and the target domain, without requiring overlapping users between domains. The model integrates the appropriate amount of knowledge from each domain in order to enrich the target domain data. Experiments with several datasets reveal that, using multiple sources and the relatedness between domains improves accuracy of results.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information search and Retrieval – *information filtering;* H.2.8 [**Database Management**]: Database Applications – *data mining.*

General Terms

Algorithms, Experimentation

Keywords

Recommender Systems, Transfer Learning, Cross Domains, Collaborative Filtering

1. INTRODUCTION

The overwhelming amount of information existing nowadays raises the need for intelligent systems that will provide personalized recommendations and services. Usually, recommender systems (RS) provide recommendations or rating predictions for a single domain (such as movies, books, etc). In this domain the RS provides recommendations based on historical usage data that was collected only in the specific domain.

In many cases, in particular when the system is new, the data sparsity problem arises, and prevents the system from generating accurate recommendations. This problem is especially critical when using collaborative filtering (CF) algorithms whose accuracy relies on the availability of observed ratings from a sufficient number of users. If two users do not share a sufficiently large set of rated items, then the user-to-user similarity computation is not reliable and cannot support accurate rating predictions [5].

Over the web there exists a considerable number of publicly available user-item rating datasets from multiple sources. One can leverage this abundance of available data for boosting the performance of a specific recommender system of interest, by reducing the sparsity of its dataset. Unfortunately, one rarely possesses datasets which contain the exact same items, and even more rarely the exact same users. Clearly some domains are more closely related to the target domain then others. For example, books are more closely related to movies than to electronic gadgets. Thus, sophisticated methods are required to leverage data collected on one domain to be useful for another domain. These methods are often known as cross-domain recommenders in the recommendation system literature.

Cross-domain techniques typically originate in the machine learning literature and most specifically from Transfer Learning (TL). Transfer learning is a set of methods for extracting knowledge from one task or domain to be used in another task or a different domain [16]. Recently, transfer learning techniques were proposed for recommender systems applications in order to improve predictions in sparse target domains by reusing related domains data [14, 15, 20]. For example, Li et al. [14] suggested transferring user-item rating patterns from a dense source rating matrix in a single domain, to a sparse rating matrix in a related target domain. However, these previous methods assume that all domains are equally related and can positively contribute to the target system.

When transferring knowledge from one domain to another, we might transfer destructive information or data which is not consistent with the target domain. For example, if the domains are rarely related, or the users in the domains behave differently (they tend to be more critical in one domain and more favorable in another), using user behavior data from one domain for the other would have a negative effect on the model. This phenomenon is known as Negative Transfer [16] and may result in decreasing the target model accuracy.

In light of the above discussion and following Li et al. [14] we propose a novel approach which we call TALMUD (Transfer Learning for Multiple Domains) to alleviate the sparsity problem. Our method enables to predict missing values in the target domain by extracting knowledge from multiple source domains and transferring it to a single sparse target domain. We do not assume an identical level of relatedness between the domains. Rather, the algorithm automatically learns the degree of relatedness according to the rating patterns correspondence between the domains. Thus, it determines the amount of knowledge to be transferred from each source domain according to its relatedness to the target domain and its sparseness. This is done without assuming that there are shared users or items between the source and target domains.

The rest of this paper is organized as follows. We survey several studies which applied cross domain techniques for recommender systems in Section 2. In Section 3 we present our method. Section 4 reports preliminary experimental results based on several recommendation datasets. Section 5 presents an improved version of the algorithm and experimental results. Concluding remarks and future work are discussed in Section 6.

2. BACKGROUND

In this section we survey recommender systems, collaborative filtering, and the data sparsity problem. Next, we discuss the transfer learning approach to enhance machine learning datasets. Finally, we discuss in depth previous attempts of using transfer learning in recommender systems.

2.1 Recommender Systems

Recommender systems provide either recommended items, or rating predictions, in many commercial applications nowadays. For example, in NetFlix[1], the popular online video rental service, a user can view the predicted ratings for movies she has not yet seen. This can help the user in making intelligent choices when deciding which movie to rent.

Perhaps the most widely used approach for computing rating prediction is collaborative filtering (CF), where user or item (e.g. movie) correlation is based on previously observed user-item ratings. In CF, the dataset used for prediction can be modeled as a user-item rating matrix, where each row represents a user and each column represents an item, and each cell represents a user-item rating. Clearly, as users seldom rate more than a few items in many domains, this matrix is expected to be very sparse. The accuracy of the rating prediction, however, is often improved as the matrix becomes denser. This is known as the sparsity problem in CF.

There are many available CF algorithms, such as direct user-user correlation [2], singular value decomposition (SVD) [12], clustering [9], and many more.

2.2 Transfer Learning

Transfer learning (TL) aims at extracting knowledge that was learned for one task in a domain and use it for a target task in a different domain [16]. In the field of Machine Learning we usually train a model based on data that is available for the problem that we are interested in (training data). This model is used for predicting the behavior in the examined domain (using the testing data). For example, in recommender systems this model can help us predict whether the user will like the movie or not.

As stated in [16] many machine learning methods are based on the hypothesis that the training and the testing datasets have common features and distribution. In contrast to those techniques, transfer learning allows the domains, tasks and distribution of the training and testing to differ. Thus, when having insufficient data in the domain of interest, we can exploit data from mature applications (domains) which already have enough data. We can transfer the knowledge that is consistent with the target domain in order to train a model for it. For the text mining field transfer learning might be used when labeled documents exist only for mature domains and are not available for other domains. For example, Dai et al. [4] utilized labeled documents from one domain in order to classify unlabeled documents of a different domain with different distribution, using co-clustering as a bridge between the source and target domain. Xue et al. [19] had also coped with the problem of unavailable labeled data for classifying text in a new domain. They built a topic-bridge PLSA model that uses the common topics between the domains in order to classify the new domain's documents.

Another important characteristic of TL is that it does not necessarily require content overlap between the different domains. Regarding the recommender systems application, TL does not require that the source and target domain will share users or items, as it aims at finding common consumption patterns which exist in related domains by recognizing latent behavior groups [14]. Let us consider for example the Music and Games domains. Although these domains do not seem to be strongly connected, we can still find the same latent groups of users in both domains. For example, there might be a group of consumers that always purchase the new and trendy items, another group which likes to consume low cost products, and other users that mainly like childish items.

