Social-behavior Transfer Learning for Recommendation Systems

Qian Xu, Evan Wei Xiang and Qiang Yang
Hong Kong University of Science and Technology
Hong Kong, China
fleurxq@cse.ust.hk, wxiang@cse.ust.hk, qyang@cse.ust.hk

Abstract

Online recommendation systems are becoming more and more popular with the development of web. Data sparseness is a major problem for collaborative filtering (CF) techniques in recommender systems, especially for new users and items. In this paper, we try to reduce the data sparseness in the collaborative filtering problem by involving Wikipedia as an auxiliary information source. In this paper, we attempt to improve the recommendation accuracy by extracting collaborative social behavior information embedded in Wikipedia co-editing history, and use this knowledge to help alleviate the data-sparsity problem in CF. In addition, we introduce a parallel computing algorithm to help scale up the transfer learning process. Our experimental results on two real world recommendation datasets show that the social co-editing knowledge in Wikipedia can be effectively transferred for CF problems.

1 Introduction

Machine learning and data mining technologies have already achieved significant success in many knowledge engineering areas including Web search, computational advertising, recommender systems, etc. A major challenge in machine learning is the data sparseness problem. In the domain of online recommender systems, we attempt to recommend information items (e.g., movies, TV, books, news, images,web pages, etc.) that are likely to be of interest to the user. However, in many CF problems, the number of user preference values is small and the data are sparse. This can be caused by the large item spaces or new services. In such cases, overfitting can easily happen when we learn a model, causing significant performance degradation. To address such data sparseness problem, some service providers turn to explicitly ask the newly registered customers to rate some selected items, such as some most sparsely rated jokes in a joke recommender system [Nathanson *et al.*, 2007]. However, this approach may degrade the customers' experience and satisfaction with the system. Alternative methods, making use of the implicit user feedbacks [Hu *et al.*, 2008], may achieve good results, However, such tracking data also suffer from the sparseness for newly launched systems.

More recently, researchers have introduced transfer learning techniques to solve the data sparseness problem [Li *et al.*, 2009]. Transfer learning methods aim at making use of the data from other information sources, e.g., some other recommender systems, to help with our prediction tasks in the target domain. However, they require user preferences expressed in auxiliary and target domains to be homogeneous such as a common rating scale of 1-5. In practice, such homogeneous data from the auxiliary domains are also very hard to obtain.

Fortunately, the development of Web 2.0 provides us an opportunity to access and share information on Internet. In order to cope with the huge needs of information from hundreds of millions of web users, Web 2.0 based social applications become more and more popular, such as sharing sites (e.g., Flickr, Picassa, YouTube), blogs (e.g., Blogger, WordPress, LiveJournal), social networks (e.g., MySpace, Facebook), and social tagging systems (e.g. Delicious, CiteULike, Digg). Among these Web 2.0 based social applications, Wikipedia is the most important and successful example.

Our intuition in this work is to transfer some knowledge from Wikipedia to help solve the data sparseness problem for collaborative filtering tasks. One way is to use the content or concept structure information in Wikipedia as the source data, where we can build the similarity measure based on the content similarity between two articles or or via the Wikipedia hyperlink graph. However, we have found that the content-based similarity does not fully reflect the closeness of users' common interests on two products, which is important in CF problems. Instead, we have found that the co-editing behaviors of Wikipedia articles can better reflect users' interests in an implicit manner. For example, if there is a group of users who favor both Action and Fantasy movies, it is more likely that a group of editors to Wikipedia's action-movie articles are also willing to edit articles on fantasy movies. Such a similarity measure is obtained from the perspective of the closeness of the *social behavior* between two groups of users.

Thus, in this paper, we explore how to directly transfer social behavior knowledge from the knowledge base of Wikipedia to help with the prediction tasks in our target recommendation systems. We introduce a new algorithm, known as 'Collaborative Editing based Domain and Item Transfer' (COEDIT), to alleviate the problem of data sparseness prob-

lems in collaborative filtering.

2 Related Works

2.1 Collaborative Filtering

Generally speaking, two common approaches are usually exploited for collaborative filtering. One is the memory-based approach and the other is the model-based approach. Memory-based approach conducts certain forms of nearest neighbor search in order to predict the rating for a particular user-item pair. Amongst memory-based methods, the user-based model has been widely used to estimate the unknown ratings of a target user based on the ratings by a set of neighboring users that tend to have similar rating behavior to the target user. A crucial component of the user-based model is the user-user similarity for determining the set of neighbors. Popular similarity measures include the Pearson Correlation Coefficient(PCC) [Resnick *et al.*, 1994][Herlocker *et al.*, 2002] and the vector similarity(VS) [Breese *et al.*, 1998]. An alternative form of the memory-based approach is the item-based model [Sarwar *et al.*, 2001][Linden *et al.*, 2003], which compares items based on the ratings they received. When using the memory-based models, it is often difficult to reliably compute user or item similarities especially when the rating matrix is sparse. To alleviate the sparseness problem, different techniques such as dimensionality reduction [Goldberg *et al.*, 2001] and data-smoothing methods[Xue *et al.*, 2005][Ma *et al.*, 2007], have been proposed to fill in the unknown ratings in the matrix.

The model-based approach for collaborative filtering uses the observed user-item ratings to train a compact model that explains the hidden pattern of the given data. Models in this category include matrix factorization [Rennie and Srebro, 2005; Paterek, 2007][Koren *et al.*, 2009], probabilistic mixture models [Hofmann, 2004; Jin *et al.*, 2003], Bayesian networks [Pennock *et al.*, 2000] and restricted boltzman machine [Salakhutdinov *et al.*, 2007]. Previous studies showed that model-based approach such as matrix factorization is one of the best-performing solutions and achieves significant improvement over memory-based methods.

