Transfer Meets Hybrid: A Synthetic Approach for Cross-Domain Collaborative Filtering with Text

Guangneng Hu
Department of Computer Science and Engineering, Hong Kong University of Science and Technology
njuhgn@gmail.com

Yu Zhang
Department of Computer Science and Engineering, Hong Kong University of Science and Technology
yu.zhang.ust@gmail.com

Qiang Yang
Department of Computer Science and Engineering, Hong Kong University of Science and Technology
qyang@cse.ust.hk

ABSTRACT

Collaborative Filtering (CF) is the key technique for recommender systems. CF exploits user-item behavior interactions (e.g., clicks) only and hence suffers from the data sparsity issue. One research thread is to integrate auxiliary information such as product reviews and news titles, leading to hybrid filtering methods. Another thread is to transfer knowledge from source domains such as improving the movie recommendation with the knowledge from the book domain, leading to transfer learning methods. In real-world applications, a user registers for multiple services across websites. Thus it motivates us to exploit both auxiliary and source information for recommendation in this paper. To achieve this, we propose a Transfer Meeting Hybrid (TMH) model for cross-domain recommendation with unstructured text. The proposed TMH model attentively extracts useful content from unstructured text via a memory network and selectively transfers knowledge from a source domain via a transfer network. On two real-world datasets, TMH shows better performance in terms of three ranking metrics by comparing with various baselines. We conduct thorough analyses to understand how the text content and transferred knowledge help the proposed model.

CCS CONCEPTS

• **Information systems** **Personalization**.

KEYWORDS

Recommender Systems; Collaborative Filtering; Deep Learning

ACM Reference Format:
Guangneng Hu, Yu Zhang, and Qiang Yang. 2019. Transfer Meets Hybrid: A Synthetic Approach for Cross-Domain Collaborative Filtering with Text. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3308558.3313543

1 INTRODUCTION

Recommender systems are widely used in various domains and e-commerce platforms, such as recommending products to buy at Amazon and videos to watch on Youtube. Collaborative Filtering (CF) is an effective approach based on an intuition that if users rated items similarly in the past then they are likely to rate items similarly in the future. Matrix Factorization (MF) techniques are its main cornerstone [26, 35] since they can learn latent factors for users and items. Recently, neural networks like multilayer perceptrons (MLP) are used to learn non-linear interaction functions from data [10, 17]. Both MF and neural CF suffer from the data sparsity and cold-start issues.

One solution is to integrate CF with the content information, leading to hybrid methods. Items are usually associated with unstructured text like the news articles and product reviews. Additional information alleviates the data sparsity issue and is essential for recommendation beyond user-item interactions. For application domains like recommending research papers and news articles, the unstructured text associated with an item is its text content [1, 47]. Other domains like recommending products, the unstructured text associated with the item is its user reviews which justify the rating behavior of consumers [21, 33, 56]. Recently, neural networks have been proposed to exploit the item content. For example, memory networks [43] are used to model item reviews [19], or to model a user's neighbors who rated the same items with this user [11].

Another solution is to transfer the knowledge from relevant domains and the cross-domain recommendation techniques address such problems [3, 27, 37]. In real-world applications, a user typically registers multiple service systems to acquire different information need. For example, a user installs applications in an app store and reads news from another website. It brings us an opportunity to improve the recommendation performance in the target service (or all services) by learning from across domains. In the above example, we can represent the app installation feedback using a binary matrix whose entries indicate whether a user has installed an app. Similarly, we use another binary matrix to indicate whether a user has read a news article. Typically these two matrices are highly sparse, and it is beneficial to learn them simultaneously. This idea is sharpened into the Collective Matrix Factorization (CMF) approach [42] which jointly factorizes these two matrices by sharing the user latent factors. It combines CF on a target domain and another CF on an auxiliary domain, enabling knowledge transfer [36, 54]. In terms of neural networks, given two activation maps from two tasks, cross-stitch network [34] and its sparse variant [22] learn linear combinations of both the input activations and feed these combinations as input to the successive layers, and hence enabling the knowledge transfer between two domains.

These two threads motivate us to exploit information from both the content and cross-domain information for recommendation in this paper. To capture text content and to transfer cross-domain

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313543

knowledge, we propose a novel neural model, TMH, for cross-domain recommendation with unstructured text. TMH can not only attentively extract useful content via a memory network but also selectively transfer knowledge across domains by a novel transfer network. A shared layer of feature interactions is stacked on the top to couple high-level representations learned from both networks. On real-world datasets, TMH shows the better performance in various settings. We conduct thorough analyses to understand how the content and transferred knowledge help the proposed model.