Transferring knowledge between domains is a very challenging task because it cannot be guaranteed that the knowledge of one domain is useful for another domain. The success of transfer learning depends on a variety of factors, e.g. how correlated the domains are (for example, it is possible that movies and books are more correlated than movies and jokes), the data characteristics (sparsity level, rating scale etc.) and whether the domains share resources like items or users.

Henceforth, we refer to the system or domain for which the recommendations are required as the ***target domain*** and the domain from which data is extracted as the ***source domain***.

2.3 Recommender Systems Cross-Domain Techniques

In recommender systems, the transfer learning problem is often known as cross-domain recommendation [13]. Several studies have been conducted on applying cross domain techniques, and transfer learning in particular.

Perhaps the simplest approach to cross-domain recommendations is by importing relevant data from a different domain and aggregating the data with the original target data. This aggregation is simple when the domains share information on the same users. Berkovsky et al. [1] refer to this problem as cross domain mediation and introduce several techniques for importing relevant data. For example, it is possible to merge the rating matrices such that the items of the remote domain are added as additional features. Then, executing any CF algorithm on the combined matrix allows the system to leverage rating behavior similarity in one domain to predict ratings in the other domain.

Possibly the most relevant work is by Li et al [14]. They suggested that when insufficient data in the target domain prevents us from training an accurate recommendation model, it is possible to borrow useful knowledge from a different domain and

[1] www.Netflix.com

use its data to train the model. Li et al. [14] introduced the idea that rating matrices of different domains may share similar user-item rating patterns. Thus, they learn a user-item rating matrix of the source domain which is referred to as a "codebook", and transfer the rating patterns to the target domain in order to fill in the target domain missing values.

Their algorithm consists of two steps. First, a rating pattern (codebook) of the dense domain is created, which summarizes the original rating data. Second, the codebook is expanded in order to learn the missing values in the target domain.

The codebook consists of k users clusters, and l item clusters and is constructed by simultaneously clustering the users (rows) and items (columns) of the source rating matrix, using the orthogonal non-negative matrix tri-factorization (ONMTF) clustering algorithm [7]. This method is equivalent to the two-way K-means clustering algorithm. As a result of the above process the codebook indicates the rating that a user who belongs to a specific user cluster will give to an item that belongs to a specific item cluster.

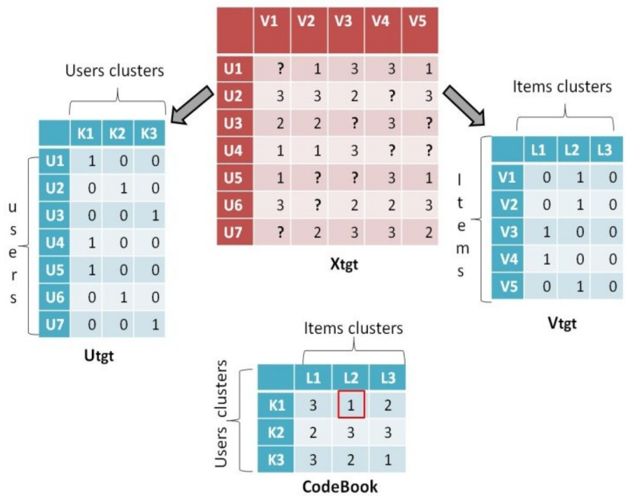
The second step of the algorithm is based on the assumption that there is a set of users/items in the target matrix that behaves like the i-th user/item cluster pattern in the source domain. Thus, the model maps target users and items to the corresponding source clusters. Those mappings are denoted by the matrices U_{tgt} and V_{tgt} respectively. Each row in U_{tgt} and V_{tgt} indicates a target user (for U_{tgt}) or item (for V_{tgt}) while each column indicates the source domain user's cluster or item's cluster respectively. U_{tgt} and V_{tgt} consist of binary values where "1" indicates a membership of a user (from U_{tgt}) or an item (from V_{tgt}) to a cluster from the source domain. Li et al. [14] assume (and we follow their assumption), that each user and item belongs only to one user's and item's cluster. Thus, only one value of "1" can be assigned to each row. For example, Figure 1 shows that user u_1 from the target domain, which is denoted as X_{tgt}, belongs to users cluster k_1 in the source domain, whereas item v_1 belongs to items cluster L_2 in the source domain. Thus, according to the source domain codebook, the missing rating of user u_1 to item v_1 is 1.

After the cluster memberships U_{tgt} and V_{tgt} are constructed, the target matrix can be filled using:

$$\tilde{X}_{tgt} = W \circ X_{tgt} + [1 - W] \circ [U_{tgt} B V_{tgt}^T]$$

The matrix W is a binary matrix where $W_{ij} = 1$ if $X_{tgt_{ij}} \neq 0$ (this entry is rated), otherwise $W_{ij} = 0$. The notation of $\circ$ denotes the entry-wise product. The target rating matrix reconstruction is demonstrated in Figure 2. The examples in Figure 1 and Figure 2 are adopted from Li et al. [14]. Experimental results have shown that additional information from a related domain can be gained and improve recommendations in the target domain. However, they do not measure the relatedness between the domains. Moreover, this method enables to utilize knowledge from one source domain only.

Li et al. [15] extended this method to integrate several sparse domains. In this work they assume that multiple sources share a single latent pattern. They discover this pattern and then learn the probabilities of each user and item to belong to this shared latent structure. In many cases, however, sources do not necessarily share such a latent pattern, a problem that we address here. Moreover, our work differs in the problem it aims to solve. While our approach searches for the optimal solution that will most accurately enrich the target domain, Li et al. [15] aim to simultaneously enrich several domains, and there is no source-target distinction. Their proposed model equally takes into consideration the limitation and needs of all the domains and therefore does not guarantee that the solution is the most optimal one for a single domain. In Transfer learning, this is called the multi task problem [16] and is generally treated differently. For completeness we will compare our suggested method with the method proposed in [15].