2.2 Transfer Learning

In machine learning community, the problem of utilizing data sets from related but different domains to build model for a target application domain is known as transfer learning [Pan and Yang, 2009]. Although transfer learning algorithms have achieved great success traditional data mining such as classification, regression, there are limited works on transfer learning in the context of collaborative filtering. Bhaskar and Hofmann [Mehta and Hofmann, 2007] considered two systems with shared users and then used manifold alignment methods to jointly build memory-based models for the two systems. Li et al. [Li *et al.*, 2009] designed a regularization function for jointly factorize two rating matrices with neither common users nor common items. However, all these existing works require the participating systems to share their rating matrix completely, which may be infeasible in practice. Moreover, the existing solutions only work with two systems and are difficult to generalize to multi-systems setting. Singh et al. [Singh and Gordon, 2008] proposed a collective matrix factorization model, where they simultaneously factor several matrices, sharing parameters among factors when an entity participates in multiple relations.The newest work done by Pan et al. [Pan *et al.*, 2010] demonstrated how a joint matrix factorization model can be used to transfer knowledge between heterogeneous source and target domains. However, they only focus on the knowledge transfer among close recommendation systems, and our focus is whether we could transfer the social behaviors on Web.

2.3 Data Mining with Wikipedia

In recent years, understanding and utilizing online knowledge repositories to aid real world data mining tasks have become a hot research topic. An example is to use the Wikipedia for feature enrichment. Gabrilovich et al. [Gabrilovich and Markovitch, 2005; 2007b] tried to use the Open Directory Project (ODP) for feature enrichment in the text classification problem. They also showed that using Wikipedia as the external Web knowledge resource for feature enrichment outperforms using ODP [Gabrilovich and Markovitch, 2006]. Gabrilovich et al. [Gabrilovich and Markovitch, 2007a] tries to explicitly represent the meaning of texts in terms of a weighted vector of Wikipedia-based concepts. Their semantic analysis is explicit in the sense that they manipulate manifest concepts grounded in human cognition, rather than latent concepts used by LSA. In [Phan *et al.*, 2008], a general framework, building classifiers with hidden topics discovered from large-scale data collections, was proposed. The framework is mainly based on latent topic analysis models such as PLSA [Hofmann, 1999] and LDA [Blei *et al.*, 2001], and machine learning methods such as maximum entropy and SVMs. The underlying idea of such a framework is that for each classification task, a very large external data, called by "universal dataset", is collected and then a classification model on both a small set of labeled training data and a rich set of hidden topics is built. Currently, a limited approaches employ auxiliary knowledge such as online knowledge database for transfer learning. Wang et al. made an extension [Wang *et al.*, 2008] to the feature-based transfer learning models by incorporating a semantic kernel [Wang *et al.*, 2007] [Wang and Domeniconi, 2008] learned from Wikipedia. Different from these traditional data mining works on Wikipedia, which focus on transferring content or structure information, our work tries to extract some collaborative social behavior information from the Wikipedia co-editing history.

3 Problem Description

Suppose that we have a target recommendation S_{tar} that is associated with m_{tar} users and n_{tar} items, denoted by $\mathcal{U}_{tar}$ and $\mathcal{V}_{tar}$, respectively. In the target system, we observe a sparse rating matrix $\mathbf{X}_{tar} \in \mathbb{R}^{m_{tar} \times n_{tar}}$ with entries $X_{tar,ij}$. Let $\mathcal{R}_{tar} = \{(i, j, r) : r = X_{tar,ij}, \text{where } X_{tar,ij} \neq 0\}$ denote the set of observed ratings in the target system.

In order to predict the unobserved ratings in S_{tar}, a popular method is to employ low-rank factorization to recover missing entries in $\mathbf{X}_{tar}$. We model the users $\mathcal{U}_{tar}$ and the items $\mathcal{V}_{tar}$ by a user factor matrix $\mathbf{U}_{tar} \in \mathbb{R}^{k \times m_{tar}}$ and an item factor matrix $\mathbf{V}_{tar} \in \mathbb{R}^{k \times n_{tar}}$, where the i-th user and j-th item

are represented by $\mathbf{u}_i$ and $\mathbf{v}_j$, corresponding to the i-th and j-th column of $\mathbf{U}_{tar}$ and $\mathbf{V}_{tar}$, respectively. The goal is to approximate the rating matrix $\mathbf{X}_{tar}$, i.e., $\mathbf{X}_{tar} \approx \mathbf{U}_{tar}^T \mathbf{V}_{tar}$. Under the low-rank factorization model, the factor matrices $\mathbf{U}_{tar}$ and $\mathbf{V}_{tar}$ can be learned by minimizing the following loss function:

$$\mathcal{L}_{tar} = \sum_{(i,j)\in\mathcal{R}_{tar}} \left(\mathbf{u}_i^T \mathbf{v}_j - X_{tar,ij}\right) + \lambda(\| \mathbf{U}_{tar} \|_F^2 + \| \mathbf{V}_{tar} \|_F^2), \quad (1)$$

where λ controls the trade-off between the rating matrix approximation errors and model complexity reflected by the Frobenius norm of the factor matrices.

However, for new recommendation services, users may rate few items, causing the matrix $\mathbf{X}_{tar}$ to be extremely sparse. Directly optimizing $\mathcal{L}_{tar}$ will suffer from severe overfitting problem.

