Semi-supervised Collaborative Filtering by Text-enhanced Domain Adaptation

Wenhui Yu
Tsinghua University
Beijing, China
yuwh16@mails.tsinghua.edu.cn

Xiao Lin
Alibaba Group
Beijing, China
hc.lx@alibaba-inc.com

Junfeng Ge
Alibaba Group
Beijing, China
beili.gjf@alibaba-inc.com

Wenwu Ou
Alibaba Group
Beijing, China
santong.oww@taobao.com

Zheng Qin*
Tsinghua University
Beijing, China
qingzh@mail.tsinghua.edu.cn

ABSTRACT

Data sparsity is an inherent challenge in the recommender systems, where most of the data is collected from the implicit feedbacks of users. This causes two difficulties in designing effective algorithms: first, the majority of users only have a few interactions with the system and there is no enough data for learning; second, there are no negative samples in the implicit feedbacks and it is a common practice to perform negative sampling to generate negative samples. However, this leads to a consequence that many potential positive samples are mislabeled as negative ones and data sparsity would exacerbate the mislabeling problem.

To solve these difficulties, we regard the problem of recommendation on sparse implicit feedbacks as a semi-supervised learning task, and explore domain adaption to solve it. We transfer the knowledge learned from dense data to sparse data and we focus on the most challenging case — there is no user or item overlap.

In this extreme case, aligning embeddings of two datasets directly is rather sub-optimal since the two latent spaces encode very different information. As such, we adopt domain-invariant textual features as the anchor points to align the latent spaces. To align the embeddings, we extract the textual features for each user and item and feed them into a domain classifier with the embeddings of users and items. The embeddings are trained to puzzle the classifier and textual features are fixed as anchor points. By domain adaptation, the distribution pattern in the source domain is transferred to the target domain. As the target part can be supervised by domain adaptation, we abandon negative sampling in target dataset to avoid label noise.

We adopt three pairs of real-world datasets to validate the effectiveness of our transfer strategy. Results show that our models outperform existing models significantly.

CCS CONCEPTS

• **Information systems** → **Recommender systems**;

KEYWORDS

Transfer learning, domain adaptation, collaborative filtering, item recommendation, user reviews, adversarial learning.

ACM Reference Format:
Wenhui Yu, Xiao Lin, Junfeng Ge, Wenwu Ou, and Zheng Qin. 2020. Semi-supervised Collaborative Filtering by Text-enhanced Domain Adaptation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403264

1 INTRODUCTION

Recommender systems gain extensive research attention due to the general use on various online platforms like E-commerce and video websites. In real-world applications, implicit feedback data (one-class data like clicks and purchases) is widely used since it is easy to collect and generally applicable. Recommendation in real-world applications usually suffers from a serious sparsity issue, which leads to two difficulties: (1) There are not enough interactions to provide information for model learning. Moreover, the data is highly unbalanced: the majority of users and items only have a few interactions with the system, which makes the recommendation tasks more difficult. (2) Since we only observe a small part of positive samples in implicit feedbacks, existing negative sampling strategies treat unobserved samples as negative ones [6, 18, 21, 25, 27]. However, in this way, many potential positive samples are mislabeled as negative ones and the model is seriously misled by the label noise, especially on sparse data.

To tackle the aforementioned challenges, we adopt transfer learning to enrich the information on the sparse dataset and we focus on cross-domain recommendation without user and item overlap. To be specific, we train the source and target models on the dense and sparse data respectively and explore domain adaptation [4, 5] to align the embeddings (i.e., latent factors), which is the key component to encode user preferences in Collaborative Filtering (CF)

*The corresponding author.

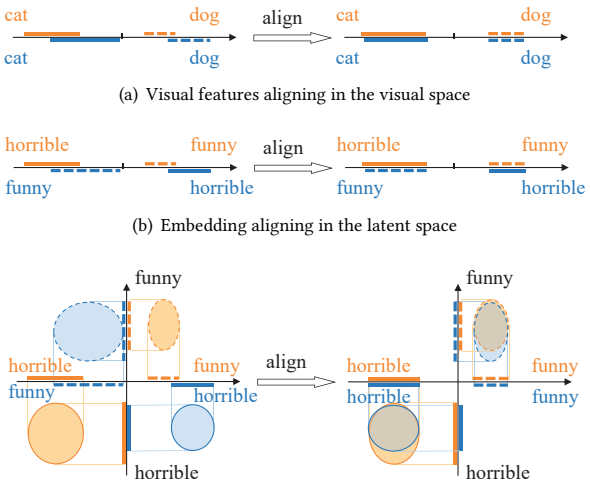
School of Software, Tsinghua National Laboratory for Information Science and Technology.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403264

models. We transfer the knowledge (such as distribution patterns) learned from the dense data to the sparse data to learn better embeddings on it. Considering the target model is supervised by both the positive samples and the domain adaptation mechanism, we abandon negative sampling on the sparse data to avoid the label noise issue. As we can see, learning the target model is a semi-supervised learning task.

To illustrate our strategy, we first give a brief introduction of domain adaptation. Ganin et al. [5] proposed Domain Adversarial Neural Network (DANN) for unsupervised image classification tasks. In DANN, a feature extractor is trained to extract visual features and a domain classifier is trained to classify which domain the current features come from. By adversarially training the feature extractor to puzzle the domain classifier, visual features of the two domains are aligned and the knowledge learned from the source domain is transferred to the target domain. More details of domain adaptation can be found in Subsection 2.1.1.



(a) Visual features aligning in the visual space

(b) Embedding aligning in the latent space

(c) Embedding aligning with textual features in the latent-textual space

Figure 1: Examples of domain adaptation in computer vision tasks and recommendation tasks. We use different colors to distinguish two domains and use solid and dotted lines to distinguish different categories. For concise illustration, the visual, latent, and textual spaces are all represented as 1 dimensional spaces.

In this paper, we aim to align embeddings by domain adaptation in recommendation tasks, however face a critical issue. We illustrate both vanilla DANN and our strategy in Figure 1. DANN is proposed for image classification tasks, which aligns high-level image representations in the visual space. As the two domains share the same feature extractor, images from the two domains are mapped into the same space, thus images with similar semantics are distributed in the similar position of the space. By domain adaptation, clusters with similar semantics are aligned together, and distribution patterns are transferred to refine the representations on the target domain. Taking the visual space represented in Figure 1(a) as an example, the negative and positive semi-axes encodes cats and dogs respectively, thus cats from different domain are both mapped to negative semi-axis and aligned together by domain adaptation.

However, in basic CF models, there is no data with specific semantics (such as images and text), and we extract high-level dense features by embedding users and items into the latent space. In this way, we map users and items from different domains into different latent spaces. Taking movies as an example in Figure 1(b), solid and dotted lines indicate horror movies and comedies respectively. As shown in Figure 1(b), aligning embeddings directly may cause misleading — horror movies from the "orange" domain and comedies from the "blue" domain are gathered and the distribution pattern are transferred incorrectly. The reason is that these embeddings are mapped into different latent spaces — on the "orange" domain, the negative and positive semi-axes encode horrible and funny respectively, and on the "blue" domain, we face the opposite situation. To address this gap, we need to conduct domain adaptation in the same space, i.e., we align the spaces as well as embeddings.

