Making Recommendations from Multiple Domains

Wei Chen Wynne Hsu Mong Li Lee
School of Computing, National University of Singapore, Singapore
{weichen,whsu,leeml}@comp.nus.edu.sg

ABSTRACT

Given the vast amount of information on the World Wide Web, recommender systems are increasingly being used to help filter irrelevant data and suggest information that would interest users. Traditional systems make recommendations based on a single domain e.g., movie or book domain. Recent work has examined the correlations in different domains and designed models that exploit user preferences on a source domain to predict user preferences on a target domain. However, these methods are based on matrix factorization and can only be applied to two-dimensional data. Transferring high dimensional data from one domain to another requires decomposing the high dimensional data to binary relations which results in information loss. Furthermore, this decomposition creates a large number of matrices that need to be transferred and combining them in the target domain is non-trivial. Separately, researchers have looked into using social network information to improve recommendation. However, this social network information has not been explored in cross domain collaborative filtering. In this work, we propose a generalized cross domain collaborative filtering framework that integrates social network information seamlessly with cross domain data. This is achieved by utilizing tensor factorization with topic based social regularization. This framework is able to transfer high dimensional data without the need for decomposition by finding shared implicit cluster-level tensor from multiple domains. Extensive experiments conducted on real world datasets indicate that the proposed framework outperforms state-of-art algorithms for item recommendation, user recommendation and tag recommendation.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Search and Retrieval-*Information Filtering*

Keywords

Recommendation, Social trust, Personalization, Collaborative Filtering

KDD' 13, August 11–14, 2013, Chicago, Illinois, USA.

1. INTRODUCTION

With the increasing popularity of social media communities, we now have data repositories from various domains such as user-item-tag data from social tagging in book and movie domains, and friendship data between users in social networks. The joint analysis of information from various domains and social networks has the potential to improve our understanding of the underlying relationships among users, items and tags and increase user acceptance in recommender systems.

For example, users who like to read romance books generally have similar preferences as users who like to watch romance movies. By learning the characteristics of romance movie lovers from the Movie domain and transferring the learned characteristics to the Book domain, recommender systems can predict users' preferences more accurately and provide more customized recommendations.

Let us consider Tables 1 and 2 which show sample data from the Movie and Book domains respectively. Suppose we want to recommend some books to user u_1' in Table 2. Unfortunately, we cannot find similar users in the Book domain to base the recommendation on since u_1' is the only user who uses the tag fantasy. However, we can utilize the denser Movie domain dataset to learn the characteristics of users and make suitable recommendations for u_1'.

Recent works [1, 2] apply transfer learning methods to utilize data in some auxiliary domain such as Movie domain, and transfer knowledge that are consistent in this domain to a target domain such as Book domain. However, they are limited to transferring only binary relationships, e.g. user-item, in the form of matrices. Shi et al. [3] use tags as a bridge for cross domain transfer by decomposing the ternary user-tag-item relation into two binary relations user-tag and item-tag. However, this decomposition is lossy and may lead to inaccurate recommendations.

For example, the ternary relationship in the Movie domain (Table 1) can be decomposed into 3 binary relationships as shown in Table 3. Based on the binary User-Tag relationship in Table 3(a), we see that u_1' is similar to u_1 and u_2 because they all like fantasy items. Since u_1 and u_2 also watch other movies like comedy/action type movie 'Big Daddy' or 'Iron man', the work in [3] will look for comedy/action books to recommend to u_1', namely, 'Good omens', 'James Bonds Girls' and 'Ghost rider'.

However, if we take a closer look at the ternary relationship in both Table 1 and Table 2, we realize that the book 'New moon', read by u_1', has been tagged fantasy and romance. Between users u_1 and u_2, we observe that u_1 watches fantasy, romance and comedy type of movies, while u_2 watches fantasy, adventure and action type of movies. Thus, we conclude that u_1' is more similar to u_1 than u_2. Further, from the Movie domain, we realize that users who like fantasy and romance type of movies also like comedy movies. Thus, we should recommend comedy books "Good omens" to u_1'.

Table 1: Example Movie domain dataset

User	Tag	Item
u_1	fantasy	Twilight
u_1	romance	Twilight
u_1	comedy	Big Daddy
u_2	fantasy	Spider man
u_2	adventure	Spider man
u_2	action	Iron Man
u_3	comedy	Big Daddy
u_3	comedy	Little man
u_4	action	Iron Man
u_4	action	Star war
u_5	adventure	Die hard
u_5	adventure	Braveheart

Table 2: Example Book domain dataset

User	Tag	Item
u_1'	fantasy	New moon
u_2'	romance	New moon
u_3'	comedy	Good omens
u_4'	action	James Bonds Girls
u_5'	action	Ghost rider
u_5'	action	James Bonds Girls
u_5'	adventure	Scorpia

Table 3: Three binary relationships in Movie domain

(a) User-Tag

User	Tag
u_1	fantasy
u_1	romance
u_1	comedy
u_2	fantasy
u_2	adventure
u_2	action
u_3	comedy
u_4	action
u_5	adventure

(b) User-Item

User	Item
u_1	Twilight
u_1	Big Daddy
u_2	Spider man
u_2	Iron Man
u_3	Big Daddy
u_3	Little man
u_4	Iron Man
u_4	Star war
u_5	Die hard
u_5	BraveHeart

(c) Item-Tag

Item	Tag
Twilight	fantasy
Twilight	romance
Spider man	fantasy
Spider man	adventure
Iron Man	action
Star war	action
Big Daddy	comedy
Little man	comedy
Die hard	adventure
Braveheart	adventure

The above example illustrates the information loss when we decompose ternary relationships to binary relations for cross domain recommendation. This is because users may have different interests for an item, and items may have multiple facets. Although u_1 and u_2 use same tag fantasy, however, u_1's notion of fantasy is related to the romance aspects of the movies while u_2's notion of fantasy is on the adventure aspects. Thus, we advocate that recommendation using cross domain data should be carried out without decomposition.

Another major source of information that has yet to be fully utilized is that of social network data. Researchers have proposed to use data from the social network domain to increase user acceptance in recommender systems [4, 5]. The assumption is that the social network structure is useful for predicting users' preferences because users' interests may be affected by their friends. However, this assumption is not realistic as it implies that if two users, say u_i and u_j, are friends, then u_i will be influenced by u_j on all topics/aspects.

Let us consider users u_4' and u_5' in Table 2. Suppose we know from some social network website that u_4' is a friend of u_5'. If we assume that u_4' is influenced by u_5' to the same degree on all topics/aspects, then we will recommend the book 'Ghost rider' and 'Scorpia' to u_4' since they have been tagged by u_5' previously. However, u_4' may be friends with u_5' due to their common interests in action-related books, and u_4' may not like adventure books. Given the multi-facet nature of social trust among users, we advocate that trust is topic-specific and model the social relationship based on topics in our framework.