As stated earlier, the historical editing logs may carry huge amount of user preference information. For example, if a user favors some products, it is likely that he will edit the product article on Wikipedia. Thus we also model Wikipedia as an information system S_{wiki}, which is associated with a set of m_{wiki} users and n_{wiki} items denoted by $\mathcal{U}_{wiki}$ and $\mathcal{V}_{wiki}$. In Wikipedia, the user editing activities are recorded and represented with a sparse matrix $\mathbf{X}_{wiki} \in \mathbb{R}^{m_{wiki}\times n_{wiki}}$. Let $\mathcal{R}_{wiki} = \{(i,j,r) : r = X_{wiki,ij}, \text{where } X_{wiki,ij} \neq 0\}$ denote the set of editing activities in Wikipedia, i.e., if user i edits article j in Wikipedia, then $x_{wiki,ij} = 1$. In contrast with $\mathbf{X}_{tar}$, $\mathbf{X}_{wiki}$ is much larger so that it may contain more information, which help us explore hidden patterns of user social behavior. Our intuition is to "borrow" user social behavior information from S_{wiki} to learn S_{tar} better.

4 Transfer Learning via Co-Edit Knowledge

4.1 Matrix Factorization in COEDIT

In order to achieve knowledge transfer from S_{wiki} to S_{tar}, we need to solve two related problems. First, we need to align the products in S_{tar} to some articles in S_{wiki} through the Search API in Wikipedia. Once the item correspondence is established, a subsequent problem is to build a compact model to fuse the information in S_{wiki} and S_{tar} together for knowledge transfer.

We can consider Wikipedia and our target recommendation system as two information systems S_{wiki} and S_{tar} with similar formation. The system s is associated with m_s users and n_s items denoted by $\mathcal{U}_s$ and $\mathcal{V}_s$, $s \in \{S_{wiki}, S_{tar}\}$, respectively. For each system s, we observe a sparse rating matrix $\mathbf{X}_s \in \mathbb{R}^{m_s\times n_s}$ with entries $X_{s,ij}$. Let $\mathcal{R}_s = \{(i,j,r) : r = X_{s,ij}, \text{where } X_{s,ij} \neq 0\}$ denote the set of observed records in each system. In the first stage, we have already established some item correspondence between S_{wiki} and S_{tar} to serve as the information bridge for knowledge transfer. More specifically, this implies that $\mathcal{V}_{wiki} \cap \mathcal{V}_{tar} \neq \emptyset$. We refer the set of items as *shared items* denoted by $\widetilde{\mathcal{V}}$. Let $\mathcal{U}^* = \mathcal{U}_{wiki} \bigcup \mathcal{U}_{tar}$ and $\mathcal{V}^* = \mathcal{V}_{wiki} \bigcup \mathcal{V}_{tar}$ denote the union of the collections of users and items, respectively, where $m^* = |\mathcal{U}^*|$ and $n^* = |\mathcal{V}^*|$ denote the total number of unique users and items in the union of two systems.

In order to derive a compact model to capture the user behavior information in both systems, we introduce the Item Bridged Joint Matrix Factorization (COEDIT) model. Under COEDIT, we model the users $\mathcal{U}^*$ and the items $\mathcal{V}^*$ by a user factor matrix $\mathbf{U} \in \mathbb{R}^{k\times m^*}$ and an item factor matrix $\mathbf{V} \in \mathbb{R}^{k\times n^*}$, where the i-th user and j-th item are represented by $\mathbf{u}_i$ and $\mathbf{v}_j$ corresponding to the i-th and j-th column of $\mathbf{U}$ and $\mathbf{V}$, respectively. Let $\mathbf{U}_s \in \mathbb{R}^{k\times m_s}$ denote the matrix formed by the rows in $\mathbf{U}$ that correspond to $\mathcal{U}_s$. Similarly, let $\mathbf{V}_s \in \mathbb{R}^{k\times n_s}$ denote the matrix formed by the rows in $\mathbf{V}$ that correspond to $\mathcal{V}_s$. The goal is to approximate each rating matrix $\mathbf{X}_s$, that is $\mathbf{X}_s \approx \mathbf{U}_s^T \mathbf{V}_s$, $s \in \{S_{wiki}, S_{tar}\}$. Under the COEDIT model, the factor matrices $\mathbf{U}$ and $\mathbf{V}$ can be learned by minimizing the loss function as follows:

$$\mathcal{L} = \sum_{s\in\{S_{wiki},S_{tar}\}} (\alpha_s \sum_{(i,j)\in\mathcal{R}_s} \left(\mathbf{u}_i^T \mathbf{v}_j - X_{s,ij}\right)^2) + \lambda(\| \mathbf{U} \|_F^2 + \| \mathbf{V} \|_F^2), \quad (2)$$

where α_s is the weight of each system, and λ controls the trade-off between the rating matrix approximation errors and model complexity reflected by the Frobenius norm of the factor matrices.

In this way $\mathbf{X}_{wiki}$ and $\mathbf{X}_{tar}$ are jointly factorized and the set of factor matrices $\mathbf{V}_{wiki}, \mathbf{V}_{tar}$ for different systems becomes inter-dependent because the features of a shared item are required to be the same for knowledge sharing. If $\mathbf{X}_{tar}$ is sparse, we can still learn a good $\mathbf{V}$, benefitting from the auxiliary user behavior information carried by $\mathbf{X}_{wiki}$. Moreover, a better estimate of $\mathbf{V}$ can further improve the estimation of $\mathbf{U}_{tar}$.