To align the latent spaces, we explore domain-invariant features as anchor points. In this paper, we leverage textual features, which can be easily extracted from the user reviews. An example is shown in Figure 1(c). We concatenate the textual features with the embeddings thus the space is extended to a textual-latent space (the transverse axis indicates the latent space and longitudinal axis indicates the textual space). In the latent space shown in Figure 1(b), different categories are not separable, while in Figure 1(c), different categories are separable by extending textual dimensions. For domain adaptation, we use the concatenated embeddings and textual features as the input of the domain classifier. Embeddings are trained adversarially with the classifier while textual features are fixed.

As we can see, the textual features in our strategy should be domain-invariant, e.g., horror movies from all domains are mapped to the negative semi-axis of the textual space. There are many existing models extracting textual features for recommendation [2, 30], while these features are not domain-invariant. To close this gap, we first propose a memory structure called **T**ext **M**emory **N**etwork (**TMN**) to extract textual features by mapping each user and item into the word semantic space. Then, we inject the features into a CF model to generate the prediction. The model consisting of the textual features and CF module is named as **T**ext **C**ollaborative **F**iltering (**TCF**) model. Finally, we train two TCF models on source and target domains synchronously, and connect them by an adaptation net. The transfer learning model is called **T**ext-enhanced **D**omain **A**daptation **R**ecommendation (**TDAR**) method.

Specifically, our contributions are listed as follows:

- We propose a domain adaptation recommendation method (TDAR) by aligning embeddings into a same latent space, which greatly improves the performances on the sparse datasets. To align the spaces and embeddings, we use the textual features as anchor points.
- As one important module in TDAR, we devise a memory network to extract domain-invariant textual features, and inject the features into a CF model to propose a text-based CF model.
- We devise comprehensive experiments on three pairs of real-world datasets to demonstrate the effectiveness of our

proposed methods. Codes are available on https://github.com/Wenhui-Yu/TDAR.

2 RELATED WORK

Recently, Recommender System (RS) has gained increasing attention due to its wide application on various online platforms. We model user preferences from the history interactions, and return personalized recommendation to each user. In various RS models, CF models mine collaborative information from the user-item interaction graph in a direct [7, 22] and a high-level [13, 20] way. Latent factor models [8, 13, 20], which are considered as a special kind of CF models, encode user preferences and item properties with embeddings, and measure the distance of embeddings in the latent space. To improve the representation capability, many variants have been proposed [6, 24, 26, 28, 30]. Though extensively studied, there is still a critical issue: RS suffers from serious sparsity problem and the performance on sparse data leaves much to be desired. In this paper, we aim to leverage transfer learning to enrich the information on sparse data and we use textual features as the anchor points, thus we introduce the most relevant aspects: cross-domain recommendation, and text-based recommendation in this section.

2.1 Cross-domain Recommendation

In this subsection, we first introduce the core technique — domain adaptation, and then introduce some related work about cross-domain recommendation with and without overlap.

2.1.1 Domain Adaptation. To learn on the data with fragmentary labels, or even without labels, transfer learning was proposed by transferring the knowledge learned from a well labeled source dataset to the target dataset [1, 4, 5, 15]. Ganin et al. [5] proposed DANN for unsupervised image classification tasks. Assume $\{\mathbf{x}_i^d, y_i^d\}_{i=1,\cdots,m;d=S}$ are labeled source data and $\{\mathbf{x}_i^d\}_{i=1,\cdots,n;d=T}$ are unlabeled target data. There are three parts in DANN: a feature extractor $G_f(\ ,\theta_f)$ (convolutional layers of CNN), a label predictor $G_y(\ ,\theta_y)$ (fully-connected layers of CNN), and a domain classifier $G_d(\ ,\theta_d)$. When training the model, θ_f and θ_y are updated to minimize the label prediction loss $\sum_i L(G_y(G_f(\mathbf{x}_i^S,\theta_f),\theta_y),y_i^S)$ on the source domain, θ_d and θ_f are trained to minimize and maximize the domain prediction loss $\sum_{i,d} L(G_d(G_f(\mathbf{x}_i^d,\theta_f),\theta_d),d)$ respectively. By adversarially training θ_d and θ_f, visual features from the two domains $G_f(\mathbf{x}_i^S,\theta_f)$ and $G_f(\mathbf{x}_i^T,\theta_f)$ are aligned and the knowledge learned from the source domain is transferred to the target domain.

2.1.2 Cross-domain Recommendation with Overlap. There are many models proposed for cross-domain recommendation with user and item overlap. Pan et al. [19] reduced the difference between source embeddings and target embeddings by directly minimizing the Frobenius norm of the difference. Lu et al. [16] proposed a selective transfer learning method which determines what to transfer based on a boost algorithm. Hu et al. [9] proposed two deep neural networks on source and target domains. By sharing user embedding layer, all users and items are mapped into the same latent space, and by constructing cross connections between the two networks, the parameters are transferred across domains. Yuan et al. [29] encoded each user by an auto-encoder and aligned the user representations by domain adaptation. Hu et al. [10] used items on the source domain which have interactions with current user to enhance the representation of the current item on the target domain, and used user reviews to improve the model performance. We can see that transfer learning is easy to achieve since there is user and item overlap. In this case, the bipartite user-item graphs of the two domains are different parts of a whole graph, and we can transfer the knowledge by simply sharing embeddings of the overlapped users and items between the two domains. However, if there is no overlap, two graphs are totally separate and embeddings are not sharable, thus we have to utilize more advanced approaches such as domain adaptation.

2.1.3 Cross-domain Recommendation without Overlap. There are several models for cross-domain recommendation without overlap. Kanagawa et al. [11] introduced an interesting task — recommending items from the source domain to users from the target domain based on text. To do so, textual features are extracted by an auto-encoder and aligned by domain adaptation. Wang et al. [23] proposed Long Short-Term Memory (LSTM) to construct user and item textual representations, and aligned them of the two domains for transfer learning. As we can see, these models [11, 23] achieve domain adaptation in the textual space since embeddings are difficult to align without overlap. In this case, cross-domain recommendation in [11, 23] is more close to a Natural Language Processing (NLP) task rather than a recommendation task. Li et al. [14] proposed a "codebook" method which transfers the rating pattern in the cluster level, nevertheless too coarse and empirical. Moreover, this method is based on users' rating pattern thus is difficult to be extended to implicit feedback situations. In this paper, we aim to refine embeddings which are key representations of CF models. To the best of our knowledge, this paper is the first work focusing on embedding aligning for cross-domain recommendation tasks without user and item overlap.