Our contributions are summarized as follows:

- The proposed model is a novel deep model that transfers cross-domain knowledge for recommendation with unstructured text by using an attention based neural network. (Sec. 4)
- We interpret the memory networks to attentively exploit the text content to match word semantics with user preferences.
- The transfer network can selectively transfer source items with the guidance of target user-item interactions by the attentive weights.
- Our model alleviates cold-user and cold-item start issues, and outperforms various baselines on real-world datasets. (Sec. 5)

2 RELATED WORKS

We review related works on three topics: collaborative filtering, hybrid methods, and cross-domain recommendation.

Collaborative Filtering Recommender systems aim at learning user preferences on unknown items from their past history. Content-based recommendations are based on the matching between user profiles and item descriptions. It is difficult to build the profile for each user when there is no/few content. CF alleviates this issue by predicting user preferences based on the user-item interaction behavior, agnostic to the content [9]. Latent factor models learn feature vectors for users and items mainly based on MF [26] which has probabilistic interpretations [35]. Factorization machines (FM) can mimic MF [40]. To address the data sparsity, an item-item matrix called Shifted Positive Pointiest Mutual Information (SPPMI) is constructed from the user-item interaction matrix in the CoFactor model [28]. It then simultaneously factorizes the interaction matrix and the SPPMI matrix in a shared item latent space, enabling the usage of co-click information to regularize the learning of the user-item matrix. In contrast, we use independent unstructured text and source domain information to alleviate the data sparsity issue in the user-item matrix. Neural networks are proposed to push the learning of feature vectors towards non-linear representations, including the Neural Network Matrix Factorization (NNMF) and MultiLayer Perceptron (MLP) [10, 17]. The basic MLP architecture is extended to regularize the factors of users and items via social and geographical information [51]. Other neural approaches learn from the explicit feedback for the rating prediction task [5, 56]. We focus on learning from the implicit feedback for top-N recommendation [50].

Hybrid Filtering Items are usually associated with the content information such as unstructured text (e.g., abstracts of articles and reviews of products). CF approaches can be extended to exploit the content information [1, 47, 48] and user reviews [16, 20, 33]. Combining matrix factorization and topic modelling technique (e.g., Topic MF) is an effective way to integrate ratings with item contents [2, 29, 33]. Item reviews justify the rating behavior of a user, and item ratings are associated with their attributes hidden in reviews [14]. Topic MF methods combine latent item factors in ratings with latent topics in reviews [2, 33]. The behavior factors and topic factors are aligned with a link function such as softmax transformation in the Hidden Factors and hidden Topics (HFT) model [33] or an offset deviation in the Collaborative Topic Regression (CTR) model [47]. The CTR model assumes the latent item vector learnt from the interaction data is close to the corresponding topic proportions learnt from the text content, but allows them to be divergent from each other if necessary. Additional sources of information are integrated into CF to alleviate the data sparsity issues including knowledge graph [49, 55]. Convolutional Networks (CNNs) have been used to extract features from audio signals for music recommendation [45] and from image for product and multimedia recommendation [6, 16]. Autoencoders are used to learn an intermediate representations from text [48, 53]. Recurrent networks [1] and convolutional networks [5, 24, 56] can exploit the word order when learning the text representations. Memory networks can reason with an external memory [43]. Due to the capability of neurally learnt word embeddings to address the problems of word sparseness and semantic gap, a memory module can be used to model item content [19] or the neighborhood of users [11]. Memory networks can learn to match word semantics with the specific user. We follow this thread by using neural networks to attentively extract important information from text content.

Cross-domain Recommendation Cross-domain recommendation [3] is an effective technique to alleviate the data sparsity issue. A class of MF-based methods has been applied to cross-domain recommendation. Typical methods include the CMF approach which jointly factorizes two rating matrices by sharing the latent user factors and hence it enables knowledge transfer. CMF has its heterogeneous variants [37], and codebook transfer [27]. The coordinate system transfer can exploit heterogeneous feedbacks [38, 52]. Multiple source domains [32] and multi-view learning [13] are also proposed for integrating information from several domains. Transfer Learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [36]. Similar to TL, Multitask Learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [4, 54]. The cross-stitch network [34] and its sparse variant [22] enable information sharing between two base networks for each domain in a deep way. Robust learning is also considered during knowledge transfer [15]. These methods treat knowledge transfer as a global process with shared global parameters and do not match source items with the specific target item given a user. We follow this research thread by using neural networks to selectively transfer knowledge from the source items. We introduce a transfer network to exploit the source domain knowledge.