4.2 COEDIT Learning Algorithm

In order to find the optimal solution for the COEDIT model, we can use the *alternating least squares* (ALS) algorithm[Zhou *et al.*, 2008] (shown in Algorithm 1) to minimize the loss function in Equation (2) with respect to $\mathbf{U}$ and $\mathbf{V}$. When one of the factor matrices is fixed, minimizing $\mathcal{L}$ with respect to the other factor matrix is equivalent to solving a least squares problem. We can easily compute the gradient of the loss function $\mathcal{L}$ with respect to different factors:

$$\nabla_{\mathbf{u}_{s,i}} \mathcal{L} = (\sum_{j\in\mathcal{V}_{s,i}} \mathbf{v}_j \mathbf{v}_j^T + \lambda E_k)\mathbf{u}_i - \sum_{j\in\mathcal{V}_{s,i}} \mathbf{X}_{s,ij}\mathbf{v}_j \quad (3)$$

$$\nabla_{\mathbf{v}_j} \mathcal{L} = (\sum_{s\in\{S_{wiki},S_{tar}\}} \alpha_s \sum_{i\in\mathcal{U}_{s,j}} \mathbf{u}_i \mathbf{u}_i^T + \lambda E_k)\mathbf{v}_j - \sum_{s\in\{S_{wiki},S_{tar}\}} \alpha_s \sum_{i\in\mathcal{U}_{s,j}} \mathbf{X}_{s,ij}\mathbf{u}_i \quad (4)$$

where E_k denotes a $k \times k$ identity matrix and $\mathcal{V}_{s,i}$ denotes the set of items rated or edited by user i in system s. Setting the gradient $\nabla_{\mathbf{u}_{s,i}} \mathcal{L}$ to zero, we obtain the following closed form expression for updating $\mathbf{u}_{s,i}$:

$$\mathbf{u}_{s,i} = \mathbf{A}_{s,i}^{-1} \mathbf{b}_{s,i} \quad (5)$$

where

$$\mathbf{A}_{s,i} = \sum_{j\in\mathcal{V}_{s,i}} \mathbf{v}_j \mathbf{v}_j^T + \lambda E_k \quad (6)$$

is a $k \times k$ matrix and

$$\mathbf{b}_{s,i} = \sum_{j \in \mathcal{V}_{s,i}} X_{s,ij} \mathbf{v}_j \tag{7}$$

is a k dimensional vector. Similarly, to update the item features $\mathbf{v}_j$, we fix the $\mathbf{U}$ and minimize $\mathcal{L}$ with respect to $\mathbf{v}_j$, which yields the following updating formulas:

$$\mathbf{v}_j = \mathbf{A}_j^{-1} \mathbf{b}_j \tag{8}$$

where

$$\mathbf{A}_j = \sum_{s \in \{S_{wiki}, S_{tar}\}} \alpha_s \sum_{i \in \mathcal{U}_{s,j}} \mathbf{u}_i \mathbf{u}_i^T + \lambda E_k \tag{9}$$

is a $k \times k$ matrix and

$$\mathbf{b}_j = \sum_{s \in \{S_{wiki}, S_{tar}\}} \alpha_s \sum_{i \in \mathcal{U}_{s,j}} \mathbf{X}_{s,ij} \mathbf{u}_i \tag{10}$$

is a k dimensional vector.

Algorithm 1 Alternating Least Squares

1: Initialize $\mathbf{U}$ and $\mathbf{V}$ with small random numbers
2: **while** $\mathcal{L}$ has not converged **do**
3: Update $\mathbf{V}$ using Equation (8)
4: Update $\mathbf{U}_{wiki}$ and $\mathbf{U}_{tar}$ using Equation (5)
5: **end while**

5 Parallel Learning for COEDIT

As we know, Wikipedia carries a huge amount of information than newly launched information systems. Thus, how to improve the efficiency of our learning algorithm is also an important issue. Recently, parallel computing algorithms are becoming more and more popular in large-scale data mining. Here we introduce a parallel learning algorithm based on the Peer-to-Peer Message Passing Interface (MPI) platform [Snir *et al.*, 1998], where our transfer learning model is implemented.

We introduce a learning algorithm implementation based on P2P communication protocol. According to the introduction of the ALS algorithm in Section 4, we may find that either of the updating procedure of $\mathbf{U}$ or $\mathbf{V}$ mainly contains three stages: i) calculate some statistics, e.g., $\mathbf{v}_j \mathbf{v}_j$ and $X_{s,ij} \mathbf{v}_j$ for $\mathbf{u}_{s,i}$; ii) collect and aggregate these statistics, e.g., Equations (9) and (10); iii) get the aggregated statistics and calculate the updated parameters, e.g., Equation (8). The stage i) and stage iii) are totally independent for different users or for different items, while only stage ii) is depending on the other entities, e.g., for each item, Equations (9) and (10) need to aggregate the statistics over a set of users.

According to above observation, we first divide the data into l blocks with equal size, and distribute them to the slave nodes. The slave nodes take over the calculation of the statistics or parameters, and the statistics aggregation for a subset of users or items. To achieve this goal, a naive method is to record which slave node takes charge of the aggregation for which user or item via a table. However, such naive method requires each slave node to keep a table, so that it is inefficient when m^* and n^* are increasing. Thus, we employ a hash function $H : n \to l$, which can easily map a user or item ID to its corresponding slave node for aggregation. The algorithm is shown as Algorithm 2.

Algorithm 2 Slave Node l P2P Procedure for COEDIT

1: Get the parameters of $\mathbf{U}_l$ and $\mathbf{V}_l$
2: **while** not instructed to terminate **do**
3: Compute the statistics of $\mathcal{M}_{l,i}^A$ and $\mathcal{M}_{l,i}^b$ using Equation (11) and (12)
4: $\mathbf{Sync_1}$: Send $\mathcal{M}_{l,i}^A$ and $\mathcal{M}_{l,i}^b$ to node $l' = H(i)$
5: $\mathbf{Sync_2}$: Receive $\mathcal{M}_{l'',i}^A$ and $\mathcal{M}_{l'',i}^b$ from other slave nodes and aggregate statistics using Equation (13) and (14)
6: $\mathbf{Sync_3}$: Send each slave node l'' the aggregated statistics
7: $\mathbf{Sync_4}$: Receive the aggregated statistics and update $\mathbf{u}_i$ using Equation (5)
8: Compute $\mathcal{M}_{l,j}^A$ and $\mathcal{M}_{l,j}^b$
9: $\mathbf{Sync_5}$: Send $\mathcal{M}_{l,j}^A$ and $\mathcal{M}_{l,j}^b$ to node $l' = H(j)$
10: $\mathbf{Sync_6}$: Receive $\mathcal{M}_{l'',j}^A$ and $\mathcal{M}_{l'',j}^b$ from other slave nodes and aggregate the statistics
11: $\mathbf{Sync_7}$: Send each slave node l'' the aggregated statistics
12: $\mathbf{Sync_8}$: Receive the aggregated statistics and update $\mathbf{v}_j$ using Equation (8)
13: **end while**