2.2 Text-based recommendation

Since we want to use textual features as the anchor points to align embeddings, we extract domain-invariant textual features for each user and item. In this subsection, we introduce some text-based recommendation models. The basic models introduced in [11, 23] are all text-based models. Kanagawa et al. [11] devised an auto-encoder and Kanagawa et al. [11] leveraged LSTM to extract user and item textual representations. Zheng et al. [30] gathered reviews for each user and item, and extract textual features from these reviews by a Convolutional Neural Network (CNN) and Chen et al. [2] further added an attention mechanism. However, textual features extracted by these existing models are not domain-invariant. Inspired by [10], we propose a memory network for textual features in this paper.

3 TEXT MEMORY NETWORK

In this paper, bold uppercase letters refer to matrices. Assuming there are M users and N items in total, we use matrix $\mathbf{R} \in \mathbb{R}^{M\times N}$ to denote the interactions between users and items. $\mathbf{R}_{ui} = 1$ if user u voted items i and $\mathbf{R}_{ui} = 0$ otherwise. Our task is to give prediction (denoted as $\hat{\mathbf{R}}$) of the missing values (0 in $\mathbf{R}$).

In this section, we construct user- and item-specific textual representation from the reviews. Please note that the model devised in this section is for signal domain and we extract textual features

for source and target domains separately. Taking user u as an example, we construct a review set $R_u = \bigcup_{i=1}^{N} r_{ui}$, where r_{ui} is a set containing words of u's review towards i. If u did not interact with i, $r_{ui} = \varnothing$. Similarly, the review set of item i is $R_i = \bigcup_{u=1}^{M} r_{ui}$. We use W to denote the set of words: $W = \bigcup_{u=1}^{M} \bigcup_{i=1}^{N} r_{ui}$, and use H to denote the total number of words: $H = |W|$. Since we want to extract domain-invariant textual features (i.e, features from all domains are in the same space), we map all users and items into the word semantic space by linearly combining word semantic vectors of the reviews. $\mathbf{S} \in \mathbb{R}^{H \times K_1}$ is the word semantic matrix[1] pretrained on GoogleNews corpus by word2vec [17], and $\mathbf{S}_w$ indicates semantic features of the word w. We use $\mathbf{E} \in \mathbb{R}^{M \times K_1}$ and $\mathbf{F} \in \mathbb{R}^{N \times K_1}$ to denote textual features we construct for users and items, respectively. Taking user u as an example, $\mathbf{E}_u = \sum_{w \in R_u} a_{uw} \mathbf{S}_w$, where a_{uw} is the weight of word w based on u's semantic preferences. We propose a Text Memory Network (TMN) to calculate weights for users $\{a_{uw}\}$ and for items $\{a_{iv}\}$ to construct textual features from word semantics.

For a user u which prefers horror movies, u may prefer relevant words (such as "horrible", "frightened", "terrifying"), and has no interest in irrelevant words (such as "this", "is", "a") and opposite words (such as "funny", "relaxing", "comical"). For the word w preferred by u, we need to set a large weight a_{uw} for w. In the item aspect, for a horror movie i, the relevant words in i's reviews provide much information about i and the irrelevant or opposite words provide little information. For a word v which is important to i, we need to set a large weight a_{iv}. It is easy to see that in this task, we aim to recommend preferred words to users and items.

Inspired by matrix factorization [13], we declare three matrices $\mathbf{P} \in \mathbb{R}^{M \times K_2}$, $\mathbf{Q} \in \mathbb{R}^{N \times K_2}$, and $\mathbf{T} \in \mathbb{R}^{H \times K_2}$ for users, items, and words, respectively. Taking users as an example, we use $e_{uw} = \mathbf{P}_u \mathbf{T}_w^{\top}$ to model u's preferences towards word w. To further emphasize important words, we input $\{e_{uw}\}$ to a softmax function to get $\{a_{uw}\}$: $a_{uw} = \frac{\exp(e_{uw})}{\sum_{w' \in R_u} \exp(e_{uw'})}$. Weights for items $\{a_{iv}\}$ are constructed in the same way. We finally predict user preferences towards items by $\hat{\mathbf{R}} = \sigma(\mathbf{E}\mathbf{F}^{\top})$, where $\sigma(\ )$ is the element wise sigmoid function, and we use cross entropy loss as our loss function:

$$\mathcal{L} = -\sum_{u,i} \mathbf{R}_{ui} \log \hat{\mathbf{R}}_{ui} + (1 - \mathbf{R}_{ui}) \log(1 - \hat{\mathbf{R}}_{ui}) + \lambda reg, \tag{1}$$

where

$$\hat{\mathbf{R}}_{ui} = \sigma\left[\left(\sum_{w \in R_u} \frac{\exp(\mathbf{P}_u \mathbf{T}_w^{\top})}{\sum_{w' \in R_u} \exp(\mathbf{P}_u \mathbf{T}_{w'}^{\top})} \mathbf{S}_w\right)\left(\sum_{v \in R_i} \frac{\exp(\mathbf{Q}_i \mathbf{T}_v^{\top})}{\sum_{v' \in R_i} \exp(\mathbf{Q}_i \mathbf{T}_{v'}^{\top})} \mathbf{S}_v\right)^{\top}\right],$$

and the regularization term reg is the Frobenius norm of model parameters $\{\mathbf{P}, \mathbf{Q}, \mathbf{T}\}$, which is learned by minimizing $\mathcal{L}$ with Adam [12]. In this text-based recommendation task, we get a tripartite graph of users, items, and words. When constructing the weights, we only leverage the connections of user-word and item-word. After constructing textual representations for users and items, we supervise the model by the user-item connections. As we can see, we indeed use three bipartite graphs in TMN.

Compared with existing CNN and RNN recommendation models [2, 23, 30], our textual feature extractor cannot model the sequential information while is good at highlighting important keywords. We consider that to extract interaction-specific textual information such as the task in [3], sequential information is important. However, to extract user- and item-specific textual information such as the task in [2, 23, 30] as well as in this paper, keywords are more important, since we want the textual features to summarize the preference elements of each user and item. Experiments also show that TMN performs very well in our task, especially in sparse cases.

[1] https://code.google.com/archive/p/word2vec

4 TEXT-ENHANCED CROSS-DOMAIN RECOMMENDATION

After extracting textual features by TMN, we introduce our Text-enhanced Domain Adaptation Recommendation (TDAR) model in this section. First, we inject the textual features into a CF model to propose the basic TCF model. Then, we train two TCF models (sharing the same interaction function, please see Figure 2) on target and source domains simultaneously, and align the user and item embeddings by domain adaptation.