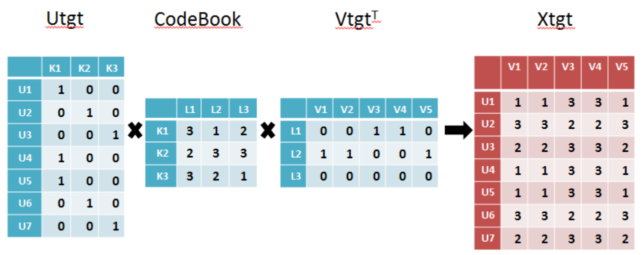
	V1	V2	V3	V4	V5
U1	?	1	3	3	1
U2	3	3	2	?	3
U3	2	2	?	3	?
U4	1	1	3	?	?
U5	1	?	?	3	1
U6	3	?	2	2	3
U7	?	2	3	3	2

	K1	K2	K3
U1	1	0	0
U2	0	1	0
U3	0	0	1
U4	1	0	0
U5	1	0	0
U6	0	1	0
U7	0	0	1

	L1	L2	L3
V1	0	1	0
V2	0	1	0
V3	1	0	0
V4	1	0	0
V5	0	1	0

	L1	L2	L3
K1	3	1	2
K2	2	3	3
K3	3	2	1

Figure 1. Decomposition of the target matrix (X_{tgt}) into user membership matrix (U_{tgt})and item membership matrix (V_{tgt})according to the source domain clusters



	K1	K2	K3
U1	1	0	0
U2	0	1	0
U3	0	0	1
U4	1	0	0
U5	1	0	0
U6	0	1	0
U7	0	0	1

	L1	L2	L3
K1	3	1	2
K2	2	3	3
K3	3	2	1

	V1	V2	V3	V4	V5
L1	0	0	1	1	0
L2	1	1	0	0	1
L3	0	0	0	0	0

	V1	V2	V3	V4	V5
U1	1	1	3	3	1
U2	3	3	2	2	3
U3	2	2	3	3	2
U4	1	1	3	3	1
U5	1	1	3	3	1
U6	3	3	2	2	3
U7	2	2	3	3	2

Figure 2. The reconstruction of the target rating matrix X_{tgt}

2.4 Other Related Works

Pan et al. [17] address two limitations of previous studies. First, they transform knowledge from domains which have heterogeneous forms of user feedback. For example, one domain has ratings records while another domain records user's events on the system. Second, they do not assume that the knowledge on both the users and the items can be learned from the same domain. They suggest that the user dimension can be learned from one domain that has information on the same users, while the items dimension can be learned from another domain that shares items with the target domain. However, their proposed method can only be applied in cases where the first source domain and the target domain share the same users, and that the second source domain and the target domain share the same items. In many real-world applications such shared data does not exist.

Cao et al. [3] refer to the recommendation problem as a Link-Prediction problem. Link prediction is defined by Getoor et al. [8] as a problem that predicts the existence of a link between two entities. In recommender systems we wish to predict a possible link between a user and an item. This paper handles data sparsity and the cold start problem, where a new user or a new item are added to the system and we do not have enough data which is relevant to them. The proposed method aims to improve the performances of all recommendation tasks, by exploring the correlation between link prediction tasks in multiple heterogeneous domains and transfer the shared knowledge among similar tasks. In order to find the different correlations between the domains, they use a task similarity kernel while assuming that the users in the different domains overlap. In our proposed approach this is not a mandatory requirement.

Zhang et al. [20] address the sparsity problem by considering collaborative filtering recommendation tasks of multiple sparse domains together as one problem. This method models the rating prediction problem using probabilistic matrix factorization. The correlation between the domains is learned and exploited in the model. However, similarly to Li et al. [15], this method refers to the multi task problem [16] and simultaneously enriches several domains.

To conclude, all the studies discussed above demonstrate the effectiveness of applying cross domain techniques to alleviate the sparsity problem comparing to other methods which mine only single domain data. However, the majority of these papers assume that the source and target domains are related but do not suggest methods to calculate or estimate this relatedness.

The studies that do measure the correlation between the domains assume that the domains share resources like items and users although this restriction is unrealistic when one wishes to leverage publicly available datasets to augment a specific recommender system. Moreover, the studies that deal with using multiple domains do not define a target domain which is sparse, but build a model from all the domains in order to enrich all of them simultaneously. The proposed approach measures the relatedness between domains, without assuming overlapping users or items, while allowing the use of multiple sources domains.

3. TALMUD

We now discuss our new method which we call TALMUD (Transfer Learning for Multiple Domains), that allows us to automatically learn the relatedness of multiple data sources, and transfer knowledge from all these sources into a single target domain.

Our method extends the algorithm proposed by Li et al. [14], for transferring data from a single domain, allowing for transferring data from multiple source domains with varying levels of relevance. We use N user-item source matrices, each represented as an $r_n \times m_n$ rating matrix, and denoted X_{src_n} for source matrix $n \in \{1 \dots N\}$. The target matrix is a sparse $p \times q$ rating matrix, denoted X_{tgt}. The codebook B_n encodes the user-item clusters in X_{src_n}. B_n is a $k_n \times l_n$ matrix where k_n and l_n are user defined parameters that define the cluster dimensions. Each codebook represents the transferred knowledge from its corresponding source domain.

The proposed method linearly integrates the rating patterns of all source domains into one model in order to enable prediction of the target matrix missing values. The main challenge is in defining the integration of knowledge from the different sources, and the amount of knowledge that needs to be transferred from each source domain. We hence define three sets of decision variables $\{U_{tgt_n}, V_{tgt_n}, \alpha_n\}$ for each source domain n. Equation 1 formulates the optimization problem that finds the best users' and items' cluster memberships and the relatedness coefficients. Thus, the error of the prediction in the target domain is minimized. Henceforth, U_{tgt_n} and V_{tgt_n} are denoted U_n and V_n for each source domain n.

$$\min_{\substack{U_n \in \{0,1\}^{p \times k_n} \\ V_n \in \{0,1\}^{q \times l_n} \\ \alpha_n \in R \ \forall n \in N}} \left\| \left[X_{tgt} - \sum_{n=1}^{N} \alpha_n \left(U_n B_n {V_n}^T \right) \right] \circ W \right\|_F^2 \tag{1}$$

$$S.T. U_n 1 = 1, V_n 1 = 1$$

The first set of variables refers to the users' cluster memberships U_n for each source n. U_n is a $p \times k_n$ matrix. The second set refers to the items' cluster memberships V_n for each source n. V_n is a $q \times l_n$ matrix. U_n and V_n are binary matrices, where "1" indicates a membership of user or item to a cluster respectively. Following Li et al. [14], we use the notation $\circ$ for entry-wise product and the notation $U_n 1 = 1, V_n 1 = 1$ to ensure that each user or item will be assigned to one cluster only. The observed ratings of the target domain are reflected in the binary Matrix W where $W_{ij} = 1$ if $X_{tgt_{ij}} \neq 0$ (i.e. this entry is rated), else $W_{ij} = 0$. Using this matrix we ensure that the error is calculated based only on the observed ratings

We find the target's users and items clusters by examining the different combinations of users/items clusters in all source domains and choosing the combination that best predicts the target ratings. This process is further described in Algorithm 1. Since we utilize knowledge form multiple source domains, we believe that the interaction between the sources affects the relatedness of the user to the cluster in each source domain. Thus the cluster memberships should be based on the user's global behavior rather than on her behavior in a single domain.