3 A BASIC NEURAL CF NETWORK

We adopt a Feedforward Neural Network (FFNN) as the base neural CF model to parameterize the interaction function [7, 8, 17]:

$$f(\boldsymbol{x}_{ui}|\boldsymbol{P}, \boldsymbol{Q}, \theta_f) = \phi_o(...(\phi_1(\boldsymbol{x}_{ui}))...), \tag{1}$$

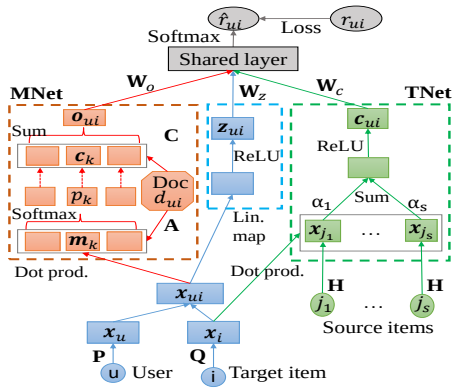

Figure 1: The architecture of TMH.

where input $\boldsymbol{x}_{ui} = [\boldsymbol{P}^T\boldsymbol{x}_u, \boldsymbol{Q}^T\boldsymbol{x}_i] \in \mathbb{R}^{2d}$ is concatenated from embeddings of user u and item i and is a projection of their one-hot encodings $\boldsymbol{x}_u \in \{0,1\}^m$ and $\boldsymbol{x}_i \in \{0,1\}^n$ via embedding matrices $\boldsymbol{P} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{Q} \in \mathbb{R}^{n\times d}$, respectively. The output and hidden layers are computed by ϕ_o and $\{\phi_l\}$ in FFNN. The sizes of users and items are m and n, respectively. The rating r_{ui} is 1 if user u has an interaction with item i and 0 otherwise.

The base network consists of four modules with the information flow from the input (u, i) to the output $\hat{r}_{ui}$ as follows.

Input: $(u, i) \rightarrow \mathbb{1}_u, \mathbb{1}_i$ This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i, and maps them into one-hot encodings $\mathbb{1}_u \in \{0,1\}^m$ and $\mathbb{1}_i \in \{0,1\}^n$ where only the element corresponding to that index is 1 and all others are 0.

Embedding: $\mathbb{1}_u, \mathbb{1}_i \rightarrow \boldsymbol{x}_{ui}$ This module firstly embeds one-hot encodings into continuous representations $\boldsymbol{x}_u = \boldsymbol{P}^T\mathbb{1}_u$ and $\boldsymbol{x}_i = \boldsymbol{Q}^T\mathbb{1}_i$ by embedding matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ respectively, and then concatenates them as $\boldsymbol{x}_{ui} = [\boldsymbol{x}_u, \boldsymbol{x}_i]$, to be the input of following building blocks.

Hidden layers: $\boldsymbol{x}_{ui} \rightsquigarrow \boldsymbol{z}_{ui}$. This module takes the continuous representations from the embedding module and then transforms through several layers to a final latent representation $\boldsymbol{z}_{ui} = (\ldots(\phi_1(\boldsymbol{x}_{ui})\ldots)$. This module consists of hidden layers to learn nonlinear interaction between users and items.

Output : $\boldsymbol{z}_{ui} \rightarrow \hat{r}_{ui}$. This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the representation $\boldsymbol{z}_{ui}$ from the last layer of multi-hop module. Since we focus on one-class collaborative filtering, the output is the probability that the input pair is a positive interaction. This can be achieved by a softmax layer: $\hat{r}_{ui} = \phi_o(z_{ui}) = \frac{1}{1+\exp(-\boldsymbol{h}^T\boldsymbol{z}_{ui})}$, where $\boldsymbol{h}$ is the parameter.

4 THE PROPOSED TMH MODEL

The architecture of the TMH model is illustrated in Fig. 1.