4.1 Text Collaborative Filtering

In this subsection, we devise our basic model. $\mathbf{U} \in \mathbb{R}^{M \times K_3}$ and $\mathbf{V} \in \mathbb{R}^{N \times K_3}$ are embeddings for users and items respectively. As illustrated in Figure 1(c), we concatenate embeddings and textual features thus the representations of users and items are $[\mathbf{U}, \mathbf{E}]$ and $[\mathbf{V}, \mathbf{F}]$. We predict user preferences by these representations:

$$\hat{\mathbf{R}}_{ui} = f([\mathbf{U}, \mathbf{E}]_u, [\mathbf{V}, \mathbf{F}]_i, \mathbf{\Theta}),$$

where $f(\ , \mathbf{\Theta})$ is the interaction function combining user and item embeddings and returning the preference prediction, such as a deep structure [8, 24, 26], and $\mathbf{\Theta}$ indicates the parameters. We also learn the model by minimizing the loss function given in Equation (1). In this model, $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Theta}$ are trainable parameters while $\mathbf{E}$ and $\mathbf{F}$ are fixed. reg in Equation (1) indicates the Frobenius norms of $\mathbf{U}$ and $\mathbf{V}$.

4.2 Text-enhanced Domain Adaptation Recommendation

In this subsection, we use superscripts s, t, u and i to indicate source domain, target domain, user, and item, respectively. $\mathbf{R}^s \in \mathbb{R}^{M^s \times N^s}$ and $\mathbf{R}^t \in \mathbb{R}^{M^t \times N^t}$ indicate interactions on source and target domains. We train two TCFs on the two domains while share the same interaction function, thus the prediction on the two datasets is given by:

$$\hat{\mathbf{R}}^s_{u^s i^s} = f([\mathbf{U}^s, \mathbf{E}^s]_{u^s}, [\mathbf{V}^s, \mathbf{F}^s]_{i^s}, \mathbf{\Theta}), \tag{2}$$

$$\hat{\mathbf{R}}^t_{u^t i^t} = f([\mathbf{U}^t, \mathbf{E}^t]_{u^t}, [\mathbf{V}^t, \mathbf{F}^t]_{i^t}, \mathbf{\Theta}). \tag{3}$$

We then add adaptation networks on the two TCFs to achieve transfer learning. Considering the distribution pattern of user and item embeddings are possibly different, we conduct domain adaptation for users and items separately. Here we take users as an example. Assuming there are two user embedding distributions $dist(\mathbf{U}^s)$ and $dist(\mathbf{U}^t)$, we use a binary variable d_u^u as the domain label, which indicates whether $\mathbf{U}_u$ come from the target distribution or from the source distribution: $d_u^u = 1$ if $\mathbf{U}_u \sim dist(\mathbf{U}^t)$ and $d_u^u = 0$ if $\mathbf{U}_u \sim dist(\mathbf{U}^s)$. The superscript of d_u^u indicates that the domain label is for user embeddings and the subscript indicates that the domain label is for the current user u.

For the adaptation net, we leverage a domain classifier denoted as $g(\ , \mathbf{\Phi}^u)$, which is trained for domain classification: $\hat{d}_u^u = g([\mathbf{U}, \mathbf{E}]_u,$

Φ^u). To align embeddings, we want the distributions $dist(\mathbf{U}^s)$ and $dist(\mathbf{U}^t)$ to be similar. The most widely-used way is to train the domain classifier to discriminate between the two distributions, and train embeddings to puzzle the classifier [4, 5]. To be specific, we update Φ^u to minimize the loss of $g(\ ,\Phi^u)$ and update $\mathbf{U}^s$ and $\mathbf{U}^t$ to maximize it. In this way, the user embeddings of two domains are not separable therefore are aligned to the same distribution. Item embeddings are aligned in the same way.

We use $\mathcal{L}^s$ and $\mathcal{L}^t$ to denote prediction loss on the source domain and target domain, and use $\mathcal{L}^u$ and $\mathcal{L}^i$ to denote domain classification loss for users and items, respectively. $\mathcal{L}^s$, $\mathcal{L}^u$, and $\mathcal{L}^i$ are all cross entropy loss of the binary predictors. For $\mathcal{L}^t$, we only use positive labels as supervision on the target domain. The loss functions are listed as follows:

$$\begin{cases} \mathcal{L}^s = -\sum\limits_{u^s,i^s} \mathbf{R}^s_{u^s i^s} \log \hat{\mathbf{R}}^s_{u^s i^s} + (1-\mathbf{R}^s_{u^s i^s}) \log(1-\hat{\mathbf{R}}^s_{u^s i^s}) + \lambda^s reg^s \\ \mathcal{L}^t = -\sum\limits_{u^t,i^t} \mathbf{R}^t_{u^t i^t} \log \hat{\mathbf{R}}^t_{u^t i^t} + \lambda^t reg^t \\ \mathcal{L}^u = -\sum\limits_{u^s,u^t} \log \hat{d}^u_{u^t} + \log(1-\hat{d}^u_{u^s}) \\ \mathcal{L}^i = -\sum\limits_{i^s,i^t} \log \hat{d}^i_{i^t} + \log(1-\hat{d}^i_{i^s}) \end{cases}, \quad (4)$$

where, u^s and i^s are the user and item from the source domain, and u^t and i^t are that from the target domain. $\hat{\mathbf{R}}^s_{u^s i^s}$ and $\hat{\mathbf{R}}^t_{u^t i^t}$ are given in Equations (2) and (3). $\hat{d}^u_u$ and $\hat{d}^i_i$ are given by $\hat{d}^u_u = g([\mathbf{U},\mathbf{E}]_u, \Phi^u)$ and $\hat{d}^i_i = g([\mathbf{V},\mathbf{F}]_i, \Phi^i)$. Please note that the classifiers $g(\ ,\Phi^u)$ and $g(\ ,\Phi^i)$ share the same structure while with different parameters. reg^s and reg^t indicate the Frobenius norms of $\{\mathbf{U}^s,\mathbf{V}^s\}$ and $\{\mathbf{U}^t,\mathbf{V}^t\}$ respectively, and λ^s and λ^t are corresponding regularization coefficients. We update the model parameters as follows:

$$\Theta \leftarrow \Theta - \nabla_\Theta(\eta^s \mathcal{L}^s + \eta^t \mathcal{L}^t), \quad (5)$$

$$\mathbf{U}^s/\mathbf{V}^s \leftarrow \mathbf{U}^s - \nabla_{\mathbf{U}^s}(\eta^s \mathcal{L}^s - \eta^- \mathcal{L}^u)/\mathbf{V}^s - \nabla_{\mathbf{V}^s}(\eta^s \mathcal{L}^s - \eta^- \mathcal{L}^i), \quad (6)$$

$$\mathbf{U}^t/\mathbf{V}^t \leftarrow \mathbf{U}^t - \nabla_{\mathbf{U}^t}(\eta^t \mathcal{L}^t - \eta^- \mathcal{L}^u)/\mathbf{V}^t - \nabla_{\mathbf{V}^t}(\eta^t \mathcal{L}^t - \eta^- \mathcal{L}^i), \quad (7)$$

$$\Phi^u/\Phi^i \leftarrow \Phi^u - \eta^+ \nabla_{\Phi^u} \mathcal{L}^u/\Phi^i - \eta^+ \nabla_{\Phi^i} \mathcal{L}^i, \quad (8)$$

where η^s, η^t, η^+ and η^- are learning rates, and $\nabla_{\mathbf{X}} f(\mathbf{X})$ is the gradient of $f(\mathbf{X})$ with respect to $\mathbf{X}$. All parameters in TDAR are trained with Adam as well. The structure of TDAR is shown in Figure 2.