In this paper, we propose a tensor factorization based framework to fuse knowledge from different domains. We design a topic-based social trust regularization to integrate social network information with cross domain data. Our contributions are as follows:

- For cross domain data, we construct a shared three dimensional cluster level tensor as a bridge to uncover the hidden knowledge between the target domain and auxiliary domain. In particular, we extend tensor factorization to the setting of transfer learning.
- For social network information, we construct a shared users' latent feature space and design a topic based social trust regularization model, which has not been well studied in cross domain recommender systems.
- Experiments on real world datasets demonstrate the effectiveness of using multiple domains and social network for recommendation.

To the best of our knowledge, this is the first study that combines cross domain recommendation and social network in a unified framework.

The rest of this paper is organized as follows. Section 2 gives the problem formulation. Section 3 describes the unified framework. Section 4 presents the experimental results. We discuss related work in Section 5 and conclude in Section 6.

2. PROBLEM FORMULATION

A tensor is a multidimensional array. An N-order tensor $\boldsymbol{\mathcal{A}}$ is denoted as $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with elements $a_{i_1 \ldots i_n}$ and dimensions I_1,I_2, $\cdots$ I_N. Let the target domain dataset be a list of tuples <u, t, v> denoting that a user u tags an item v with tag t. We model this target domain dataset as a 3-order tensor $\boldsymbol{\mathcal{A}_{tgt}} \in U_{tgt} \times T_{tgt} \times V_{tgt}$, where U_{tgt} is the set of users, T_{tgt} is the set of tags, and V_{tgt} is the set of items/resources. $\boldsymbol{\mathcal{A}_{tgt}}(u, t, v)$ has a value of 1 if the tuple <u, t, v> exists, otherwise it has a value of 0.

For example, we can model the tagging activities of users in Table 1 as a 3-order tensor $\mathcal{A}$ with dimensions $5 \times 5 \times 8$. Entry $\boldsymbol{\mathcal{A}}$(1,1,1) has a value of 1 since it corresponds to the tuple

$<u_1$,'fantasy','Twilight'$>$ which is found in Table 1. On the other hand, the entry $\boldsymbol{\mathcal{A}}$(1,3,1) has a value of 0 since its corresponding tuple $<u_1$,comedy, 'Twilight'$>$ does not exist in Table 1.

Similarly, we model the dataset in the auxiliary domain as $\boldsymbol{\mathcal{A}_{aux}} \in U_{aux} \times T_{aux} \times V_{aux}$. Note that our proposed approach can handle the case when $U_{tgt} \cap U_{aux} = \emptyset$ and/or $V_{tgt} \cap V_{aux} = \emptyset$.

At the same time, suppose the users in the target domain are connected to each other via some social network. We model the user connections as a $U_{tgt} \times U_{tgt}$ trust matrix, $\boldsymbol{F} = [f_{u,w}]$ where $u, w \in U_{tgt}$ and $f_{u,w} \in [0, 1]$ denotes the degree of social trust that u has on w. A value of 0 implies u does not trust w while a value of 1 suggests that u trusts w completely.

We formulate the recommendation problem as a tensor missing value prediction problem. The goal is to generate a ranked list of users/items/tags based on the predicted value in the tensor. Here, we show how to extract the informative, yet compact cluster-level tensor (knowledge we want to transfer) from the auxiliary domain along with the mappings of users, items and tags between target and auxiliary domains, and the social trust knowledge in the target domain to enable better prediction results in the target domain. In other words, we want to predict the missing values in $\boldsymbol{\mathcal{A}_{tgt}}$ with knowledge from $\boldsymbol{\mathcal{A}_{aux}}$ and the trust matrix $\boldsymbol{F} = [f_{u,w}]$.

Let $\boldsymbol{\mathcal{A}^*_{tgt}}$ be the tensor obtained. Based on $\boldsymbol{\mathcal{A}^*_{tgt}}$, we can use it to perform the following recommendation tasks.

- Tag recommendation. This is to find the top-N tags that user u is most likely to use for an item v and can be derived from
$$\underset{t \in T_{tgt}}{\overset{N}{\operatorname{argmax}}} [\boldsymbol{\mathcal{A}^*_{tgt}}]_{u,t,v}$$
- Item recommendation. This task recommends the top-N items for user u based on the set of tags T_u s/he has used previously. The top-N items is determined from
$$\underset{v \in V_{tgt}}{\overset{N}{\operatorname{argmax}}} \sum_{t \in T_u} [\boldsymbol{\mathcal{A}^*_{tgt}}]_{u,t,v}$$
- User recommendation. This task recommends the top-N most likely friends for user u as follows:
$$\underset{u' \in \{U_{tgt} - \{u\}\}}{\overset{N}{\operatorname{argmax}}} \sum_{(u,t,v) \in \boldsymbol{\mathcal{A}_{tgt}}} [\boldsymbol{\mathcal{A}^*_{tgt}}]_{u',t,v}$$

3. UNIFIED FRAMEWORK

In this section, we first describe our approach to establish a bridge from the auxiliary domain to the target domain. Then we present our framework to fuse the social network information and the cross domain data to generate recommendations.

3.1 Cluster-Level Tensor

The key to a successful knowledge transfer from the auxiliary domain to the target domain lies in extracting the appropriate information from the auxiliary domain and establishing a mapping from the extracted knowledge back to the target domain. Here, the knowledge we want to extract are groupings of users, items, and tags that have similar characteristics. Our proposed method will construct a cluster tensor in the auxiliary domain. Then we will map the users, tags and items in the target domain to the clusters in the auxiliary domain.

We first perform a PARAFAC tensor decomposition on the auxiliary tensor $\boldsymbol{\mathcal{A}_{aux}}$. This decomposition maps users, items and tags into a shared latent feature space. In this shared space, we perform clustering to obtain groups of similar users, items, and tags.