Matching Word Semantics with User Preferences We adapt a memory network (MNet) to integrate unstructured text since it can learn to match word semantics with user preferences [12, 19, 23, 44]. The MNet consists of one internal memory matrix $\mathbf{A} \in \mathbb{R}^{L\times 2d}$, where L is the vocabulary size (typically $L = 8,000$ after processing [47]) and $2d$ is the dimension of each memory slot, and one external memory matrix $\mathbf{C}$ with the same dimensions as $\mathbf{A}$. The function of the two memory matrices works as follows.

Given a document $d_{ui} = (w_1, w_2, \ldots, w_l)$ corresponding to the (u, i) interaction, we form the memory slots $\boldsymbol{m}_k \in \mathbb{R}^{2d}$ by mapping each word w_k into an embedding vector with matrix $\mathbf{A}$, where $k = 1, \ldots, l$ and the length of the longest document is equal to the memory size. We form a preference vector $\boldsymbol{q}^{(ui)}$ corresponding to the given document d_{ui} and the user-item interaction (u, i) where each element encodes the relevance of user u to these words given item i as: $q_k^{(ui)} = \boldsymbol{x}_u^T\boldsymbol{m}_k^{(u)} + \boldsymbol{x}_i^T\boldsymbol{m}_k^{(i)},\ k = 1, \ldots, l$, where we split the $\boldsymbol{m}_k = [\boldsymbol{m}_k^{(u)}, \boldsymbol{m}_k^{(i)}]$ into the user part $\boldsymbol{m}_k^{(u)}$ and the item part $\boldsymbol{m}_k^{(i)}$.

Then, we compute the attentive weights over words for a given user-item interaction to infer the importance of each word's unique contribution: $p_k^{(ui)} = \text{Softmax}(q_k^{(ui)}) = \frac{\exp(\beta q_k^{(ui)})}{\sum_{k'}\exp(\beta q_{k'}^{(ui)})}$, where parameter β is introduced to stabilize the numerical computation and can amplify or attenuate the precision of the attention like a temperature [18]. We set $\beta = d^{-\frac{1}{2}}$ by scaling along with the dimensionality [46].

We construct the high-level representations by interpolating the external memories with the attentive weights as the output:

$$\boldsymbol{o}_{ui} = \sum\nolimits_k p_k^{(ui)}\boldsymbol{c}_k, \tag{2}$$

where the external memory slot $\boldsymbol{c}_k \in \mathbb{R}^d$ is another embedding vector for word w_k by mapping it with matrix $\mathbf{C}$.

Selecting Source Items to Transfer We propose a novel transfer network (TNet) which can selectively transfer source knowledge for specific target item in a coarse-to-fine way. Given the source items $[j]^u = (j_1, j_2, \ldots, j_s)$ with which the user u has interacted in the source domain, TNet learns a transfer vector $\boldsymbol{c}_{ui} \in \mathbb{R}^d$ to capture the relations between the target item i and source items given the user u. The similarities between target item i and source items can be computed by their dot products: $a_j^{(i)} = \boldsymbol{x}_i^T\boldsymbol{x}_j,\ j = 1, \ldots, s$, where $\boldsymbol{x}_j \in \mathbb{R}^d$ is the embedding for the source item j by an embedding matrix $\boldsymbol{H} \in \mathbb{R}^{n_S\times d}$. This score computes the compatibility between the target item and the source items consumed by the user.

Then, we normalize similarity scores to be a probability distribution over source items: $\alpha_j^{(i)} = \text{Softmax}(a_j^{(i)})$. Finally the transfer vector is a weighted sum of the corresponding source item embeddings:

$$\boldsymbol{c}_{ui} = \text{ReLU}(\sum\nolimits_j \alpha_j^{(i)}\boldsymbol{x}_j), \tag{3}$$

where we introduce non-linearity on the transfer vector by the rectified linear unit $\text{ReLU}(x) = max(0, x)$.

Putting It All Together We firstly use a simple neural CF model (CFNet) which has one hidden layer to learn a nonlinear representation for the user-item interaction:

$$\boldsymbol{z}_{ui} = \text{ReLU}(\boldsymbol{W}\boldsymbol{x}_{ui} + \boldsymbol{b}), \tag{4}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight and bias parameters in the hidden layer. Usually the dimension of $\boldsymbol{z}_{ui}$ is half of that $\boldsymbol{x}_{ui}$ in a typical tower-pattern architecture.