In TDAR, the two basic models are supervised by both labels and the domain classifier. As discussed in Introduction, negative labels on the target domain are contaminated by label noise seriously nevertheless the positive labels are pure, hence we abandon negative samples. Unfortunately, supervision with only positive samples in implicit feedback cases leads to a new problem — the model tends to predict all items as positive items. To deal with this issue, we leverage the domain adaptation mechanism to supervise the basic model on the target domain together with positive samples. However, as a dual-domain system, negative sampling for the whole system is necessary. Considering that the source domain is usually much denser than the target domain and the quality of negative labels is much higher, we adopt the source domain for negative supervision, and transfer the negative supervision to the target domain by domain adaptation.

For the textual features $\mathbf{E}$ and $\mathbf{F}$, we can pretrain and fix them during TDAR training and can also train them jointly with TDAR from scratch. Experiments show that joint training makes the model more difficult to tune yet achieves no performance improvement, so we choose the former strategy. Experiments also show that learning textual features is much more robust to label noise than learning user and item embeddings (please see Figure 3, especially obvious in Figures 3(c) and 3(d)). As a result, for the textual feature extractor — TMN, we train it dependently on the two domains without aforementioned transfer learning strategy. The motivation is that we aim to use the textual features to guide embedding aligning hence do not want to align them blindly. For example, textual representations of two different domains (such as movies and clothes) should be different.

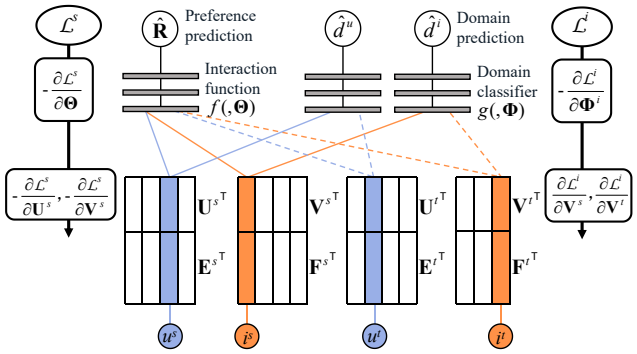


Figure 2: Illustration of TDAR. U and V are user and item embeddings and E and F are user and item textual features. $f(\ ,\Theta)$ is the interaction function and $g(\ ,\Phi)$ is the domain classifier. Lines indicate the forward propagation of the model and we use the solid lines and dotted lines to distinguish between different samples. Thick arrows indicate the backward propagation. We only show the prediction loss on the source domain $\mathcal{L}^s$ and classification loss for items $\mathcal{L}^i$ and corresponding gradients for concise illustration.

5 EXPERIMENT

In this section, we conduct experiments to validate the effectiveness of our model. We utilize three pairs of real-world datasets to compare our models against several state-of-the-art models, and report the experiment result. We focus on answering three following research questions:

RQ1. How is the performance of our TDAR model?

RQ2. How is the performance of our TMN and TCF models?

RQ3. How is the effectiveness of our text-enhanced domain adaptation strategy?

5.1 Experimental Setup

In this subsection, we introduce some details about experimental setup.

5.1.1 Datasets. We adopt *Amazon*[2] dataset, which is the user reviews collected from the E-commerce website *Amazon.com*, to evaluate our model. We select four categories of the *Amazon* dataset:

[2] http://snap.stanford.edu/data/amazon/productGraph/

Table 1: Statistics of datasets

Dataset	Interaction	User	Item	Sparsity	Source/Target
Movies	1,697,532	123,960	50,052	99.9726%	Source
Videos	11,137	4,555	1,580	99.8453%	Target
CDs	43,903	25,400	24,904	99.9931%	Target
Clothes	89,176	35,669	21,554	99.9884%	Target

Movies, *Videos*, *CDs*, and *Clothes*. Statistics of these datasets are illustrated in Table 1. All datasets are filtered by 5-core (removing items and users less than 5 records). We select the largest one — *Movies* as the source domain and the others as target domains. Since we want a dense source dataset, we filter *Movies* by 30-core to increase the density. For target datasets, we want them to simulate real-world applications which contain many cold users and items (with less than 5 records). To close this gap, we randomly delete a part of interactions in the three target datasets. Since *Amazon* is explicit feedback data (ratings), we set the interaction (u, i) as "1" if u has rated i and "0" otherwise to construct implicit feedbacks. Each dataset is split into training set (80%), validation set (10%), and test set (10%) randomly. We train models on the training set; determine all hyperparameters on the validation set; and report the performance on the test set. The target datasets are selected on consideration of different sparsity and similarity to the source dataset. To our intuition, *Videos*, *CDs*, and *Clothes* are very similar, relatively similar, and different to *Movies*, respectively.

5.1.2 Model Setting. $g(\ )$ is a deep fully-connected (FC) structure with 4 layers. The size of input layer is $K_1 + K_3$ and of FC layers is 64. For the interaction function $f(\ , \mathbf{\Theta})$, we adopt the inner product activated by sigmoid, i.e., $f(\mathbf{U}_u, \mathbf{V}_i, \mathbf{\Theta}) = \sigma(\mathbf{U}_u \mathbf{V}_i^\top)$, where $\mathbf{\Theta} = \varnothing$. We choose this shallow and simple structure since we gain no performance enhancement by deep recommendation models in our experiments (please see Figure 3 for details).