$$\mathcal{M}_{l,i}^A = \sum_{j \in \mathcal{U}_{l,i}} \mathbf{v}_j \mathbf{v}_j^T \tag{11}$$

$$\mathcal{M}_{l,i}^b = \sum_{j \in \mathcal{U}_{l,i}} X_{L,ij} \mathbf{v}_j \tag{12}$$

$$\mathbf{A}_i = \sum_l \mathcal{M}_{l,i}^A + \lambda E_k \tag{13}$$

$$\mathbf{b}_i = \sum_l \mathcal{M}_{l,i}^b \tag{14}$$

Then Equation (5) can be updated via $\mathbf{A}_i$ and $\mathbf{b}_i$

6 Experimental Results

6.1 Data Description

Movie Recommendation Datasets We conduct the experiments using datasets from two real-world recommender systems: Netflix and MovieLens. The MovieLens dataset consists of around 10 million ratings for 10,681 movies by around 71,000 users. We successfully align about 9,600 movies to the articles in Wikipedia by movie title matching. We randomly sample users' ratings for 10 times, resulting in 50,000 ratings in each time. The Netflix dataset contains over 100 million ratings from over 480 thousand users on around 17,770 movies. We also align about 11,000 movies to the articles in Wikipedia. We randomly split the 480k users into 10 folds.

Wikipedia Datasets In this paper, we incorporate Wikipedia as our external data source. Wikipedia is currently the largest knowledge repository on the Web, and the quality of its article content is remarkable due to the open editing strategy. In our experiments, we use the mirror of Wikipedia on Aug. 10, 2009. Over 10 million users register Wikipedia. There are over 20 million pages and about 3 million of them are content articles and there are over 300 million editing record, which implies each article is edited 15 times on average. Note that in cases where the different systems use different rating scales, an additional normalization step can be conducted to convert them into a common scale.

Evaluation Metrics We use Root Mean Square Error (RMSE) to evaluate the prediction quality of different models. The metric RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(X_{i,j} - \widehat{X}_{i,j})^2}{N}}, \qquad (15)$$

where $X_{i,j}$ and $\widehat{X}_{i,j}$ are the observed ratings and predicted ratings, respectively; and N is the total number of ratings included in the test set. Our objective is to determine, to what extent our methods can transfer the social behavior information from S_{wiki} to S_{tar}. Therefore, we evaluate the performance using the RMSEs computed on the test set of S_{tar}.

6.2 Effectiveness Test

In this section, we will evaluate the effectiveness of transferring social behaviors for collaborative filtering tasks in the target domain. For each of the two target domains, Netflix and MovieLens, We have prepared 10 folds of data. In each fold of data, we hold out 30% as the test data for evaluation and the remaining as the training set. In addition, to examine the effect of sparsity, we randomly sample a proportion training data to simulate the scenarios of different levels of sparsity. We define the density of a matrix as the ratio of known values over all matrix elements; sparsity is high when the density is low. The density varies from 0.1% to 0.9%. For parameter settings, we tune $k \in \{3, 5, 10, 15, 20\}$ and $\lambda \in \{1, 2, 5, 10, 20\}$ over the 10 folds, and report the results with best mean RMSEs. In the real-world scenario, we can set them via cross validation. For the domain weight α_s, because we already prepare the item correspondence between S_{tar} and S_{wiki}, we will not investigate the effect of domain difference in this work. For simplicity, we set $\lambda_{tar} = \lambda_{tar} = 1$. In the future work, we will further consider using some techniques for setting domain weights according to domain differences [Zhang *et al.*, 2010] .

Effectiveness on COEDIT for Knowledge Transfer In the first set of experiments, we evaluate our methods with several baseline collaborative filtering models (shown in Table 1). The first two baselines are the average filling method (AF) and latent factorization model (LFM) [Bell and Koren, 2007], which directly learn a model based on the training data in the target domain. We also include two baseline models: $T_{Content}$ considers the description articles of the movies in Wikipedia and uses the movie-word matrix to serve as $\mathbf{X}_{wiki}$ in COEDIT; T_{Link} considers the hyperlink structure of the movies in Wikepdia and uses the movie-neighborhood matrix to serve as $\mathbf{X}_{wiki}$ in COEDIT. Although these two models also transfer some knowledge from Wikipedia, either the content or the hyperlink can only reflect the topical relatedness between two items. In contrast, our model COEDIT$_{Edit}$ tries to transfer knowledge from Wikipedia in terms of social behaviors. The experimental results show that although the content based transfer learning methods $T_{Content}$ and T_{Link} may take some effect when the target domain is severely sparse, the improvements over non-transfer learning approaches, especially comparing with AF, are not so obvious. However, our method COEDIT$_{Edit}$ achieves a significant improvement on all of the target datasets, even when the target domain's density is larger than 0.7%.

In the second set of experiments, we try to answer the following question: will data sparsity in S_{wiki} affect knowledge transfer performance? As we mentioned above, in the first stage, we establish the item correspondence between S_{tar} and S_{wiki}. After that, we collected the users' editing activities on the movie articles. For Netflix, we align 11,000 movies edited by around 56,000 users, and for Movielens, we align 9,600 movies edited by about 54,000 users. Then we sort the users in a descending order according to the numbers of their edits. By cutting down different proportion of the tailed users, we can simulate different levels of sparsity for S_{wiki}. We vary the density level of S_{wiki} from 0.3% to 1.1%. The prediction performance is shown in Table 2. We can observe that when the density of S_{wiki} is low ($\leq$ 0.5%), which means S_{wiki} is very sparse, the effect of knowledge transfer is not so good. However, once the density increases to larger than 0.5%, the results of transfer learning become more and more stable.