For example, let us assume that we have only a single movie source domain and respectively only one codebook B_1, where user u_1 fits best to cluster k_1 according to the learned model. When we take into account an additional music source domain, and more knowledge is available, the best cluster memberships for the same user u_1 based on data from all domains simultaneously, might become cluster k_2 of the movie codebook B_1.

Based on this intuition, we learn for each source domain n its correspondent matrices U_n and V_n based on all source domains simultaneously, rather than learning each domain separately and combining the models. This process results in multiple codebooks.

The third set of variables is α_n where $\alpha_n \in R \ \forall n \in N$. α_n denotes the relatedness coefficient between source domain n and the target domain. We aim to find the optimal α_n values such that the MSE is minimized, while U_n and V_n are fixed. The error function denoted as E is defined in Equation 2. In practice, minimizing E is equivalent to minimizing the MSE.

$$E = \left\| \left[X_{tgt} - \sum_{n=1}^{N} \alpha_n \left(U_n B_n {V_n}^T \right) \right] \circ W \right\|_F^2 \tag{2}$$

For simplicity, henceforth we denote $(U_n \, B_n \, V_n^T) \circ W$ as T_n and $X_{tgt} \circ W$ as X. Note that $[U_n B_n {V_n}^T] \in R^{p \times q}$.

Equation 2 can be rewritten as follows:

$$E = \sum_{i=1}^{p}\sum_{j=1}^{q}\left(\left[X_{ij} - \sum_{n=1}^{N}\alpha_n[T_n]_{ij}\right]\right)^2 \qquad (3)$$

Thus, the optimization formulation is defined as

$$\vec{\alpha} = argmin_{\vec{\alpha}}[E] \qquad (4)$$

In order to solve equation 4, we calculate the gradient of $\vec{\alpha}$ and set it equal to zero as shown in equation 5.

$$\frac{\partial E}{\partial \alpha_{n^*}} = -2\sum_{i,j}\left(X_{ij} - \sum_{n=1}^{N}\alpha_n[T_n]_{ij}\right)\ [T_{n^*}]_{ij} = \\ -\sum_{i,j}X_{ij}[T_{n^*}]_{ij} + \sum_{n=1}^{N}\sum_{i,j}\alpha_n[T_n]_{ij}[T_{n^*}]_{ij} = 0 \qquad (5)$$

And rearranging gives:

$$\sum_{n=1}^{N}\sum_{i,j}\alpha_n[T_n]_{ij}[T_{n^*}]_{ij} = \sum_{i,j}X_{ij}[T_{n^*}]_{ij} \qquad (6)$$

Thus we obtain the following set of linear equations:

$$\begin{pmatrix} \sum_{i,j}[T_1]_{ij}^2 & \cdots & \sum_{i,j}[T_1]_{ij}[T_N]_{ij} \\ \sum_{i,j}[T_2]_{ij}[T_1]_{ij} & \cdots & \vdots \\ \vdots & \ddots & \vdots \\ \sum_{i,j}[T_N]_{ij}[T_1]_{ij} & \cdots & \sum_{i,j}[T_N]_{ij}^2 \end{pmatrix}\begin{pmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_N\end{pmatrix} = \begin{pmatrix}\sum_{i,j}X_{ij}[T_1]_{ij}\\ \sum_{i,j}X_{ij}[T_2]_{ij}\\ \vdots\\ \sum_{i,j}X_{ij}[T_N]_{ij}\end{pmatrix} \qquad (7)$$

Finally, the optimal α_n values can be obtained by solving (7):

$$\begin{pmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_N\end{pmatrix} = \begin{pmatrix} \sum_{i,j}[T_1]_{ij}^2 & \cdots & \sum_{i,j}[T_1]_{ij}[T_N]_{ij} \\ \sum_{i,j}[T_2]_{ij}[T_1]_{ij} & \cdots & \vdots \\ \vdots & \ddots & \vdots \\ \sum_{i,j}[T_N]_{ij}[T_1]_{ij} & \cdots & \sum_{i,j}[T_N]_{ij}^2 \end{pmatrix}^{-1}\begin{pmatrix}\sum_{i,j}X_{ij}[T_1]_{ij}\\ \sum_{i,j}X_{ij}[T_2]_{ij}\\ \vdots\\ \sum_{i,j}X_{ij}[T_N]_{ij}\end{pmatrix} \qquad (8)$$

We learn the decision variables $\{\ U_n\ , V_n, \alpha_n\ \}$ until converging to the local minimum.

After solving the optimization problem in equation (1) we can construct from each source domain a full matrix with the same dimensions as the target matrix using $U_nB_nV_n^T$. As shown in equation 9, a linear combination between those matrices, weighted by the set of α_n, will establish the full target matrix. We denote the full rating matrix as $\tilde{X}_{tgt}$.

$$\tilde{X}_{tgt} = W \circ X_{tgt} + [1-W]\circ[\sum_{n=1}^{N}\alpha_n(U_nB_nV^T{}_n)] \qquad (9)$$

3.1 Algorithm

The proposed cross-domain recommendation algorithm consists of the following stages. On the first stage the algorithm learns the rating patterns of each source domain separately. It constructs a codebook B_n from each source domain X_{src_n} as described by Li et al. [14]. On the next stage, the optimization problem defined above (shown in Algorithm 1) is solved by learning the target cluster memberships U_n and V_n and the relatedness coefficients α_n $\{\forall n \in N\}$. The optimization problem is solved iteratively; Each iteration t consists of three steps. On each step we solve one of the three variable set types $\{U_n, V_n, \alpha_n\}$ while fixing the other two [10]. First, we initialize the three sets of variables (lines 1-9). Then, to solve U_n (lines 11-14) we create possible rating rows for each user by expanding the codebooks, based on the item clusters V_n that were discovered at the former iteration. Then, we find the best linear combination of the target user's clusters, using the α values that minimize the error of the observed target ratings. In line 12 we examine the different existing combinations of user's clusters in all source domains and choose the combination that decreases the differences between the observed ratings and the