The PARAFAC tensor decomposition [6] for a tensor $\mathcal{A}$ of size $I_1 \times I_2 \cdots \times I_N$ with an input rank R is:

$$\hat{\boldsymbol{\mathcal{A}}} \approx \sum_{j=1}^{R} [\hat{\boldsymbol{U}}^{(1)}]_{*j} \circ [\hat{\boldsymbol{U}}^{(2)}]_{*j} \circ \cdots [\hat{\boldsymbol{U}}^{(N)}]_{*j}$$

where $\hat{\boldsymbol{U}}^{(n)}$ of size $I_n \times R$ for $n = 1, ..., N$ and $[\hat{\boldsymbol{U}}^{(i)}]_{*j}$ denotes the j^{th} column of matrix $\hat{\boldsymbol{U}}^{(n)}$ and $||\boldsymbol{\mathcal{A}} - \hat{\boldsymbol{\mathcal{A}}}||_F^2$ is minimized. $||.||_F^2$ is the square frobenius norm and is defined as $||\boldsymbol{\mathcal{A}}||_F^2 = \sum_{i_1=1}^{R_1} \cdots \sum_{i_N=1}^{R_N} \boldsymbol{\mathcal{A}}(i_1 \cdots, i_N)^2$. $\circ$ is the outer product between vectors. The entry $\hat{\boldsymbol{\mathcal{A}}}(i_1, \cdots, i_N)$ is equal to $\sum_{j=1}^{R} [\hat{\boldsymbol{U}}^{(1)}]_{i_1 j} \times [\hat{\boldsymbol{U}}^{(2)}]_{i_2 j} \cdots \times [\hat{\boldsymbol{U}}^{(N)}]_{i_N j}$.

For the Movie dataset in Table 1, the PARAFAC tensor decomposition factorizes the auxiliary tensor $\boldsymbol{\mathcal{A}_{aux}}$ in the form of the latent feature representation $\hat{\boldsymbol{U}}^{(i)}$ $(1 \leq i \leq 3)$ as follows:

$$\boldsymbol{\mathcal{A}_{aux}} \approx \sum_{j=1}^{R=5} [\hat{\boldsymbol{U}}^{(1)}]_{*j} \circ [\hat{\boldsymbol{U}}^{(2)}]_{*j} \circ [\hat{\boldsymbol{U}}^{(3)}]_{*j}$$

where $[\hat{\boldsymbol{U}}^{(i)}]_{*j}$ denotes the j^{th} column of matrix $\hat{\boldsymbol{U}}^{(i)}$, and $\hat{\boldsymbol{U}}^{(1)} \in \mathbb{R}^{|U_{aux}| \times 5}$, $\hat{\boldsymbol{U}}^{(2)} \in \mathbb{R}^{|T_{aux}| \times 5}$, and $\hat{\boldsymbol{U}}^{(3)} \in \mathbb{R}^{|V_{aux}| \times 5}$.

The projection matrices $\hat{\boldsymbol{U}}^{(i)}$ $(1 \leq i \leq 3)$ obtained for the Movie dataset are as follows:

$$\hat{\boldsymbol{U}}^{(1)} = \begin{pmatrix} 0 & 0.53 & 0 & 1 & 0 \\ 0.53 & 0 & 1 & 0 & 0 \\ 0 & 0.85 & 0 & 0 & 0 \\ 0.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\hat{\boldsymbol{U}}^{(2)} = \begin{pmatrix} 0 & 0 & 0.71 & 0.71 & 0 \\ 0 & 0 & 0 & 0.71 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.71 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\hat{\boldsymbol{U}}^{(3)} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0.85 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.85 & 0 & 0 & 0 & 0 \\ 0 & 0.52 & 0 & 0 & 0 \\ 0.52 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.71 \\ 0 & 0 & 0 & 0 & 0.71 \end{pmatrix}$$

Based on the projection matrices, we apply some existing clustering algorithm to cluster the users, items, and tags. Table 4 shows the clusters obtained.

With this, we replace the ids of user, item and tag in the auxiliary dataset with their respective cluster id to obtain a cluster-level tensor, denoted as $\boldsymbol{\mathcal{A}^{cluster}_{aux}} \in \mathbb{R}^{R \times R \times R}$. Table 5 shows the cluster-level tensor obtained for the Movie dataset. The Val column is the normalized count of the duplicate tuples obtained after replacing the ids. We use this tensor to transfer the knowledge from the auxiliary domain (Movie) to the target domain (Book).

Transferring knowledge from $\boldsymbol{\mathcal{A}_{aux}}$ to $\boldsymbol{\mathcal{A}_{tgt}}$ is achieved through a reverse process of summarization in the auxiliary domain. By assuming that there exists implicit correspondence between the user/tag/item group of the auxiliary domain and those of the target domain. Based on the cluster-level tensor and the correspondence of user/tag/item group, we reconstruct the tensor $\boldsymbol{\mathcal{A}^*_{tgt}}$ as follows:

$$\boldsymbol{\mathcal{A}^*_{tgt}} = \boldsymbol{\mathcal{A}^{cluster}_{aux}} \times_1 \hat{\boldsymbol{U}^{(1)}_{tgt}} \times_2 \hat{\boldsymbol{U}^{(2)}_{tgt}} \times_3 \hat{\boldsymbol{U}^{(3)}_{tgt}} \qquad (1)$$

Table 4: Clusters for the Movie domain in Table 1

(a) Users

Cluster ID	Cluster
User-G1	{ u_1 }
User-G2	{ u_2 }
User-G3	{ u_3 }
User-G4	{ u_4 }
User-G5	{ u_5 }

(b) Tags

Cluster ID	Cluster
Tag-G1	{ fantasy }
Tag-G2	{ romance }
Tag-G3	{ comedy }
Tag-G4	{ adventure }
Tag-G5	{ action }

(c) Items

Cluster ID	Cluster
Item-G1	{ Twilight }
Item-G2	{ Big Daddy, Little man }
Item-G3	{ Spider man }
Item-G4	{ Iron man, Star war }
Item-G5	{ Die hard, Braveheart }

Table 5: Cluster-level tensor in Movie domain.

User	Tag	Item	Val
User-G1	Tag-G1	Item-G1	0.5
User-G1	Tag-G2	Item-G1	0.5
User-G1	Tag-G3	Item-G2	0.5
User-G2	Tag-G1	Item-G3	0.5
User-G2	Tag-G4	Item-G3	0.5
User-G2	Tag-G5	Item-G4	0.5
User-G3	Tag-G3	Item-G2	1
User-G4	Tag-G5	Item-G4	1
User-G5	Tag-G4	Item-G5	1

where $\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}} \in \mathbb{R}^{|U_{tgt}|\times R}$, $\hat{\boldsymbol{U}}^{(\mathbf{2})}_{\boldsymbol{tgt}} \in \mathbb{R}^{|T_{tgt}|\times R}$, and $\hat{\boldsymbol{U}}^{(\mathbf{3})}_{\boldsymbol{tgt}} \in \mathbb{R}^{|V_{tgt}|\times R}$ are user latent feature matrix, tag latent feature matrix and item latent feature matrix which we want to learn respectively, and $\times_n$ is the n-mode product. The **n-mode product** of a tensor $\mathcal{A}$ = $\mathbb{R}^{I_1\times\cdots\times I_N}$ by a matrix $\boldsymbol{U} = \mathbb{R}^{J_n\times I_n}$, denoted by $\boldsymbol{\mathcal{A}} \times_n \boldsymbol{U}$, is a $(I_1 \times I_2 \cdots I_{n-1} \times J_n \times I_{n+1} \cdots \times I_N)$-tensor where the entries are given by:

$$(\boldsymbol{\mathcal{A}} \times_n \boldsymbol{U})_{i_1 i_2 i_3 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 i_2 i_3 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} \cdot u_{j_n i_n}$$