The outputs from the three individual networks can be viewed high-level features of the content text, source domain knowledge, and the user-item interaction. They come from different feature space learned by different networks. Thus, we use a shared layer on

the top of the all features: $\hat{r}_{ui} = \frac{1}{1+\exp(-\boldsymbol{h}^T \boldsymbol{y}_{ui})}$, where $\boldsymbol{h}$ is the parameter. The joint representation, $\boldsymbol{y}_{ui} = [\boldsymbol{W}_o \boldsymbol{o}_{ui}, \boldsymbol{W}_z \boldsymbol{z}_{ui}, \boldsymbol{W}_c \boldsymbol{c}_{ui}]$, is concatenated from the linear mapped outputs of individual networks where matrices $\boldsymbol{W}_o, \boldsymbol{W}_z, \boldsymbol{W}_c$ are the corresponding linear mapping transformations.

Learning Due to the nature of the implicit feedback and the task of item recommendation, the squared loss $(\hat{r}_{ui} - r_{ui})^2$ may be not suitable since it is usually for rating prediction. Instead, we adopt the binary cross-entropy loss: $\mathcal{L} = -\sum_{(u,i)\in\mathcal{S}} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui})$, where the training samples $\mathcal{S} = \boldsymbol{R}_T^+ \cup \boldsymbol{R}_T^-$ are the union of observed target interaction matrix and randomly sampled negative pairs. Usually, $|\boldsymbol{R}_T^+| = |\boldsymbol{R}_T^-|$ and we do not perform a predefined negative sampling in advance since this can only generate a fixed training set of negative samples. Instead, we generate negative samples during each epoch, enabling diverse and augmented training sets of negative examples to be used. The objective function can be optimized by stochastic gradient descent (SGD) and its variants like adaptive moment(Adam) method [25].

Complexity In the model parameters Θ, the embedding matrices $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{H}$ contain a large number of parameters since they depend on the input size of users and (target and source) items, and their scale is hundreds of thousands. Typically, the number of words, i.e., the vocabulary size is $L = 8,000$ [47]. The dimension of embeddings is typically $d = 100$. Since the architecture follows a tower pattern, the dimension of the outputs of the three individual networks is also limited within hundreds. In total, the size of model parameters is linear with the input size and is close to the size of typical latent factors models [42] and neural CF approaches [17] with a hidden layer. During training, we compute the outputs of the three individual networks in parallel using mini-batch stochastic optimization which can be trained efficiently by back-propagation. TMH is scalable to the number of the training data. It can easily update when new data examples come by just feeding them into the training mini-batch. Thus, TMH can handle the scalability and dynamics of items and users similar to an online fashion. In contrast, topic modeling techniques have difficulty in benefitting from these advantages to some extent.

5 EXPERIMENTS

In this section, we conduct empirical study to answer the following questions: 1) how the proposed TMH model performs compared with state-of-the-art recommender systems; and 2) how the text content and the source domain information contributes to the proposed framework. We firstly introduce the evaluation protocols and experimental settings, and then compare the performance of different recommender systems. We further analyze the TMH model to understand the impact of the memory and transfer components. We also investigate that the improved performance comes from the cold-users and cold-items to some extent.

5.1 Experimental Settings

Dataset We evaluate on two real-world cross-domain datasets. The first dataset, **Mobile**, is provided by a large internet company, i.e., Cheetah Mobile (http://www.cmcm.com/en-us/) [30]. The information contains logs of user reading news, the history of app installation, and some metadata such as news publisher and user gender collected in one month in the US. We removed users with fewer than 10 feedbacks. For each item, we use the news title as its text content. Following [47], we filter stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary. This yields a corpus of 612K words. The average number of words per news is less than 10. The dataset we used contains 477K user-news reading records and 817K user-app installations. There are 15.8K shared users which enable the knowledge transfer between the two domains. We aim to improve the news recommendation by transferring knowledge from app domain. The data sparsity is over 99.6%.

Table 1: Datasets and statistics.