Algorithm 1 Optimization Problem Solution

Input: Target rating matrix X_{tgt}, source domains Codebooks B_n, number of sources N.
Output: the filled-in target rating matrix $\tilde{X}_{tgt}$

```
1.  for n←1,…N do
2.      for i←1,…,q do
3.          Randomly select $\hat{\jmath}$ from $\{1,\dots,l_n\}$
4.          $[V_n^{(0)}]_{i\hat{\jmath}} \leftarrow 1; [V_n^{(0)}]_{i\hat{\jmath}} \leftarrow 0$ for $j \in \{1,\dots,l_n\}/\hat{\jmath}$
5.      end for
6.  end for
7.  for $n \leftarrow 1,\dots,N$ do
8.      $\alpha_n^{(0)} \leftarrow \frac{1}{N}$
9.  end for
10. for $t \leftarrow 1,\dots,T$ do
11.     for $i \leftarrow 1,\dots,p$ do
12.         $\vec{\jmath} = argmin_{\vec{\jmath}}\left\|[X_{tgt}]_{i*} - \sum_{n=1}^{N}\alpha_n^{(t-1)}\left[B_n[V_n^{(t-1)}]^T\right]_{j*}\right\|_{W_{i*}}^2$
13.         $[U_n^{(t)}]_{i,\vec{\jmath}[n]} \leftarrow 1; [U_{tgt}^{(t)}]_{i,j} \leftarrow 0$ for $j \in \{1,\dots,k_n\}/\hat{\jmath}[n]$
14.     end for
15.     for $i \leftarrow 1,\dots,q$ do
16.         $\vec{\jmath} = argmin_{\vec{\jmath}}\left\|[X_{tgt}]_{*i} - \sum_{n=1}^{N}\alpha_n^{(t-1)}[U_n^{(t)}B_n]_{*j}\right\|_{W_{*i}}^2$
17.         $[V_n^{(t)}]_{i,\vec{\jmath}[n]} \leftarrow 1; [V_n^{(t)}]_{ij} \leftarrow 0$ for j∊{1,…,l_n}/$\vec{\jmath}[n]$
18.     end for
19.     $\vec{\alpha}^{(t)} = \min_{\alpha_n}\left\|\left[X_{tgt} - \sum_{n=1}^{N}\alpha_n^{(t)}\left(U_nB_nV_n{}^T\right)\right]\circ W\right\|_F^2$
20. end for
21. $\tilde{X}_{tgt} = W\circ X_{tgt} + [1-W]\circ[\sum_{n=1}^{N}\alpha_n(U_nB_nV_n^T)]$
```

ratings that are predicted using the model. We evaluate these differences using a quadratic loss function. We adopt Li et al. [14] notation to denote the weighted l_2 norm $\|X\|_{W_{i*}}^2 = [X^T diag(W_{i*})X]$, where $[\cdot]_{i*}$ denotes the i-th row in a matrix and $[\cdot]_{*i}$ denotes the i-th column. We use an auxiliary vector $\vec{\jmath}$ (line 12) to maintain the best user's clusters indexes from each domain. Then we calculate $U_n^{(t)}$ (line 13) for each source domain and use it to create optional rating columns to the target item. Similarly, we find the item's clusters correspondence and calculate $V_n^{(t)}$ (lines 15-18). Then, we learn the relatedness coefficient α_n (line 19), by minimizing the error based on the observed target ratings as shown in equation 4.

This iterative process stops with convergence to the local minimum (or after performing t iterations). Finally, we fill the target rating matrix using equation 9. The knowledge required for filling the rating matrix X_{tgt} is drawn from each source domain according to the degree of relatedness α_n.

4. PRELIMINARY RESULTS

We now present some preliminary experiments that show the strength of TALMUD, but also expose some problems. Then, we propose how these problems can be reduced.

4.1 Datasets

For our experiments we used two benchmark datasets in the RS community - Netfilx and Jester[2]. Additionally we used two proprietary datasets obtained from Deutsche Telecom

[2] http://goldberg.berkely.edu/jester-data/

Laboratories. Following is a summarization of the main data set characteristics.

- We used a subset of Netflix –A movies ratings dataset with a rating scale of 1 to 5. We extracted 110 users, 110 items and 12,100 rating records to ensure no sparsity (the matrix is full of ratings) for using it as source matrix in the experiments.
- Jester is a jokes rating dataset (scale -10 to 10). We used a subset with no sparsity that contains 500 users, 100 items and a total of 50,000 rating records. In order to achieve consistent rating scale of 1-5 ratings, we uniformly normalized the rating scale from 1 to 5.
- The Music Loads[3] and Games Loads[4] datasets consist of recorded user's events on products in the system (e.g., a user buys an item, a user clicks on an item, etc). Since the matrices are very sparse (99.99%), the most dense matrices that we managed to extract are of ~97% sparsity. We randomly selected 632 users that have events on at least 10 different items, and 817 music items with at least 5 events of different users. We performed the same process on the Games Loads dataset resulting with 632 users and 1264 items. As a preparation of the data for the preliminary evaluation we transformed the data from events to ratings. We use an exponential scale to score the events in order to emphasis the difference between the events. E.g. the most meaningful user event, buying an item, gets the highest score. Moreover, we considered the number of events that a user performed on the same item. That is, if a user plays a song a couple of times it may indicate that the user really likes the item compared to another song that he plays only once. The process is demonstrated in Table 1 and Table 2. Table 1 shows score categories. For example, a score of 2 is assigned when the user either clicks on an item or clicks to recommend the item. Table 2 demonstrates the aggregation of scores in a case that a user has multiple events on the same item. This is done by multiplying the score of the event by the number of times it occurred. In order to make sure that the rating scale is identical for all the users, we performed individual normalization for each user, so that the rating is between1-5.

4.2 Experimental Settings

The goal of the proposed method is to learn the missing values in a target matrix using multiple domains. Thus, we compare the accuracy results of our method TALMUD (Transfer Learning for Multiple Domains) to Li et al. [14] method, CBT, which learns only from one source domain.