Effectiveness on Wikipedia Data Selection In the above section, we only use the aligned movies articles together with their editing histories for knowledge transfer. In this section, we try to investigate the question: can we benefit more from an extended scope of social behaviors in Wikipedia? That means, besides the co-editing behavior on movie articles, we are interested in whether the user co-editing behavior on other related items in Wikipedia can help solve the data sparseness problem in the target recommendation systems. We try to extend the set of related items with three approaches.

The first approach is to extend the item set via co-editing bipartite graph. Our intuition is that, if two users belong to the same interest group, it is likely that they may co-edit other related items as well, e.g., books, music, etc. Thus, we first collect the set of users who edited the movie article in Wikipedia, and then we collect all the articles they have edited in Wikipedia before and add them to the item set $\mathcal{V}_{wiki}$. The editing records of the extended item set are used to form $\mathbf{X}_{wiki}$. In total, we retrieve about 100,000 items with around 20 million editing records for both Netflix and MovieLens, and the density of $\mathbf{X}_{wiki}$ is 0.31%. The second approach is to extend the item set via hyperlink graph in Wikipedia. The reason for choosing hyperlink is that, when users are editing an article of their interest, they might follow some hyper-

Table 1: Prediction performance of average filling (AF), latent factorization model (LFM), collective matrix factorization(CMF) and COEDIT. Numbers in boldface (e.g., **0.993**) are the best results among all methods

Mean and Std of RMSEs on Netflix					
Methods	Without Transfer		With Transfer		
Target density	AF	LMF	$T_{Content}$	T_{Link}	COEDIT$_{Edit}$
0.1%	1.005±0.003	1.021±0.006	1.019±0.007	1.011±0.005	**0.993**±0.003
0.3%	0.968±0.003	0.981±0.002	0.978±0.005	0.972±0.004	**0.945**±0.002
0.5%	0.930±0.002	0.957±0.002	0.951±0.004	0.945±0.004	**0.921**±0.002
0.7%	0.921±0.001	0.932±0.001	0.929±0.003	0.920±0.003	**0.894**±0.001
0.9%	0.918±0.001	0.900±0.001	0.899±0.002	0.891±0.002	**0.869**±0.001
Mean and Std of RMSEs on MovieLens					
Methods	Without Transfer		With Transfer		
Target density	AF	LMF	$T_{Content}$	T_{Link}	COEDIT$_{Edit}$
0.1%	1.041±0.004	1.057±0.009	1.051±0.009	1.045±0.007	**1.030**±0.005
0.3%	0.988±0.003	1.012±0.004	1.005±0.008	1.001±0.005	**0.963**±0.004
0.5%	0.956±0.002	0.983±0.003	0.973±0.005	0.968±0.004	**0.920**±0.003
0.7%	0.920±0.002	0.945±0.003	0.941±0.004	0.938±0.003	**0.885**±0.002
0.9%	0.912±0.002	0.894±0.002	0.890±0.003	0.888±0.003	**0.858**±0.001

Table 2: Prediction performance of COEDIT with different density of S_{wiki}. Numbers in boldface (e.g., **0.993**) are the best results among all the levels of density

Mean and Std of RMSEs on Netflix					
	Density of S_{wiki}				
Target density	0.3%	0.5%	0.7%	0.9%	1.1%
0.1%	1.007±0.005	1.001±0.005	0.998±0.004	0.995±0.004	**0.993**±0.003
0.3%	0.965±0.003	0.959±0.003	0.950±0.003	0.948±0.002	**0.945**±0.002
0.5%	0.940±0.002	0.937±0.002	0.930±0.002	0.925±0.002	**0.921**±0.002
0.7%	0.914±0.002	0.905±0.002	0.901±0.002	0.898±0.002	**0.894**±0.001
0.9%	0.896±0.001	0.886±0.001	0.873±0.002	0.870±0.001	**0.869**±0.001
Mean and Std of RMSEs on MovieLens					
	Density of S_{wiki}				
Target density	0.3%	0.5%	0.7%	0.9%	1.1%
0.1%	1.045±0.005	1.040±0.005	1.035±0.005	1.031±0.005	**1.030**±0.005
0.3%	0.976±0.004	0.971±0.004	0.967±0.004	0.965±0.004	**0.963**±0.004
0.5%	0.940±0.004	0.936±0.004	0.930±0.004	0.923±0.003	**0.920**±0.003
0.7%	0.935±0.004	0.923±0.004	0.910±0.003	0.896±0.002	**0.885**±0.002
0.9%	0.898±0.002	0.884±0.002	0.872±0.002	0.865±0.001	**0.858**±0.001

Table 3: Prediction performance of COEDIT with different sizes of extended items by user co-editing history from the Wikipedia. Numbers in boldface (e.g., **0.982**) are the best results among all the sizes