To compute the optimal $\boldsymbol{\mathcal{A}}^{*}_{\boldsymbol{tgt}}$ for recommendation, we need to find the $\hat{\boldsymbol{U}}^{(\boldsymbol{i})}_{\boldsymbol{tgt}}$ (1≤i≤3) such that the difference between the observed tensor $\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}}$ and the reconstructed tensor $\boldsymbol{\mathcal{A}}^{*}_{\boldsymbol{tgt}}$ is minimized, that is,

$$\min_{\hat{U}^{(\mathbf{1})}_{\boldsymbol{tgt}} \cdots \hat{U}^{(\mathbf{3})}_{\boldsymbol{tgt}}} |||[\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}} - \boldsymbol{\mathcal{A}}^{*}_{\boldsymbol{tgt}}]|||^2_F \tag{2}$$

Table 6 shows the correspondence $\hat{\boldsymbol{U}}^{(\boldsymbol{i})}_{\boldsymbol{tgt}}$ (1≤i≤3) between the users, items and tags in the Book domain and the user, item and tag clusters in the Movie domain. The $Weight$ column indicates how similar a user/item/tag is to the cluster.

Suppose we want to recommend some books to user u'_1 in Table 2. User u_1 in the Movie domain forms a cluster User-G1. From Table 6, we observe that the mapping between user u'_1 and cluster User-G1 has a weight of 0.2, indicating that u'_1 has similar interests as the users in cluster User-G1. Since users in User-G1 like comedy movies 'Big Daddy', we may infer that user u'_1 may also like comedy books and thus recommend the book 'Good Omens' to u'_1.

Table 6: Mapping between Book and Movie domains.

(a) Users

User	Cluster ID	Weight
u'_1	User-G1	0.2
u'_2	User-G1	0.61
u'_3	User-G3	1.8
u'_4	User-G4	0.46
u'_5	User-G2	1.5

(b) Tags

Tag	Cluster ID	Weight
fantasy	Tag-G1	0.3
romance	Tag-G2	2.68
comedy	Tag-G3	0.50
adventure	Tag-G4	1.47
action	Tag-G5	1.24

(c) Items

Item	Cluster ID	Weight
New moon	Item-G1	1.22
Good omens	Item-G2	1
Scorpia	Item-G3	0.89
James Bonds Girls	Item-G4	1.24
Ghost rider	Item-G4	0.76

3.2 Fusing Social Network Information

Besides cross domain data, another valuable source of information is the social network information. Existing works on social recommendations [4][5] are all based on the assumption that friends in the social network will have similar interests in all topics and areas. They incorporate such a network-based similarity property among users to regulate the latent factor modeling as follows.

$$\begin{aligned} h &= \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{F}_{ij} |||[\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{i*} - [\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{j*}||^2 \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{F}_{ij} \left[\sum_{r=1}^{R} ([\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{ir} - [\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{jr})^2\right] \\ &= \sum_{r=1}^{R} [\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]^T_{*r} (\boldsymbol{D} - \boldsymbol{F}) [\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{*r} \\ &= tr([\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]^T (\boldsymbol{D} - \boldsymbol{F}) \hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}) \end{aligned} \tag{3}$$

where F_{ij} is the similarity between users u_i and u_j (defined in terms of either Vector Space Similarity (VSS) or Person Correction Coefficient (PCC) [5]), N is the number of users in the target domain, $\boldsymbol{D}$ is a diagonal matrix whose diagonal elements $\boldsymbol{D}_{ii}$=$\sum_j \boldsymbol{F}_{ij}$, and tr(·) denotes the trace of a matrix. The terms target to minimize the difference between the latent vectors of $[\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{i*} \in \mathbb{R}^{1\times R}$ and $[\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}]_{j*} \in \mathbb{R}^{1\times R}$ for all r topics ($1 \le r \le R$) with same weight.

Here, we want to differentiate user interest based on topics. We define a similarity matrix $\boldsymbol{F}^{(r)}$ for each topic r ($1 \le r \le R$), where R is the dimension of user latent feature $\hat{\boldsymbol{U}}^{(\mathbf{1})}_{\boldsymbol{tgt}}$. If users i and j are friends, then we define their similarity on a topic r, denoted

by $\boldsymbol{F}_{ij}^{(r)}$, as follows:

$$\boldsymbol{F}_{ij}^{(r)} = \frac{[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{ir}[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{jr}}{\sqrt{\sum_{k=1}^{R}[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{ik}}\sqrt{\sum_{k=1}^{R}[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{jk}}}$$

Otherwise their similarity $\boldsymbol{F}_{ij}^{(r)} = 0$.

We introduce the topic-based similarity function into the latent factor model and modify Eq (3) to the following:

$$\begin{aligned} h &= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{R}\boldsymbol{F}_{ij}^{(r)}||[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{ir} - [\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{jr}||^2 \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{F}_{ij}^{(r)}\left[\sum_{r=1}^{R}([\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{ir} - [\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{jr})^2\right] \\ &= \sum_{r=1}^{R}[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r}^{T}(\boldsymbol{D}^{(r)} - \boldsymbol{F}^{(r)})[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r} \\ &= tr([\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]^T(\boldsymbol{D}^{(r)} - \boldsymbol{F}^{(r)})\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}) \end{aligned} \quad (4)$$

where N is the number of users in the target domain, $\boldsymbol{D}^{(r)}$ is a topic-based diagonal matrix whose diagonal elements $\boldsymbol{D}^{(r)}{}_{ii}=\sum_j \boldsymbol{F}_{ij}^{(r)}$, and tr(·) denotes the trace of a matrix.