We conducted two experiments. In the first experiment we use the Games loads dataset as target domain while the other domains were used as source domains. In the second experiment the Music loads dataset was used as target domain. The target domain was divided into training and testing datasets. The training data consisted of 80% of the ratings while the testing data consisted of the remaining 20%. Since the Loads datasets events have time stamps we used the most recent 20% of the events for testing.

Both TALMUD and CBT were executed with three different combinations of source domains for building the model. Each combination of two source domains was examined (e.g. on the first experiment the combinations were: movies and jokes, music and jokes, music and movies). Using this setting we are able to

[3] http://www.musicload.de/

[4] http://www.gamesload.de/

Table 1. Music Loads and Games Loads events scores

Score 2	Score 4	Score 8	Score 16
User clicks item	User adds to watch list	User plays item	User buys item
User clicks recommend ed item	User tags item	User streams item	User buys recommended item
	User comments item	User adds to shopping cart	
	Users adds to play list		

Table 2. An example of calculating the score of user u to item v based on his events regarding the item

Event	Score	Number of events	Sum score
User clicks item	2	5	10
User plays item	8	3	24
User buys item	16	1	16
Total score	-	-	50

create multiple scenarios to compare the two approaches. The codebook dimensions, k and l , were set according to the intuition suggested by Li et al. [15] that the clusters model should be compact enough to avoid over-fitting, but expressive enough to capture significant behavior patterns. We set k and l to 20 for all source domains similarly to Li et al. [15].

In addition, we compared our method with a trivial linear combination of multiple source domains. That is, we built a model on each source domain separately using the CBT algorithm and aggregated the models by extracting an equal amount of knowledge from each source domain. This method was applied with the same domain combinations as TALMUD. We used Mean Absolute Error as the evaluation metric (MAE).$MAE = \frac{\sum_{i=1}^{T}|P_i - R_i|}{T}$ where P_i is the predicted rating and R_i is the actual rating. Smaller MAE values indicate higher accuracy.

4.3 Experimental Results

The experiment results are reported in Table 3 and Table 4. Table 3 shows the results of the first experiment, where games is the target domain, while Table 4 shows the results of the second experiment where music is used as the target domain. In each section of the tables we present a scenario in which only two specific source domains are available. In each scenario we examine the effectiveness of using just one of the sources or the combination of the two sources. We tested the results for significance using a paired sign test. For each scenario, an asterisk (*) indicates that the method was significantly outperformed by TALMUD, whereas a plus (+) sign indicates that the method significantly outperformed TALMUD.

Examining the results in Table 3, we can see that when Jokes and Music are the available sources, using more than one source domain improves the prediction accuracy. The same trend is observed in the second scenario. These results reinforce our assumption that the use of multiple source domains can improve the accuracy of the prediction.

Table 3. MAE on Games Loads using jokes movies and music as source domains

CBT		TALMUD		Simple Linear Combination	
Jokes	Music	Jokes + Music		Jokes + Music	
		α=2.520	α=-1.331	α=0.5	α=0.5
0.914*	0.883*	**0.5644**		0.9092*	
Jokes	Movies	Jokes + Movies		Jokes + Movies	
		α=0.511	α=0.681	α=0.5	α=0.5
0.914*	0.533*	**0.4867**		0.9366*	
Music	Movies	Music + Movies		Music + Movies	
		α=-1.120	α=2.030	α=0.5	α=0.5
0.883*	**0.533**	0.6144		0.9068*	

Table 4. MAE on Music Loads using jokes, movies and games as source domains

CBT		TALMUD		Simple Linear Combination	
Jokes	Games	Jokes + Games		Jokes + Games	
		α=3.568	α=-1.606	α=0.5	α=0.5
0.907*	1.098*	**0.7957**		0.7960	
Jokes	Movies	Jokes + Movies		Jokes + Movies	
		α=0.419	α=0.660	α=0.5	α=0.5
0.907*	**0.7484**	0.7614		0.8358*	
Games	Movies	Games + Movies		Games + Movies	
		α=-0.489	α=1.631	α=0.5	α=0.5
1.098*	0.7484+	0.8208		**0.6743+**	

In some cases, the optimal relatedness coefficient for a source domain changes when we use this source together with different additional source domains. For example, in Table 3, when using jokes and movies as sources, the relatedness coefficient of jokes is significantly smaller compared to the use of jokes with music. It should be noted that the relatedness coefficient are learned for specific datasets of domains, but they do not represent a general relatedness level between the domains. The relatedness may be affected by various factors such as the data sparseness level, the interaction between the source domains, etc. Thus, the relatedness coefficient should be learned specifically for each problem at hand. In future work we intend to investigate the factors that the relatedness coefficient is influenced by. The meaning of a nonzero coefficient value, positive or negative, is that the source domain contributes to the model and balances the predicted error. A zero coefficient means that it is better not to use the source because it would have a negative impact on the model.

In the third scenario, when using music and movies as sources, the algorithm sets nonzero α_i values to both source domains, which means that the learner uses both sources, but it performs worse than a learner which uses a single source only. This is an example of the Negative Transfer phenomenon which the algorithm fails to recognize due to the well-known over fitting problem in machine learning [6]. As we add more source domains to the problem, the model becomes more complex and therefore adapts itself better to the training data. Thus, when building the model on the training data, it seems better to use multiple sources in order to improve prediction. But, when examining the results on the testing data, it is preferable to consider only one source domain. The same problem can be observed in Table 4. In the first scenario, when only games and jokes are the available datasets, the use of both source domains improves accuracy. But for the second and third scenarios using only one source domain outperformed the use of multiple source domains.