Mean and Std of RMSEs on Netflix					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	25,000	50,000	75,000	100,000
0.1%	0.993±0.003	0.989±0.003	0.985±0.003	0.983±0.003	**0.982**±0.003
0.3%	0.945±0.002	0.939±0.002	0.935±0.002	0.934±0.002	**0.933**±0.002
0.5%	0.921±0.002	0.917±0.002	0.913±0.002	0.910±0.002	**0.909**±0.002
0.7%	0.894±0.001	0.891±0.001	0.888±0.001	0.887±0.001	**0.886**±0.001
0.9%	0.869±0.001	0.865±0.001	0.863±0.002	0.862±0.001	**0.861**±0.001
Mean and Std of RMSEs on MovieLens					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	25,000	50,000	75,000	100,000
0.1%	1.030±0.005	1.018±0.005	1.015±0.005	1.010±0.005	**1.006**±0.005
0.3%	0.963±0.004	0.955±0.004	0.947±0.004	0.946±0.005	**0.944**±0.004
0.5%	0.920±0.003	0.916±0.003	0.912±0.003	0.911±0.003	**0.909**±0.003
0.7%	0.885±0.002	0.883±0.002	0.878±0.002	0.876±0.002	**0.875**±0.002
0.9%	0.858±0.001	0.849±0.001	0.847±0.001	0.845±0.001	**0.844**±0.001

Table 4: Prediction performance of COEDIT with different sizes of extended items by hyperlink graph from the Wikipedia. Numbers in boldface (e.g., **0.983**) are the best results among all the sizes

Mean and Std of RMSEs on Netflix					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	20,000	30,000	40,000	50,000
0.1%	0.993±0.003	0.988±0.003	0.985±0.003	0.984±0.003	**0.983**±0.003
0.3%	0.945±0.002	0.942±0.002	0.936±0.002	0.935±0.002	**0.934**±0.002
0.5%	0.921±0.002	0.919±0.002	0.916±0.002	0.913±0.002	**0.912**±0.002
0.7%	0.894±0.001	0.893±0.001	0.890±0.001	0.889±0.001	**0.888**±0.001
0.9%	0.869±0.001	0.864±0.001	0.863±0.002	0.863±0.001	**0.862**±0.001

Mean and Std of RMSEs on MovieLens					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	10,000	20,000	30,000	40,000
0.1%	1.030±0.005	1.020±0.005	1.012±0.005	1.010±0.005	**1.009**±0.005
0.3%	0.963±0.004	0.956±0.004	0.947±0.004	0.945±0.004	**0.944**±0.004
0.5%	0.920±0.003	0.915±0.003	0.913±0.003	0.912±0.003	**0.909**±0.003
0.7%	0.885±0.002	0.883±0.002	0.881±0.002	0.880±0.002	**0.879**±0.002
0.9%	0.858±0.001	0.848±0.001	0.847±0.001	0.846±0.001	**0.845**±0.001

links through the article. If some hyperlinks point to some interesting articles, they may click and edit these articles as well. Thus, we collect the Tier-1 neighborhood of the aligned movie pages for extending the item set. In total, for Netflix, we retrieved 55,000 items with 6 million editing records, and the density of $\mathbf{X}_{wiki}$ is 0.19%. For MovieLens, we retrieved 42,000 items with 5 million editing records, and the density of $\mathbf{X}_{wiki}$ is 0.22%.

The last approach is to extend the item set via categorical structure. The intuition is that, it is likely that users will edit the articles belonging to the same category. Thus, we first collect the category set of the aligned movie articles, i.e., 450,000 for Netflix and 380,000 for MovieLens. Then we filter out the categories including only one movie article. Finally, we add the articles under these remaining categories to the extended item set. To summarize, for Netflix, we retrieve about 42,000 items with around 4,300,000 editing records, and the density of $\mathbf{X}_{wiki}$ is 0.18%; for MovieLens, we retrieve about 30,000 items with around 2,000,000 editing records, and the density of $\mathbf{X}_{wiki}$ is 0.14%.

The experimental results for these three methods are shown in Tables 3, 4 and 5, respectively. We can observe that when the number of related items increases, the effect of transfer learning is further improved, although slowly. Comparing with the three extending methods, the co-editing approach is a bit better. That may imply that the related items that we are interested in mainly derive their similarity more from social behaviors rather than topical closeness.

6.3 Efficiency Test

In this section, we will test the efficiency of our parallel learning algorithm. We examine the run time of the Peer-to-Peer mode algorithm that runs on a single machine with all the data from S_{tar} and S_{wiki}. For the test bed, we used a 1Gb/s LAN based cluster of 8 servers with Intel 8-core 2.93GHz CPU and 24GB memory. Each of the master and slave processes is performed using one individual core. For testing data, we use one fold of Netflix with density 0.9% as S_{tar} together its co-editing graph extended S_{wiki}. Thus, in total, we have 10 million ratings from S_{tar} and 20 million editing records in S_{wiki}. We plot the speed up achieved along with the ideal case of linear speedup in Figure 1. We can see that the parallel algorithm based on the Peer-to-Peer communication protocol achieves nearly linear speedup.

7 Conclusions and Future Work

In this work, we proposed the algorithm COEDIT to transfer coediting knowledge in Wikipedia to solve the data sparseness problem in collaborative filtering tasks. Different from traditional data mining works on Wikipedia, which focus on transferring content or structure information, our work tries to extract some collaborative social behavior in the form of co-editing history. Our experimental studies clearly demonstrate that these social knowledge can effectively help solve the data sparseness problem in other target domains. In order to improve the efficiency of our learning algorithm, we also introduced a parallel algorithm implementation for model learning.

In the future work, we will study how other social knowledge can be used to help with the tasks in other domains. We would also investigate how to analyze the domain differences for source data selection.

References

[Bell and Koren, 2007] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*, pages 43–52, 2007.

[Blei *et al.*, 2001] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *Proceedings of the 14st Annual Conference on Neural Information Processing Systems, NIPS 2001, Vancouver, British Columbia, Canada, December 3-8*, pages 601–608, 2001.

[Breese *et al.*, 1998] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of UAI 1998*, pages 43–52, 1998.