By combining Equations (2) and (4), we obtain the objective function for minimization:

$$\begin{aligned} \boldsymbol{f} &= \min_{\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}\cdots\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})}} ||\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}}^{*}||_F^2 \\ &+\lambda\cdot\textstyle\sum_{r=1}^{R} tr([\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]^T(\boldsymbol{D}^{(r)} - \boldsymbol{F}^{(r)})\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}) \end{aligned} \quad (5)$$

Equation (5) can be reduced to a non-negative tensor factorization problem with regularization [7]. We derive the multiplicative updating rules for $\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\boldsymbol{i})}$ $(1 \le i \le 3)$ as follows:

$$[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r} \leftarrow [\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r} \circledast \frac{[\boldsymbol{A}_{(1)}\boldsymbol{S}_{(1)}^T]_{*r} + \lambda\boldsymbol{F}^{(r)}[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r}}{[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^T]_{*r} + \lambda\boldsymbol{D}^{(r)}[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r}} \quad (6)$$

$$\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{2})} \leftarrow \hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{2})} \circledast \frac{\boldsymbol{A}_{(2)}\boldsymbol{S}_{(2)}^T}{\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{2})}\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^T} \quad (7)$$

$$\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})} \leftarrow \hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})} \circledast \frac{\boldsymbol{A}_{(3)}\boldsymbol{S}_{(3)}^T}{\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})}\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^T} \quad (8)$$

where $\boldsymbol{A}_{(n)}$ ($1 \le n \le 3$) is matrix unfolding of tensor $\boldsymbol{\mathcal{A}}$ at mode n, $\boldsymbol{S}_{(n)} = [\boldsymbol{\mathcal{A}}_{\boldsymbol{aux}}^{\boldsymbol{cluster}} \times_{m\neq n} \hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\boldsymbol{m})}]_{(n)}$, and $\circledast$ is the Hadamard product. The matrix unfolding of an N-order tensor $\boldsymbol{\mathcal{A}} = \mathbb{R}^{I_1\times\cdots\times I_N}$ along the dimension d are vectors obtained by keeping the index d fixed while varying the other indices and is denoted as $\boldsymbol{A}_{(d)}$. The Hadamard product of a matrix $\boldsymbol{U} = \mathbb{R}^{I\times J}$ by a matrix $\boldsymbol{V} = \mathbb{R}^{I\times J}$, denoted as $\boldsymbol{U} \circledast \boldsymbol{V} = \mathbb{R}^{I\times J}$ where the entries are given by

$$[\boldsymbol{U} \circledast \boldsymbol{V}]_{ij} = [\boldsymbol{U}]_{ij}\cdot[V]_{ij}$$

where $1 \le i \le I$ and $1 \le j \le J$.

These multiplicative update rules have stationary points at local minimum, and will not break the non-negativity constraint for the matrix $\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\boldsymbol{i})}$ $(1 \le i \le 3)$ [8]. The convergence of the above multiplicative update rules can be proven by using the auxiliary function method similar to the work in [8].

Based on the above multiplicative update rules, we design an iterative algorithm to obtain $\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\boldsymbol{i})}$ $(1 \le i \le 3)$ which minimize the objective function. Algorithm 1 shows the details. We observe that lines 5, 7 and 8 are the most time consuming steps. We exploit the sparsity of $\boldsymbol{A}_{(n)}$ and compute $\boldsymbol{A}_{(n)}[\boldsymbol{S}_{(n)}]^T$ in a single scan of nonzero elements of $\boldsymbol{A}_{(n)}$. Let N_1 denote the number of nonzeros elements in $\boldsymbol{A}_{(i)}$. Then the complexity of Algorithm 1 is $O(NumIter \times (R \times N_1 + (|U||V|+|U||T|+|V||T|) \times R^2)$. Our experiments show that $NumIter$ is typically less than 15.

Algorithm 1 FUSE

Input:
List of tuples <users, tags, items>; λ;
Cluster-level tensor $\boldsymbol{\mathcal{A}}_{\boldsymbol{aux}}^{\boldsymbol{cluster}} \in \mathbb{R}^{R\times R\times R}$
Output:
Tensor $\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}}^{*}$:
1: Initialization: From the tuple (users, items tag), we construct tensor $\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}} \in \mathbb{R}^{|U|\times|T|\times|V|}$, where $|U|$, $|V|$ and $|T|$ are the number of users, items and tags respectively
2: Random initialize $\left[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}\right]^0$, $\left[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{2})}\right]^0$ and $\left[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})}\right]^0$ to random nonnegative value.
3: **for** $t = 1\ to\ NumIter$ **do**
4: **for** $r = 1\ to\ R$ **do**
5: Update $[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})}]_{*r}^{t}$ using Equation (6).
6: **end for**
7: Update $[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{2})}]^t$ using Equation (7).
8: Update $[\hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})}]^t$ using Equation (8).
9: $\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}}^{*} \approx \boldsymbol{\mathcal{A}}_{\boldsymbol{aux}}^{\boldsymbol{cluster}} \times_1 \hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{1})} \times_2 \hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{2})} \times_3 \hat{\boldsymbol{U}}_{\boldsymbol{tgt}}^{(\mathbf{3})}$
10: **end for**

Table 7 shows the $\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}}^{*}$ obtained using cross domain information with social network. Note that the last 6 tuples are newly added. Previously, we are unable to recommend any books to u_1' since s/he is the only one who has used the tag 'fantasy'. However, the new tuple <u_1', *comedy*, *Good Omens*, 0.15> associates the book 'Good Omens' and the tag 'comedy' with user u_1' with a weight of 0.15. Thus, we can now recommend the comedy book 'Good Omens' to u_1'. In addition, although u_4' and u_5' are friends, 'Scorpia' is not recommended to u_4' since action is their only common topic of interest. With the new tuples, we can recommend 'Ghost rider' to u_4'.

Table 7: Output tensor $\boldsymbol{\mathcal{A}}_{\boldsymbol{tgt}}^{*}$

User	Tag	Item	Val
u_1'	fantasy	New moon	0.04
u_2'	romance	New moon	1
u_3'	comedy	Good Omens	0.97
u_4'	action	James Bonds Girls	0.72
u_5'	action	Ghost rider	1.17
u_5'	action	James Bonds Girls	0.72
u_5'	adventure	Scorpia	1
u_1'	romance	New moon	0.33
u_1'	comedy	Good Omens	0.05
u_2'	fantasy	New moon	0.11
u_2'	comedy	Good Omens	0.15
u_4'	action	Ghost rider	0.45
u_5'	fantasy	Scorpia	0.20

4. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed framework for recommendation. We implemented 3 versions of FUSE for the various recommendation tasks:

- **FUSE$^+$** utilizes topic-based social regularization
- **FUSE** does not utilize topic-based social regularization
- **FUSE$^-$** does not utilize social network information. This is achieved by setting $\lambda = 0$ for FUSE

We implement our framework in MATLAB and perform the experiments on a 2.33Ghz Intel Core 2 CPU with 4GB RAM, running Windows 7-64 bit. By default, $R = 50$ and $\lambda = 10$.