Dataset	Domain	Statistics	Amount
Mobile News	Shared	#Users	15,890
	Target	#News	84,802
		#Reads	477,685
		Density	0.035%
		#Words	612,839
		Avg. Words Per News	7.2
	Source	#Apps	14,340
		#Installations	817,120
		Density	0.359%
Amazon Product	Shared	#Users	8,514
	Target	#Clothes (Men)	28,262
		#Ratings/#Reviews	56,050
		Density	0.023%
		#Words	1,845,387
		Avg. Words Per Review	32.9
	Source	#Products (Sports)	41,317
		#Ratings/#Reviews	81,924
		Density	0.023%

The second dataset is a public **Amazon** dataset (http://snap.stanford.edu/data/web-Amazon.html), which has been widely used to evaluate the performance of collaborative filtering approaches [16]. We use two categories including Amazon Men and Amazon Sports as two domains [16, 20]. The original ratings are from 1 to 5 where five stars indicate that a user shows a positive preference on the item while the one star is not. We convert the ratings of 4-5 as positive samples. The dataset we used contains 56K positive ratings on Amazon Men and 81K positive ratings on Amazon Sports. There are 8.5K shared users, 28K Men products, and 41K Sports goods. We aim to improve the recommendation on the Men domain by transferring knowledge from relevant Sports domain. The data sparsity is over 99.7%. We filter stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary [47]. The average number of words per review is 32.9.

The statistics of the two datasets are summarized in Table 1. As we can see, both datasets are very sparse and hence we hope to improve performance by transferring knowledge from the auxiliary domain and exploiting the text content as well. Note that Amazon dataset are long text of product reviews (the number of average words per item is 32), while Cheetah Mobile contains short text of news titles (the number of average words per item is 7).

Table 2: Results on Amazon data.

Method	$topK = 5$			$topK = 10$			$topK = 20$		
	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
BPRMF	.0810	.0583	.0509	.1204	.0710	.0561	.1821	.0864	.0602
CDCF	.1295	.0920	.0797	.2070	.1167	.0897	.3841	.1609	.1015
CMF	.1498	.0950	.0771	.2224	.1182	.0863	.3573	.1521	.0957
HFT	.1077	.0815	.0729	.1360	.0907	.0767	.2782	.1252	.0854
TextBPR	.1517	.1208	.1104	.1777	.1291	.1138	.2268	.1414	.1171
CDCF++	.1314	.0926	.0800	.2102	.1177	.0901	.3822	.1605	.1016
MLP	.2100	.1486	.1283	.2836	.1697	.1371	.3820	.1899	.1426
MLP++	.2263	.1626	.1417	.2992	.1862	.1514	.3810	.2069	.1570
CSN	.2340*	.1680*	.1462*	.3018*	.1898*	.1552*	.3944*	.2091*	.1605*
LCMR	.2024	.1451	.1263	.2836	.1678	.1356	.3951	.1918	.1420
TMH	**.2575**	**.1796**	**.1550**	**.3490**	**.2077**	**.1666**	**.4443**	**.2311**	**.1727**
Our improve	10.04%	6.90%	6.01%	15.63%	9.43%	7.34%	12.65%	10.52%	7.60%

Evaluation Protocol For item recommendation task, the Leave-One-Out (LOO) evaluation is widely used and we follow the protocol in [17]. That is, we reserve one interaction as the test item for each user. We determine hyper-parameters by randomly sampling another interaction per user as the validation/development set. We follow the common strategy which randomly samples 99 (negative) items that are not interacted by the user and then evaluate how well the recommender system can rank the test item against these negative ones. Since we aim at top-K item recommendation, the typical evaluation metrics are hit ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR), where the ranked list is cut off at $topK = \{5, 10, 20\}$. HR intuitively measures whether the reserved test item is present on the top-K list and is defined as: $HR = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\delta(p_u \leq topK)$, where p_u is the hit position for the test item of user u, and $\delta(\cdot)$ is the indicator function. NDCG and MRR also account for the rank of the hit position, respectively, and they are defined as: $NDCG = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{\log 2}{\log(p_u+1)}$, and $MRR = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{p_u}$. A higher value with lower cutoff indicates better performance.

Baselines We compare with various baselines, categorized as shallow/deep, single/cross-domain, and hybrid methods. MLP++: We combine two MLPs by sharing user embeddings. CDCF++: We extend CDCF by augmenting the feature vector with words.