5.1.3 Baselines. We adopt the following methods as baselines for comparison:

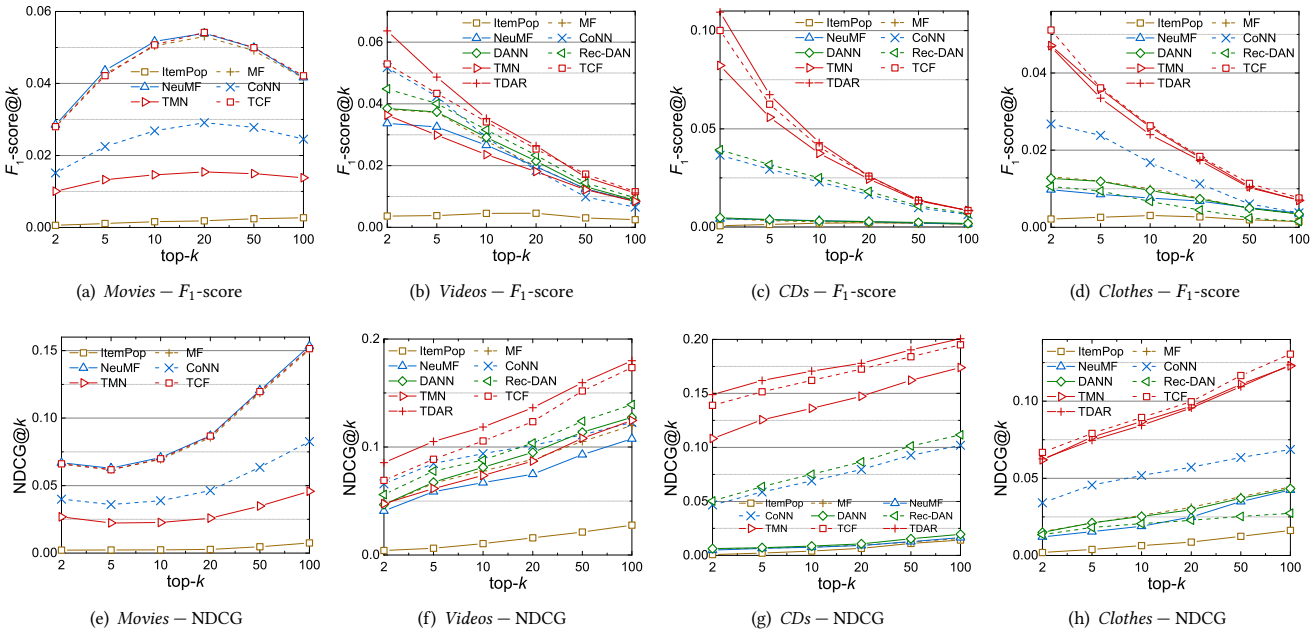
- **ItemPop:** This method ranks items based on **Item Pop**ularity. It is a non-personalized method to benchmark the recommendation performance [8, 21].
- **MF:** This **M**atrix **F**actorization [13] method is the basic yet competitive model. MF uncovers the underlying latent factors that encode the user preferences and item properties to predict missing values.
- **NeuMF:** This **Neu**ral **M**atrix **F**actorization method is the state-of-the-art neural network method for recommendation [8]. NeuMF learns non-linear combination of user and item embeddings by wide and deep structures.
- **CoNN:** This Deep **Co**operative **N**eural **N**etworks is the state-of-the-art text-based recommendation method [30]. CoNN leveraged CNN to extract user and item textual features from reviews to predict the user preferences.
- **DANN:** This **D**omain **A**dversarial **N**eural **N**etwork (**DANN**) [5] is proposed to align high-level representations in various machine learning tasks. We utilize it to align embeddings directly in the recommendation context. We use MF as the basic models, which are supervised on two domains.
- **Rec-DAN:** This **D**iscriminative **A**dversarial **N**etworks for **Rec**ommender Systems is the state-of-the-art cross-domain recommendation model [23]. Rec-DAN used DANN to align textual features to transfer the knowledge. The target part of Rec-DAN is unsupervised.

Since several baselines are designed for explicit feedbacks (MF, CoNN, Rec-DAN), we use cross entropy to optimize them in our experiments. Note that though we introduced many cross-domain recommendation models in Related Work, most of them are not comparable. Only [14, 23] are designed for the no overlap case. The codebook approach [14] is widely used yet cannot be extend to implicit feedbacks, we leave out the comparison with it.

5.1.4 Evaluation Protocols. To evaluate the performance of our proposed model and baselines in implicit feedback case, we rank all items for each user in validation/test set and recommend the top-k items to the user. We then adopt two metrics, F_1-score and normalized discounted cumulative gain (NDCG) to evaluate the recommendation quality. F_1-score, which is defined as harmonic mean of precision and recall, is extensively used to test the accuracy of binary classifier. NDCG is a position-sensitive metric widely used to measure the ranking quality. We recommend top-k items to each user and calculate metrics, and finally use the average metrics of all users to remark the performance of the models.

5.1.5 Parameter Setting. To compare fairly, all models in our experiments are tuned with the following strategies: The maximum iteration number is set to 200. In each iteration, we enumerate all positive samples and test the model. The learning rate η and the regularization coefficient λ are determined by grid search in the coarse grain range of $\{0.0001, 0.001, 0.01, 0.1\} \otimes \{0.001, 0.01, 0.1, 1\}$ and then in the fine grain range, which is based on the result of coarse tuning. For example, if a certain model achieves the best performance when $\eta = 0.01$ and $\lambda = 0.1$, we then tune it in the range of $\{0.002, 0.005, 0.01, 0.02, 0.05\} \otimes \{0.02, 0.05, 0.1, 0.2, 0.5\}$. Considering some baselines (CoNN and Rec-DAN) use dropout for generalization, we tune these models with respect to η and the dropout rate also by the coarse and fine grid search. We determine the batch size in range of $\{64, 128, \cdots, 1024\}$ and the embedding length in the range of $\{8, 16, \cdots, 128\}$. The experiments are repeated 5 times for each parameter setting in model tuning and 10 times for the final comparison. Our experiments are conducted by predicting Top-2, 5, 10, 20, 50, 100 favourite items and we tune models according to F_1-score@2.

The aforementioned part is the tuning strategy for single-domain models, now we take TDAR as example to introduce the tuning strategy of cross-domain models. We train and tune TCF on the source domain, and initialize source domain embeddings of TDAR ($\mathbf{U}^s$ and $\mathbf{V}^s$) by pretrained TCF. Also, hyperparameters (η^s in Equations (5) and (6) and λ^s in Equation (4)) are set as the best hyperparameter setting of pretrained TCF. For learning rates η^+, we set η^s, η^t, and η^- as 0 in TDAR to train a embedding classification task to determine the best η^+. To determine η^-, we set η^+ as the best η^+ we got in last step and set η^s and η^t as 0 to train a embedding aligning task. η^+ and η^- are both tuned in range of $\{1, 0.1, \cdots, 0.00001\}$ according



(a) *Movies* — F_1-score (b) *Videos* — F_1-score (c) *CDs* — F_1-score (d) *Clothes* — F_1-score

(e) *Movies* — NDCG (f) *Videos* — NDCG (g) *CDs* — NDCG (h) *Clothes* — NDCG

Figure 3: Overall performance comparison (test set)

to the classification accuracy of $g(\ )$. For remained hyperparameters η^t and λ^t, we tune TDAR with respect to them by the tuning strategy designed for single-domain models.