We use the following data sets in our experiments:

- MovieLens dataset[1] (Auxiliary domain): This is a publicly available dataset which comprises of two files. The first file contains users' tags on different movies. The second file contains users' ratings on different movies on a scale of 1 to 5, with 1 being bad and 5 being excellent. By joining these two files over user and movie, we obtain the quadruples $<user, movie, tag, rating>$. We have a total of 24563 quadruples with 2,026 users, 5,088 movies, and 9,078 tags. We pre-process these quadruples to generate a subset such that each user, movie and tag occur at least 10 times in the dataset. The resulting dataset has 24,185 tuples with 339 users, 982 movies, and 582 tags.
- LibraryThing dataset[2] (Target domain): Librarything is an online book review website. This dataset also comprises of two files. The first file contains users' tags and ratings on a scale of 1 to 5, with 1 being bad and 5 being excellent on different books. The second file contains users' trust statements on different users (binary value is recorded here to indicate the friendship). We have a total of 2,056,487 tuples with 7,279 users, 37,232 books, and 10,559 tags. We pre-process these tuples to generate a subset such that each user, book and tag occur at least 5 times. The resulting dataset has 402,246 tuples with 2,834 users, 2,768 books, 1,012 tags and 7,279 trust statements.

Table 8 summarizes the characteristics of these two datasets.

Table 8: Characteristics of datasets.

Statistics	Movie	Books
Users	339	2,834
Items	982	2,768
Tags	582	1,012
Social Relations	N.A	7,279
# of tuples	24,185	402,246

4.1 Experiments on Item Recommendation

We first evaluate the effectiveness of the proposed unified framework for item recommendation. We compare our methods with the following existing methods:

1. **UPCC** [9]. This method uses the Pearson's Correlation Coefficient to cluster similar users and recommend items based on these similar users.
2. **IPCC** [10]. This method uses the Pearson's Correlation Coefficient to cluster similar items for recommendation.
3. **TSA** [11]. This method recommends items based on the target domain data only, which is a ternary semantic analysis on users-items-tags.
4. **RMGM** [2]. This is a state-of-the art cross domain collaborative filtering algorithm that utilizes the user-item networks. Latent factor is set to 50.
5. **TagCDCF** [3]. This is a state-of-the art cross domain collaborative filtering algorithm that utilizes the tagging networks by reducing the three-dimensional correlations to two 2D correlations. Latent factor is also set to 50.

We use the Hit Ratio [10] as the metric to evaluate the effectiveness of the various item recommendation methods. For each user $u \in U$, we randomly choose one item v that has tagged by user previously and withhold the tuples involving u and v [10]. Then we run the various methods to generate the top N items recommended for this user. If the item v is among the top N recommended items, then we say that a hit has occurred. The hit ratio of a method is given by:

$$HitRatio = \frac{Number\ of\ hits}{|U|}$$

Figure 1(a) shows the effect of utilizing cross domain information for item recommendation. We observe that FUSE$^-$ consistently outperforms TagCDCF, RMGM, UPCC and IPCC as we vary N from 10 to 100. In particular, RMGM outperforms UPCC and IPCC, indicating that cross domain transfer of binary relationships (user-rating) can improve recommendation accuracy. Further, TagCDCF outperforms RMGM demonstrating that tag information is useful in cross domain recommendation. However, since TagCDCF requires the decomposition of ternary relationship into two binary relationships (user-item and item-tag), there is information loss resulting in reduced accuracy compared to FUSE$^-$.

Figure 1(b) shows the effect of utilizing social trust for item recommendation we vary N from 10 to 100. We observe that FUSE consistently outperforms FUSE$^-$ indicating the benefits of incorporating topic-specific social regularization. Further, FUSE$^+$ outperforms FUSE by an average of at least 8 % demonstrating that accurate modeling of topic-specific trust relationships leads to more accurate item recommendation.

Figure 1(c) shows the performance of FUSE$^+$ which combines both cross-domain and social trust information against existing recommendation algorithms such as UPCC, IPCC, and TSA. We observe that FUSE$^+$ is a clear winner, indicating that the joint analysis of cross domain information and social network are useful in understanding the users' interests better and providing better item recommendation compared to TSA which makes use of the social tagging network, and UPCC/IPCC which make use of the rating network only.

4.2 Experiments on Tag Recommendation

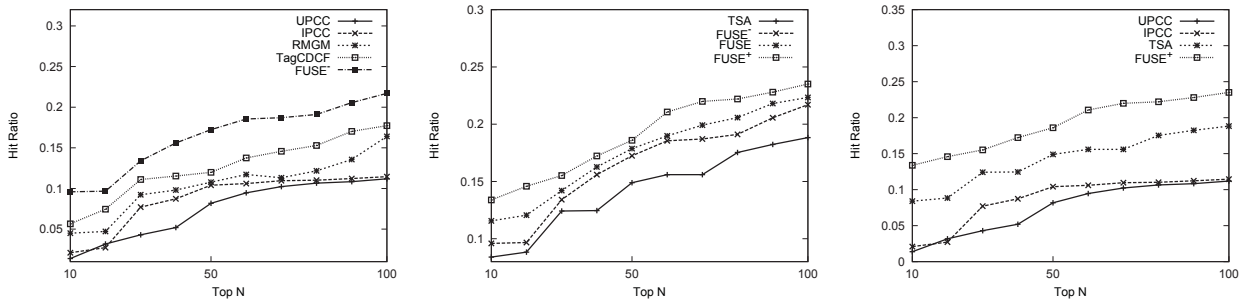
For the tag recommendation task, we evaluate our algorithm against two state-of-the-art methods: TSA [11] and RTF [12]. For each user $u \in U$, we randomly choose one item v and remove all tuples involving u and v from the dataset [12]. Then we run the methods to generate the top N tags recommended for this user.

We use the standard recall and precision measures to evaluate the results:

$$Precision = \frac{Number\ of\ Hits}{N}$$

$$Recall = \frac{Number\ of\ Hits}{|T_{u,v}|}$$

[1] http://www.grouplens.org/node/73
[2] http://www.librarything.com/

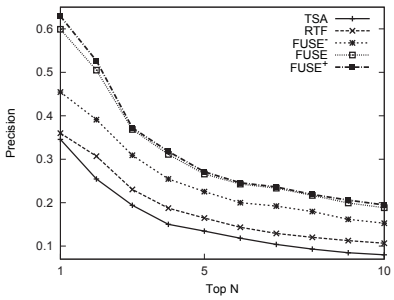


(a) Effect of cross-domain (b) Effect of social trust (c) Effect of cross-domain and social trust

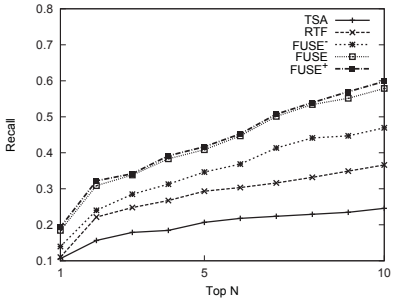
Figure 1: Item Recommendation

where $T_{u,v}$ is the set of tags used by user u on item v.