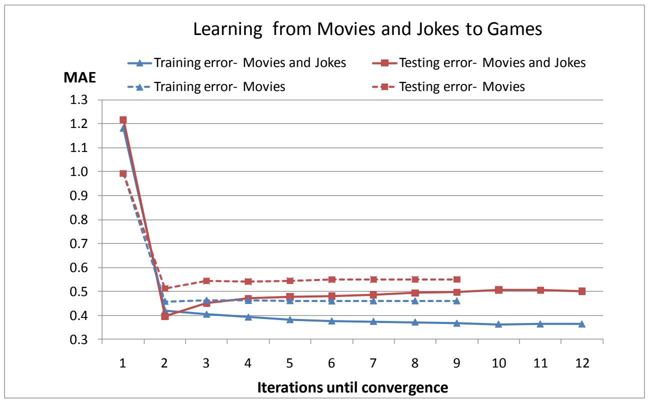


Figure 3. The learning process from a single domain (Movies) and multiple domains (Movies and Jokes)

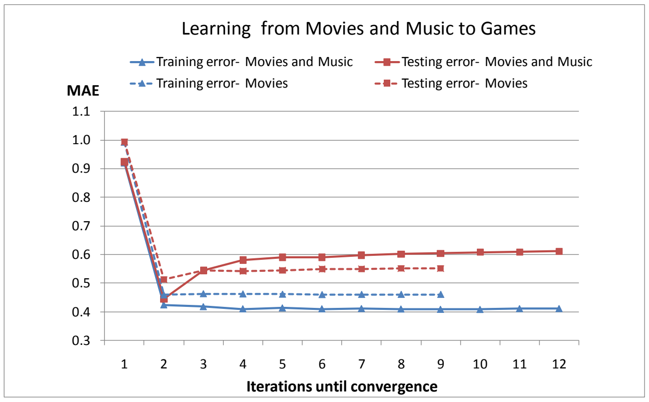


Figure 4. The learning process from a single domain (Movies) and multiple domains (Movies and Music)

The over fitting phenomenon is demonstrated using Figure 3 and Figure 4. In both figures, we compare the MAE results of the learning process from a single source domain (movies) and the results of learning from multiple domains (movies and jokes in Figure 3, and movies and music in Figure 4). At the end of each iteration we compute the MAE value that is achieved on the training data and on the testing data. We present for the learning process only the first iterations since the exact same trend continuous until convergence.

In Figure 3, observing the training curves, we see that the model that uses the two source domains achieves a smaller error on the training data, compared to the model that learns from one source domain only. This is also the case when examining the models on the testing data. The curve that describes multiple domains transfer learning is always below the curve that describes the single domain transfer learning, which means that the error is consistently smaller.

In Figure 4, however, a different trend is observed. While the use of multiple source domains is superior when measuring the MAE of training data, the accuracy on the testing data shows opposite results. That is, on the training data, the model that uses two sources is superior, while on the testing data, the model that uses a single source performs better.

The complexity of the model that uses two source domains leads to an over fitting of the model to the training data and degrades the accuracy of the resulting model on the testing data. We next propose an improved algorithm that mitigates the problem of over fitting and prevents negative transfer.

When comparing TALMUD to the simple linear combination for predicting games ratings (Table 3), it can be seen that TALMUD is significantly superior in all examined scenarios.

For predicting music (Table 4), TALMUD is superior in two scenarios out of three. When using games and movies to predict music, a smaller error is achieved while using a simple linear combination due to the over fitting problem. In the other scenarios TALMUD outperformed the simple linear combination significantly. This clearly demonstrates the value of learning the relatedness coefficients in most cases.

5. HEURISTIC DOMAIN ORDERING

In order to deal with the over fitting problem, we suggest an improvement to our algorithm. We propose a heuristic method that considers the trade- off between the knowledge that is gained by using multiple source domains and the over fitting problem that arises when the model complexity increases due to the additional source.

The method consists of two stages. First, it heuristically ranks the available sources from the most correlated to the target domain to the least correlated. Then, sources are added by decreasing heuristic correlation estimate until an over fitting problem is detected.

For computing the heuristic correlation estimate, the algorithm constructs a codebook for each domain using the method proposed by Ding et al. [7]. We then permute the rows and columns, and calculate the distance between each source matrix and the target matrix. The distance is computed by applying the codebook transfer phase of the CBT algorithm from the source domain codebook to the target domain codebook. Thus, the similarity between the domains' clusters (codebooks) is computed. The underlying heuristic assumption is hence that the source cluster that is most similar to the target cluster (has the smallest distance) is also most correlated to it.

After computing the heuristic estimates, the algorithm builds a model only from the most correlated source domain. Then, it tries to add more sources in decreasing heuristic estimate order. We use a wrapper method [11] in order to examine if using N source domains is better than using $N+1$ source domains. We build the model with only 80% of the training data and examine the error on the remaining 20% of the training data (often called the validation set). We then compare the error achieved by each model and decide whether to stop or not.

At the end of this process we identify the source domains that can be used for building the complete model without over fitting. Then, we use the entire training data to build the final complete model that will be used to predict ratings.

5.1 Experimental Results

We ran the improved algorithm with four domains, namely: games, music, movies and jokes as described above. In two separate runs we use games and music as target domains and the other three domains as optional sources from which the algorithm chooses the related sources that it is worth learning from. Here we also compare our results to results obtained by running the RMGM method [15] that aims to fill missing values in all sparse domains. For RMGM we use the same cluster level settings ($l = 20, k = 20$), and we examine the results obtained for the defined target domains.

Table 5 and Table 6 present the results of building a model for the games target domain while Table 7 and Table 8 present the results of building a model for the music target domain. Table 5 shows the MAE observed on the training data that are used to build the model, while Table 6 presents the results obtained from using the final model. The first row in Table 5 shows the similarity computation results between each of the codebook source domains and the codebook target domain. As can be seen in the table, Movies is the most correlated domain (has the smallest distance) to the target and Music is the least correlated, given our heuristic estimate. The rest of Table 5 (Rows 2, 3, 4) presents the decisions that the algorithm made at each step based on the computed MAE, where each row corresponds to an iteration of the algorithm. As explained above, on this stage the algorithm uses only the training data. The model is built using 80% of the training data and the error is computed using the validation set, which is the remaining 20% of the training data.

On the first step, the algorithm computes the MAE from learning only from Movies (the most correlated domain) using the CBT method [14]. Then, the algorithm adds the next correlated source domain, Jokes, and performs TALMUD for multiple source domains (row 3). The use of two source domains is preferable and obtains more accurate results (MAE=0.4864) than using one source domain. Therefore, the algorithm continues to add another source domain, Music, to the model (row 4). As can be observed in rows 3 and 4, the use of two source domains outperformed the use of three source domains. Thus, the algorithm selects only two source domains, Movies and Jokes, for the complete model construction.

Finally, the algorithm uses the complete training data to train the model using Movies and Jokes and computes the final MAE on the testing data. The final model is presented in Table 6. The results in Table 5 are consistent with results from Table 3. The most accurate predictions were achieved using the Movies and Jokes source domains. Given the three source domains we were able to reach this result without examining all possible combinations (e.g., movies and music, jokes & music, or each of the domains separately). The improved algorithm built the most accurate model given the three source domains.