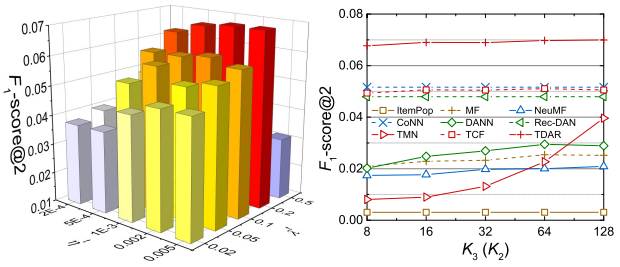
5.2 Performance of TDAR (RQ1)

We compare our models, TMN, TCF, and TDAR against several baselines and show the performance in Figure 3. Lines in red, green, blue, and brown indicate our proposed models, cross-domain baselines, deep baselines, and shallow baselines, respectively. ItemPop is a non-personalized baseline to benchmark the recommendation performance on a dataset, and from its performance we can see that these three target datasets are very challenging due to the sparsity. Compared with ItemPop, all learning models capture users' personal preferences and outperform it significantly — 8.23 times higher on F_1-score for the best case. MF is the basic learning baseline and NeuMF is the corresponding deep version, in which FC layers are leveraged to learn the interaction of user and item latent factors. NeuMF is a competitive model widely used in various recommendation scenarios and usually outperforms MF significantly [8, 24], however in our experiments, NeuMF fails to outperform MF. The possible reason is that the datasets we adopt are extremely sparse and NeuMF faces a more serious overfitting problem due to its stronger representation ability. Considering that our model is mainly designed for sparse datasets, we use inner product as our interaction function.

CoNN is also a deep model for recommendation, which extracts textual features from user reviews. In our experiments, this text-based model outperforms latent factor models (MF and NeuMF) in most cases, thus we come to the conclusion that without sufficient supervision, side information boosts the performance significantly, because that latent factor models only get information from interactions while CoNN additionally gets information from user reviews. Comparing Table 1 and Figure 3, we can see that on sparse datasets (especially on *CDs*), interactions cannot provide enough information, thus the performance of latent factor models is extremely dreadful. Supported by textual information, CoNN shows more robust performance.

DANN and Rec-DAN are two cross-domain recommendation models by domain adaptation. DANN aligns latent factors and Rec-DAN aligns textual feature to transfer knowledge. As we discussed, latent factor models is uncompetitive compared with text-based models on sparse datasets, thus Rec-DAN performs much better than DANN — 7.45 times improvement on F_1-score for the best case. Enhanced by both the novel textual feature extractor and the text-based domain adaptation, TDAR gains significant improvement. TDAR outperforms the best baseline 178.47% for the best case and 11.53% for the worse case on F_1-score, and 195.87% for the best case and 23.95% for the worse case on NDCG (without special note, we use relative improvement by default). Since our accuracy boost is caused by two parts, we discuss the effectiveness of them separately in Subsections 5.3 and 5.4.

We then report some tuning details in Figure 4. To save space, we only report the result on *Videos* due to the similar performance. Figure 4(a) shows the impact of learning rate η^t and regularization coefficient λ^t. TDAR achieves the best performance when $\eta^t = 0.005$ and $\lambda^t = 0.2$. Figure 4(b) illustrates the sensitivity analysis of latent factor number (denoted as K_3 for latent factor models and K_2 for TMN). In TMN, we use semantic features pretrained on GoogleNews corpus by word2vec [17], and K_1 is fixed to 300, thus we tune TMN only with respect to K_2. Note that ItemPop, CoNN, and Rec-DAN are not latent factor models, their performance keeps



(a) Impact of η^t and λ^t (b) Impact of K_3 (K_2)

Figure 4: Model tuning (*Videos*, validation set)

constant. Larger K_3 (K_2) means stronger representation ability, however, when the sparse dataset cannot provide enough information, strong representation ability helps a little, therefore the performance of MF, NeuMF, and DANN increases very slightly with the increasing of K_3. TCF and TDAR is stronger in representation ability – they make prediction with both textual features and latent factors, hence the increasing of K_3 leads to almost no improvement. The supervision of general models is user-item interaction data while of TMN is user-item-word interaction data, which provides much more information, thus the performance of TMN increases significantly with the increasing of K_2.

5.3 Effectiveness of Textual Information (RQ2)

In this subsection, we discuss the effectiveness of our textual features. We proposed a memory network TMN as the feature extractor, and injected the feature into MF to propose TCF. The performance of TMN and TCF is also shown in Figures 3 and 4(b).

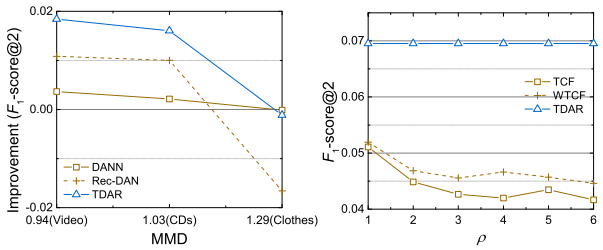
We first compare the text-based models (TMN, TCF, and CoNN) with pure latent factor models (MF and NeuMF). They perform quite differently on the datasets with different sparsity — the enhancement gained from textual information increases significantly with the increasing of sparsity. On the most sparse dataset *CDs*, performance of MF and NeuMF is extremely bad and CoNN outperforms MF 8.16 times on F_1-score for the best case. Compared with *CDs*, *Videos* is much denser, where the performance of text-based models and latent factor models is very similar. Filtered by 30 core, *Movies* is the densest dataset. On *Movies*, latent factor models perform much better than models with only text features (CoNN and TMN).

We then compare our proposed textual features (TMN) with the existing one (CoNN). Compared with CoNN, TMN is good at sparse cases. On *CDs* and *Clothes* datasets, TMN performs pretty well — TMN outperforms CoNN 1.25 times on F_1-score for the best case. However, on *Movies* and *Videos*, CoNN performs much better — CoNN outperforms TMN 88.84% on F_1-score for the best case. TMN and CoNN both have specific superiorities: TMN is good at highlighting important keywords while CoNN can model sequential information. As we discussed in Section 3, sequential information is not important in our task. This viewpoint is supported by the experiment — TMN outperforms CoNN in sparse cases.

We finally study the effectiveness of combining textual features with latent factors by comparing TMN and TCF. From Figure 3 we can observe that we gain more improvement on sparse datasets: TCF outperforms TMN significantly on *CDs*, *Clothes*, and *Videos* while they perform pretty similarly on *Movies*. We consider that on sparse datasets, textual features and latent factors complement each other in encoding user preferences, while on dense datasets, implicit feedback data provides enough information thus extra data helps little.

5.4 Effectiveness of Our Domain Adaptation Strategy (RQ3)

In this subsection, we discuss the effectiveness of our text-based domain adaptation strategy by comparing TDAR against DANN and Rec-DAN. We have compared them according to the accuracy in Subsection 5.2. Considering TDAR outperforms DANN and Rec-DAN significantly mainly due to the outstanding effectiveness of our textual features, we use another measurement: improvement over the basic models to evaluate the effectiveness of each domain adaptation approaches. We measure the absolute improvement, i.e., the difference between performances of a cross-domain model and of its basic model. The basic model of DANN is MF, and learning on the target domain is supervised by both positive and negative samples. The basic model of Rec-DAN is LSTM and learning on the target domain is unsupervised.