Figures 2(a) and 2(b) show the precision and recall of the methods for varying values of N. We see that FUSE$^+$ is able to achieve a higher recall and precision compared to the other three methods. FUSE$^+$ outperforms FUSE by 2.5% on average in both recall and precision, indicating that topic-specific trust regularization can improve tag recommendation compared to traditional trust regularization. Both FUSE$^+$ and FUSE outperform FUSE$^-$, indicating the effectiveness of incorporating social trust in tag recommendation. All our methods outperform state-of-the-art TSA demonstrating the effectiveness of using cluster-level tensor in transferring knowledge from the Movie domain to Book domain.



(a) Precision



(b) Recall

Figure 2: Tag recommendation

4.3 Experiments on User Recommendation

For user recommendation, we compare our algorithm with TSA [11]. For each user $u \in U$, we randomly choose one of his/her friend u_f and remove u_f from u's friendship list. Then we run the algorithms to generate the top N users recommended for this user. We use the standard recall measures to evaluate the results:

$$Recall = \frac{Number\ of\ Hits}{|U|}$$

Figure 3 shows the results for varying values of N. We observe that FUSE$^+$ achieves the best performance and outperforms FUSE by 10% on average, while the performance of FUSE$^-$ is very close to TSA. This confirms that both topic-specific trust and social network information are useful in user recommendation task.

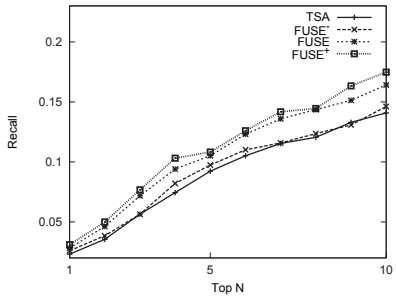


Figure 3: User recommendation

In order to evaluate the effectiveness of our methods in recommending interesting users, we first determine the similarity of items among the recommended top N users [13] since users with shared interests are more likely to tag and rate similar items and with similar friends. We compute the item similarity as the cosine similarity of their TF $\times$ IDF tag term vector [13].

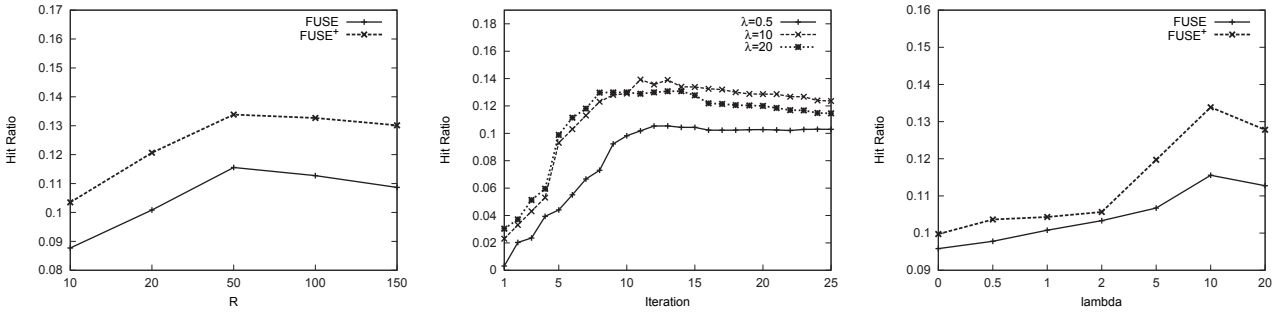
Let NB_u be the set of top N users recommended to u. The intra-neighborhood similarity is given by the average cosine similarity of all items for the users in NB_u:

$$IntraSim(NB_u) = \frac{\sum_{w \in NB_u} \sum_{i \in I_u, j \in I_w} sim(i,j)}{\sum_{w \in NB_u} |I_u||I_w|}$$

where I_u and I_w are the sets of items tagged by users u and w.

Let $Random_u$ be the set of N users randomly chosen from the set of users $U - \{u\}$. We can determine the inter-neighborhood similarity as follows:

$$InterSim(Random_u) = \frac{\sum_{w \in Random_u} \sum_{i \in I_u, j \in I_w} sim(i,j)}{\sum_{w \in Random_u} |I_v||I_w|}$$



(a) (b) (c)

Figure 4: Sensitivity Analysis

where I_u and I_w are the sets of items tagged by users u and w respectively.

Table 9 shows the intra-similarity and inter-similarity of FUSE$^+$ and TSA. We observe that the average intra-similarity is generally higher than the average inter-similarity for all the two methods. Furthermore, FUSE$^+$ have much higher intra-similarity and lower inter-similarity as compared to TSA. This indicates that more relevant users are found by FUSE$^+$ and hence lead to more accurate user recommendation.

Table 9: Intra- and inter- similarity between FUSE and TSA

Method	Intra-similarity	Inter-similarity
TSA	0.15	0.09
FUSE$^+$	**0.225**	**0.037**

4.4 Sensitivity Experiments

We also examine the effect of various parameters on the performance of Algorithm FUSE and FUSE$^+$ for item recommendation. Figure 4(a) shows the results as we vary the tensor dimension R. We observe that the proposed method FUSE$^+$ consistently outperforms the FUSE. This provides a evidence that the topic-based social recommendation is useful and can be used to improve the recommendation accuracy. We also find that the hit ratio of both FUSE and FUSE$^+$ increase as R increases, but decrease after $R = 50$ which may be caused by model over-fitting when the latent dimensions are large. Thus we set $R = 50$.

Figure 4(b) shows the hit ratio for various values of λ as we vary the number of iterations from 1 to 25. We observe that when we increase the iteration to be around 10, there seem to be little improvement for any large iteration. This suggests that a small number of iteration (such as 10) is enough for models. In other words, our algorithms typically converge after 10 iterations.

Figure 4(c) shows the impact of λ on the recall rate of our algorithms. Recall that the parameter λ control how much the information from social network will dominate the learning process. In the extreme case where $\lambda = 0$, the social network information is not used. As we can see from Figure 4(c), adopting a larger λ value can help to avoid the sparsity problem suffered by most MF-based CF methods. When we set $\lambda > 0$, we can achieve better results. This clearly demonstrates the impact of social network information, that is, adding more social network information can improve the generalization ability of the model. Moreover, Figure 4(c) also shows that the performance might degrade when λ is too large. In practice, we should choose a moderate value of λ. We observe that the best recall is obtained when $\lambda = 10$ indicating that social network information helps to improve item recommendation.