On the games domain, however, the error on the validation set increases after adding the second source domain. As Table 7 shows, our algorithm indicates that if we start with Movies, and then add the Jokes source, then the MAE increase. Thus, it is better to use only a single source in this case – the movies domain. This is consistent with the results in Table 4, where the movies domain alone was superior to TALMUD**.**

Table 9 presents the results of comparing TALMUD to the RMGM method. Table 9 shows that TALMUD outperforms RMGM significantly for both target domains.

TALMUD heuristically chooses the source domain to learn from and does not necessarily use all the available related domains while RMGM builds the model from all source domains. Moreover, TALMUD aims to optimize predictions with regards to a single target domain while RMGM optimizes for all domains simultaneously.

Table 5. A demonstration of the algorithm performances of choosing the source domains when Games Loads is the target domain

	Method	Source domains			MAE on validation data
		Movies	Jokes	Music	
1	Heuristic correlation	0.0198	0.5286	0.6100	-
2	CBT	α=1	-	-	0.5338
3	TALMUD	α=0.67	α=0.50	-	**0.4864**
4	TALMUD	α=2.78	α=2.99	α=-4.76	0.6859

Table 6. Final model results for Games Loads

	Source domains		MAE on testing data
	Jokes	Movies	
TALMUD	α=0.511	α=0.681	**0.4867**

Table 7. A demonstration of the algorithm performances of choosing the source domains when Music Loads is the target domain

	Method	Source domains			MAE on validation data
		Movies	Jokes	Games	
1	Heuristic correlation	0.026	0.045	0.678	-
2	CBT	α=1	-	-	**0.7810**
3	TALMUD	α=0.693	α=0.373	-	0.7835

Table 8. Final model results for Music Loads

	Source domains	MAE on testing data
	Movies	
CBT	α=1	**0.7484**

Table 9. A comparison between the MAE results using TALMUD and RMGM

Target domain	TALMUD	RMGM
Games loads	**0.4867**	0.5458*
Music loads	**0.7484**	0.7806*

6. CONCLUSION AND FUTURE WORK

In this paper we presented TALMUD, a new multi-domain transfer learning method aimed at addressing the sparsity problem in recommender systems. The proposed method augments the codebook-based knowledge transfer (CBT) method which extracts knowledge from only one source domain to transfer knowledge from multiple source domains. Our method takes into consideration the possible interaction between the source domains, as well as the different degrees of relatedness between the sources and the target domain. It learns this relatedness and linearly integrates the rating patterns of all source domains into one model in order to enable prediction of the target matrix missing values. Although the complexity of building the model is high, it is a preprocessing stage that is performed offline that does not affect the recommendation time (and therefore should not pose a problem in practice). We address an applicative problem in which there is no guarantee of overlapping users or items between the domains, as is common in real-world scenarios. Our results show that using multiple source domains leads to more accurate predictions of the missing ratings. Thus, our method is useful for collaborative filtering applications. We further demonstrate that learning from multiple sources can lead to an over-fitting problem. We address this issue by heuristically choosing the best sources to learn from. The experimental results in the last section demonstrate the improvement in the accuracy gained by heuristically choosing the source domains according to their relatedness to the target domain.

In future work we intend to extend this research in the following directions:

An analytical proof of convergence. In this paper we assume Algorithm 1 convergence based on empirical evidence of our experiments. We demonstrate the convergence process in Figure 1 and Figure 2. We next intend to provide an analytical proof that Algorithm 1 monotonically decreases the objective function that is described in equation 1.

Evaluating the effect of data features. We further intend to examine the influence of different data features on the algorithm's performance. This will be done by extending the number of source domains as well as using larger datasets with different degrees of density.

Parameters calibration. The codebook dimensions, k and l, directly influence the quality of knowledge that is learned from the source domains. The number of the clusters should not be too small since it may cause a loss of knowledge, but using too many clusters can lead to over-fitting and is also computationally intensive. Thus, we intend to examine this tradeoff and formulate a rule of thumb that will help to determine the size of the codebooks.

Handling over fitting. We would like to further investigate the use of methods that may overcome the over fitting problem and compare them to our proposed solution.

Handling binary data. Prediction accuracy is the most discussed property in the recommendation system literature [18]. However, in industry most data is composed of binary preference or event data since users do not cooperate in providing explicit ratings. We would like to extend our method to support binary data.

7. REFERENCES

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

[2] J. Breese, D. Heckerman, C. Kadie, et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pages 43-52, 1998.

[3] B. Cao, N. Liu, and Q. Yang. Transfer learning for collective link prediction in multiple heterogeneous domains. In *Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel*. Citeseer, 2010.

[4] W. Dai, G. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210-219. ACM, 2007.

[5] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender Systems Handbook*, pages 107-144, 2011.

[6] T. Dietterich. Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, 27(3):326-327, 1995.

[7] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126-135. ACM, 2006.

[8] L. Getoor and C. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3-12, 2005.

[9] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 688-693. LAWRENCE ERLBAUM ASSOCIATES LTD, 1999.

[10] H. Kiers. Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems. *Computational statistics & data analysis*, 41(1):157-170, 2002.

[11] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 192 U-197, 1995.

[12] Y. Koren and R. Bell. Advances in collaborative filtering. *Recommender Systems Handbook*, pages 145-186, 2011.

[13] B. Li. Cross-domain collaborative filtering: A brief survey. In *2011 23rd IEEE International Conference on Tools with Artificial Intelligence*, pages 1085-1086. IEEE, 2011.

[14] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st international jont conference on Artificial intelligence*, pages 2052-2057. Morgan Kaufmann Publishers Inc., 2009.

[15] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 617-624. ACM, 2009.

[16] S. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345-1359, 2010.

[17] W. Pan, E. Xiang, N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the 24rd AAAI Conference on Artificial Intelligence,* 2010.

[18] G. Shani and A. Gunawardana. Evaluating recommendation systems. *Recommender Systems Handbook*, pages 257-297, 2011.

[19] G. Xue, W. Dai, Q. Yang, and Y. Yu. Topic-bridged plsa for cross-domain text classification. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 627-634. ACM, 2008.

[20] Y. Zhang, B. Cao, and D. Yeung. Multi-domain collaborative filtering. *In Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, California, USA*, 2010.