Table 5: Prediction performance of COEDIT with different sizes of extended items by category graph from the Wikipedia. Numbers in boldface (e.g., **0.982**) are the best results among all the sizes

Mean and Std of RMSEs on Netflix				
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$			
Target density	0	20,000	30,000	40,000
0.1%	0.993±0.003	0.987±0.003	0.982±0.003	**0.982**±0.002
0.3%	0.945±0.002	0.942±0.002	0.937±0.002	**0.936**±0.002
0.5%	0.921±0.002	0.918±0.002	0.915±0.002	**0.913**±0.002
0.7%	0.894±0.001	0.893±0.001	0.890±0.001	**0.889**±0.001
0.9%	0.869±0.001	0.864±0.001	0.863±0.001	**0.862**±0.001

Mean and Std of RMSEs on MovieLens				
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$			
Target density	0	10,000	20,000	30,000
0.1%	1.030±0.005	1.019±0.005	1.015±0.005	**1.012**±0.005
0.3%	0.963±0.004	0.953±0.004	0.950±0.004	**0.949**±0.004
0.5%	0.920±0.003	0.916±0.003	0.914±0.003	**0.912**±0.003
0.7%	0.885±0.002	0.884±0.002	0.883±0.002	**0.881**±0.002
0.9%	0.858±0.001	0.848±0.001	0.847±0.001	**0.846**±0.001

[Gabrilovich and Markovitch, 2005] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI 2005, Edinburgh, Scotland, UK, July 30-August 5*, pages 1048–1053, 2005.

[Gabrilovich and Markovitch, 2006] Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of The 21th National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, AAAI 2006, Boston, Massachusetts, USA, July 16-20*, 2006.

[Gabrilovich and Markovitch, 2007a] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, Hyderabad, India, January 6-12*, pages 1606–1611, 2007.

[Gabrilovich and Markovitch, 2007b] Evgeniy Gabrilovich and Shaul Markovitch. Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization. *J. Mach. Learn. Res.*, 8:2297–2345, 2007.

[Goldberg *et al.*, 2001] Kenneth Y. Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[Herlocker *et al.*, 2002] Jon Herlocker, Joseph A. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4):287–310, 2002.

[Hofmann, 1999] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI 1999 Stockholm, Sweden, July 30-August 1*, pages 289–296, 1999.

[Hofmann, 2004] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[Jin *et al.*, 2003] Rong Jin, Luo Si, Chengxiang Zhai, and Jamie Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of CIKM 2003*, pages 309–106, 2003.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[Li *et al.*, 2009] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, pages 2052–2057, 2009.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[Ma *et al.*, 2007] Hao Ma, Irwin King, and Michael R. Lyu. Effective missing data prediction for collaborative filtering. In *Proc. of SIGIR 2007*, pages 39–46, 2007.

[Mehta and Hofmann, 2007] Bhaskar Mehta and Thomas Hofmann. Cross system personalization and collaborative filtering by learning manifold alignments. pages 244–259. 2007.

[Nathanson *et al.*, 2007] Tavi Nathanson, Ephrat Bitton, and Ken Goldberg. Eigentaste 5.0: constant-time adaptability in a recommender system using item clustering. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 149–152, New York, NY, USA, 2007. ACM.

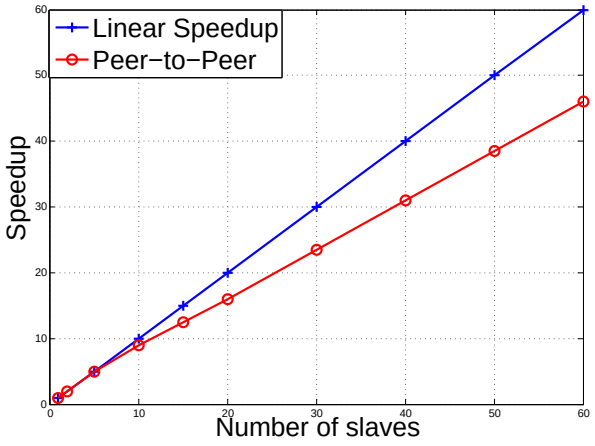

Figure 1: Speedups of the parallel learning algorithm

[Pan and Yang, 2009] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 99(1), 2009.

[Pan *et al.*, 2010] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, 2010.

[Paterek, 2007] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

[Pennock *et al.*, 2000] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proc. of UAI*, pages 473–480, 2000.

[Phan *et al.*, 2008] Xuan Hieu Phan, Minh Le Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25*, pages 91–100, 2008.

[Rennie and Srebro, 2005] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 713–719, New York, NY, USA, 2005. ACM.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[Salakhutdinov *et al.*, 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

[Sarwar *et al.*, 2001] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[Singh and Gordon, 2008] Ajit Paul Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[Snir *et al.*, 1998] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core*. MIT Press, Cambridge, MA, USA, 1998.

[Wang and Domeniconi, 2008] Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, Las Vegas, Nevada, USA, August 24-27*, pages 713–721, 2008.

[Wang *et al.*, 2007] Pu Wang, Jian Hu, Hua-Jun Zeng, Lijun Chen, and Zheng Chen. Improving text classification by using encyclopedia knowledge. In *Proceedings of the 7th IEEE International Conference on Data Mining, ICDM 2007, Omaha, Nebraska, USA, October 28-31*, 2007.

[Wang *et al.*, 2008] Pu Wang, Carlotta Domeniconi, and Jian Hu. Using wikipedia for co-clustering based cross-domain text classification. In *Proceedings of the 8th IEEE International Conference on Data Mining, ICDM 2008, Pisa, Italy, December 15-19*, pages 1085–1090, 2008.

[Xue *et al.*, 2005] Gui-Rong Xue, Chenxi Lin, Qiang Yang, Wensi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of SIGIR 2005*, pages 114–121, 2005.

[Zhang *et al.*, 2010] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. In *UAI*, pages 725–732, 2010.

[Zhou *et al.*, 2008] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *AAIM '08: Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management*, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.