4.5 Scalability

Finally, we show the scalability of Algorithm 1 after mapping it to the MapReduce framework. The expensive operations in the algorithm are the matrix multiplication in the update formulae in Eq. (6), (7). Following the idea of [14], we implemented the MapReduce version of Algorithm 1 on our in-house cluster, Awan[3]. The cluster consists of 72 computing nodes, each of which has one Intel X3430 2.4GHz processor, 8GB of memory, two 500GB SATA hard disks and gigabit ethernet. On each node, we install CentOS 5.5 operating system, Java 1.6.0 with a 64-bit server VM, and Hadoop 0.23.6. All the nodes are connected via three high-speed switches.

We vary the dataset size from 2 million to 10 million by duplicating the users, items and tags in the original datasets and run the experiment by setting the model dimension R to 10 and 20 respectively. Figure 5 shows the results. We observe that the runtime increases linearly with respect to the dataset size for both $R = 10$ and $R = 20$. This shows that our algorithm is scalable with respect to the dataset size.

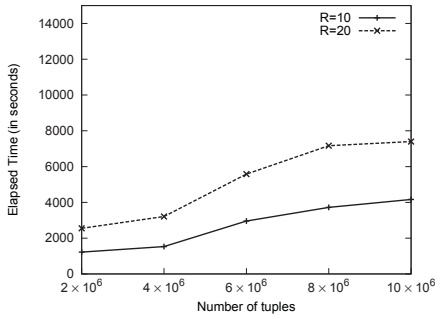


Figure 5: Scalability

5. RELATED WORK

Collaborative filtering (CF) in recommender systems can be roughly divided into two major categories. Memory-based methods aim at finding like-minded users to predict the active user's preference

[3]http://awan.ddns.comp.nus.edu.sg/ganglia/

[9][10][15]. Model-based methods [11][12][16] model the user-item-rating or user-item-tagging interaction based on the observed rating or tagging. However, in reality, such data is sparse as users tend not to give much rating or tagging information. Data sparsity is a major challenge for CF methods.

Recently, researchers consider using auxiliary data to address the data sparsity problem. One promising approach is the user-side knowledge transfer for trust-based recommendation [4][5][17][18]. The other approach is user-item side knowledge transfer using related but not aligned cross domain data for collaborative filtering [1][2][3].

For the trust-based recommendation, two different approaches have been proposed to compute trust: model-based [4][5][19], and memory-based [17][18]. Model-based approaches learn the parameters of a model to determine the trust between users, while memory-based approaches typically use exploration and heuristics. The works in [4][5] propose a matrix factorization with social regularization approach for social recommendation. [19] develop a joint personal and social latent factor (PSLF) model which combines the collaborative filtering and social network modeling approaches.

On the other hand, memory-based methods such as [17] perform a modified breath first search in the trust networks to compute a prediction. They find users with the shortest path from the source user and aggregates their ratings based on the weights (degree of trust) between the source user and them. [18] proposes a random walk method to combine trust-based and item-based recommendation. All these works assume single trust relationships between users. However, we have shown that trust is topic-specific.

For the cross domain collaborative filtering, researchers have tried to utilize knowledge of users' behavior in a different domain. These methods can be categorized into (a) binary relationships knowledge transfer [1][2]; and (b) ternary relationship knowledge transfer with decomposition [3]. For binary relationships knowledge transfer, [1] introduces a coordinate system transfer over multiple domains and transfer framework consisting of multiple data domains. These approaches share user/item latent feature spaces across CF domains and knowledge can be transferred through the shared latent features. [2] design a probabilistic method transfer for solving adaptive transfer learning problem in CF.

The work in [3] propose a matrix factorization based method use tags as bridge for cross domain transfer, by reducing the ternary relation to two 2D correlations and use these for regularization. A major difference from our work is that current cross domain recommendation systems can only deal with the transfer binary relationships knowledge such as users-items relationship [1][2] and/or ternary relationship with decomposition [3], while our method learns a ternary relation representation directly without information loss. To the best of our knowledge, this is the first work to integrate the cross domain and social trust for personalized recommendation.

6. CONCLUSIONS

In this work, we have presented a novel collaborative filtering method for integrating social network and cross domain network in a unified framework via latent feature sharing and cluster-level tensor sharing. This framework utilizes data from multiple domains and allows the transfer of useful knowledge from auxiliary domain to the target domain. The results of extensive experiments performed on real world datasets show that our unified framework outperforms the state-of-the-art techniques in all the three recommendation tasks. We have also implemented the algorithm on a map-reduce infrastructure and have demonstrated its scalability.

7. REFERENCES

[1] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang, "Transfer learning to predict missing ratings via heterogeneous user feedbacks." in *IJCAI*, 2011, pp. 2318–2323.

[2] B. Li, Q. Yang, and X. Xue, "Transfer learning for collaborative filtering via a rating-matrix generative model," in *ICML*, 2009, pp. 617–624.

[3] Y. Shi, M. Larson, and A. Hanjalic, "Tags as bridges between domains: improving recommendation with tag-induced cross-domain collaborative filtering," in *UMAP*, 2011, pp. 305–316.

[4] M. Jamali and M. Ester, "A transitivity aware matrix factorization model for recommendation in social networks." in *IJCAI*, 2011, pp. 2644–2649.

[5] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *WSDM*, 2011.

[6] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1253–1278, March 2000.

[7] Y.-D. Kim and S. Choi, "Nonnegative tucker decomposition." in *CVPR*. IEEE Computer Society, 2007, pp. 1 –8.

[8] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2000, pp. 556–562.

[9] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *CSCWC*, 1994, pp. 175–186.

[10] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems*, vol. 22, pp. 143–177, 2004.

[11] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis," *IEEE TKDE*, 2010.

[12] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *KDD*, 2009, pp. 727–736.

[13] C. Wei, W. Hsu, and M. L. Lee, "A unified framework for recommendations based on quaternary semantic analysis," in *SIGIR*, 2011, pp. 1023–1032.

[14] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," ser. WWW, 2010, pp. 681–690.

[15] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *SIGIR*, 2005, pp. 114–121.

[16] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *KDD*, 2008, pp. 426–434.

[17] J. A. Golbeck, "Computing and applying trust in web-based social networks," Ph.D. dissertation, 2005.

[18] M. Jamali and M. Ester, "Trustwalker: a random walk model for combining trust-based and item-based recommendation," in *KDD*, 2009, pp. 397–406.

[19] Y. Shen and R. Jin, "Learning personal + social latent factor model for social recommendation," in *KDD*, 2012, pp. 1303–1311.