(a) Impact of domain similarity (b) Sampling quality (*Videos*)

Figure 5: Analysis for domain adaptation (validation set)

The improvement of DANN, Rec-DAN, and TDAR on three target datasets is shown in Figure 5(a) (here we report the result on validation sets, since improvement on test set can be found in Figure 3). As shown in Figures 3 and 4(b), TDAR outperforms cross-domain baselines on predicting accuracy. From Figures 3 and 5(a) we can see that TDAR also performs well on accuracy improvement. Figures 5(a) shows that TDAR outperforms the best cross-domain baseline 70.06% on improvement for the best case. Comparing Figures 3 and 5(a) we can see that another advantage of TDAR is that we can gain stable improvement on test set. For example, on *Videos*, both DANN and TDAR outperform their basic models on the validation set (DANN outperforms MF 9.29% and TDAR outperforms TCF 35.98% on F_1-score@2), while on the test set, only TDAR still achieves significant improvement (DANN outperforms MF 0.73% and TDAR outperforms TCF 20.21% on F_1-score@2). This is an experimental evidence for our key idea shown in Figure 1: it is not helpful to align embeddings directly in recommendation context. It is also our motivation to propose our text-directed domain adaptation.

As mentioned in Subsection 5.1.1, we select three target datasets by considering the different similarity to the source dataset. To

quantify the similarity, we calculate Maximum Mean Discrepancy (MMD) [15] of text features of source and target datasets. Figure 5(a) also shows the impact of domain similarity, and MMD in the abscissa is the distance of two distributions. As shown in the figure, the more similar the two datasets are, the more improvement we gain by transfer learning. Since *Movies* and *Clothes* are very different and there is little useful knowledge to transfer, Rec-DAN performs dreadfully in this case since it is unsupervised on the target domain.

As we discussed in Subsection 4.2, we use domain adaptation instead of negative sampling to supervise the model on target domain, therefore our TDAR can be regarded as an approach to improve sampling quality. To evaluate the effectiveness of our sampling strategy, we compare TDAR with two baselines:

- **TCF:** TCF (Subsection 4.1) is the basic model of TDAR optimized by cross entropy (Equation (1)). To improve the sampling quality, we randomly select ρ negative samples for each positive one.
- **WTCF:** We introduce this **W**eighted **TCF** method inspired by Weighted Matrix Factorization (WMF) [6]. When optimized by cross entropy, we weight each positive sample with 1 and weight each negative sample i with $p_i/\bar{p}$, where $\bar{p}$ is the average popularity and p_i is the popularity of i. Popular unobserved items are considered less likely to be neglected hence with more confidence to be true negative samples.

Experiment result is illustrated in Figure 5(b). In sparse data, conventional sampling strategies take little effect. There is much label noise in negative labels and more negative samples means more noise, thus the performance of TCF and WTCF decreases when ρ increases. The improvement of the weighted sampling strategy is very marginal. Compared with conventional sampling strategies, TDAR provides a novel way to avoid negative sampling: We learn a basic model on source domain with label level supervision, and learn another basic model on target domain with both partial label level supervision and embedding level supervision. Comparing the performance of MF on *Movies* and *CDs*, we can see that the performance on target domains is seriously damaged since there is too much label noise. In contrast, label noise on the source domain is totally acceptable. As a result, though TDAR is still affected by noise on source domain, we gain considerable improvement by avoiding noise on the target domain.

6 CONCLUSION

In real-world applications, we usually face extremely sparse datasets, where conventional sampling strategies cause much label noise. In this paper, we aim to learn knowledge from dense datasets and transfer it to sparse ones. To do so, we focus on cross-domain recommendation without overlap by aligning embeddings. As the embeddings of different datasets encode different information, aligning them directly is meaningless. To address this gap, we use textual features to direct domain adaptation. We first map users and items of all domains to the same semantic space to construct the textual representation, and then align embeddings with these domain-invariant features. Comprehensive experiments show that our text-enhanced domain adaptation strategy for recommendation outperforms existing approaches significantly. For the future work, we are interested in exploring other side information such as images or tags to direct domain adaptation. We also want to validate the effectiveness of our transfer learning approach in other graph embedding tasks such as social networks.

REFERENCES

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. 2018. Partial Transfer Learning With Selective Adversarial Networks. In *CVPR*. 2724–2732.
[2] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW*. 1583–1592.
[3] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized Key Frame Recommendation. In *SIGIR*. 315–324.
[4] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *ICML*. 1180–1189.
[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *The Journal of Machine Learning Research* (2016), 2096–2030.
[6] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Bayesian personalized ranking for non-uniformly sampled items. In *KDDCup*.
[7] David Goldberg. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the Acm* (1992), 61–70.
[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[9] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *CIKM*. 667–676.
[10] Guangneng Hu, Yu Zhang, and Qiang Yang. 2019. Transfer Meets Hybrid: A Synthetic Approach for Cross-Domain Collaborative Filtering with Text. In *WWW*. 2822–2829.
[11] Heishiro Kanagawa, Hayato Kobayashi, Nobuyuki Shimizu, Yukihiro Tagami, and Taiji Suzuki. 2018. Cross-domain Recommendation via Deep Domain Adaptation. *CoRR* (2018).
[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[13] Yehuda Koren. 2009. The bellkor solution to the netflix grand prize. *Netflix prize documentation* (2009), 1–10.
[14] Bin Li, Qiang Yang, and Xiangyang Xue. 2009. Can Movies and Books Collaborate?: Cross-domain Collaborative Filtering for Sparsity Reduction. In *IJCAI*. 2052–2057.
[15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In *ICML*. 97–105.
[16] Zhongqi Lu, Weike Pan, Evan Wei Xiang, Qiang Yang, Lili Zhao, and Erheng Zhong. 2013. Selective Transfer Learning for Cross Domain Recommendation. In *SDM*. 641–649.
[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*. 3111–3119.
[18] Weike Pan and Li Chen. 2013. GBPR: group preference based Bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*. 2691–2697.
[19] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. 2010. Transfer Learning in Collaborative Filtering for Sparsity Reduction. In *AAAI*.
[20] S. Rendle. 2010. Factorization Machines. In *ICDM*. 995–1000.
[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *WWW*. 285–295.
[23] Cheng Wang, Mathias Niepert, and Hui Li. 2019. RecSys-DAN: Discriminative Adversarial Networks for Cross-Domain Recommender Systems. *CoRR* (2019).
[24] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.
[25] Wenhui Yu and Zheng Qin. 2019. Spectrum-enhanced Pairwise Learning to Rank. In *WWW*. 2247–2257.
[26] Wenhui Yu and Zheng Qin. 2020. Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters. In *ICML*.
[27] Wenhui Yu and Zheng Qin. 2020. Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning. In *SIGIR*.
[28] Wenhui Yu, Huidi Zhang, Xiangnan He, Xu Chen, Li Xiong, and Zheng Qin. 2018. Aesthetic-based Clothing Recommendation. In *WWW*. 649–658.
[29] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. DARec: Deep Domain Adaptation for Cross-Domain Recommendation via Transferring Rating Patterns. In *IJCAI*. 4227–4233.
[30] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *WSDM*. 425–434.