Baselines	Shallow method	Deep method
Single-domain	BPRMF [41]	MLP [17]
Cross-domain	CDCF [31], CMF [42]	MLP++, CSN [34]
Hybrid	HFT [33], TextBPR [16, 20]	LCMR [19]
Cross + Hybrid	CDCF++	TMH (ours)

Implementation For BPRMF, we use LightFM's implementation which is a popular CF library. For CDCF and CDCF++, we adapt the official libFM implementation. For CMF, we use a Python version reference to the original Matlab code. For HFT and TextBPR, we use the code released by their authors. The word embeddings used in the TextBPR are pre-trained by GloVe [39]. For latent factor models, we vary the number of factors from 10 to 100 with step size 10. For MLP, we use the code released by its authors. The MLP++ and CSN are implemented based on MLP. The LCMR model is similar to our MNet model. Our methods are implemented using TensorFlow. Parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01^2)$. The optimizer is Adam with initial learning rate 0.001. The size of mini batch is 128. The ratio of negative sampling is 1. The MLP and MLP++ follows a tower pattern, halving the layer size for each successive higher layer. Specifically, the configuration of hidden layers in the base MLP network is $[64 \rightarrow 32 \rightarrow 16 \rightarrow 8]$ as in the original paper [17]. For CSN, it requires that the number of neurons in each hidden layer is the same and the configuration is $[64]*4$ (i.e., $[64 \rightarrow 64 \rightarrow 64 \rightarrow 64]$). We study several typical configurations $\{16, 32, 64, 80\} * 4$. The embedding dimension is $d = 75$.

5.2 Comparison Results

In this section, we report the recommendation performance of different methods. The comparison results are shown in Table 3 and Table 2 respectively on the Mobile and Amazon datasets where the last row is the relative improvement of TMH vs the best baseline. We have the following observations. Firstly, we can see that the proposed TMH model performs better than all baselines on the two datasets at each setting, including the MLP network, shallow cross-domain models (CMF and CDCF), deep cross-domain models (MLP++ and CSN), and hybrid methods (HFT and TextBPR, LCMR). These results demonstrate the effectiveness of the proposed neural model.

On the Mobile dataset, the differences between TMH and other methods are more pronounced for small numbers of recommended items including top-5 or top-10 where we achieve average 2.25% relative improvements over the best baseline. This is a desirable feature since we often recommend only a small number of top ranked items to consumers to alleviate the information overload issue.

Note that the relative improvement of the proposed model vs. the best baseline is more significant on the Amazon dataset than on the Mobile dataset, obtaining average 9.56% relative improvements over the best CSN baseline, though the Amazon data is sparser than the Mobile data (see Table 1). We show the benefit of combining text content by comparing with CSN. One explanation is that the relatedness of the Men and Sports domains is larger than that

Table 3: Results on Mobile data.

Method	$topK = 5$			$topK = 10$			$topK = 20$		
	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
BPRMF	.4380	.3971	.3606	.4941	.4182	.3694	.5398	.4316	.3730
CDCF	.5066	.3734	.3293	.5325	.4089	.3441	.5452	.4374	.3519
CMF	.4789	.3535	.3119	.5846	.3879	.3263	.6662	.4086	.3320
HFT	.4966	.3617	.3175	.5580	.4093	.3365	.6547	.4379	.3445
TextBPR	.4948	.4298	.3826	.5466	.4499	.3913	.6123	.4682	.3958
CDCF++	.4981	.3693	.3267	.6055	.4041	.3411	.6244	.4335	.3491
MLP	.5380	.4121	.3702	.6176	.4381	.3810	.6793	.4529	.3851
MLP++	.5524	.4284	.3871	.6319	.4535	.3976	.6910	.4691	.4019
CSN	.5551*	.4323*	.3920*	.6327*	.4574*	.4025*	.6908	.4732*	.4068*
LCMR	.5476	.4189	.3762	.6311	.4460	.3874	.6927*	.4619	.3918
TMH	**.5664**	**.4427**	**.4018**	**.6438**	**.4680**	**.4124**	**.6983**	**.4820**	**.4163**
Our improve	2.04%	2.42%	2.51%	1.75%	2.32%	2.47%	0.81%	1.86%	2.34%

between the news and app domains. This will benefit all cross-domain methods including CMF, CDCF, MLP++, and CSN, since they exploit information from both two domains. Another reason is that the text content contains richer information on the Amazon dataset. As it is shown in Table 1, the average number of words in the product reviews is more than that in the news titles. This will benefit all hybrid methods including HFT, TextBPR, and LCMR. We show the benefit of transferring source items by comparing with LCMR.

The hybrid TextBPR model composes a document representation by averaging the words's embeddings. This can not distinguish the important words to match user preferences. This may explain that it has some difficulty in improving the recommendation performance when integrating text content. For example, it cannot consistently outperform the pure CF method, MLP. The CSN model transfers every representations from the source network with the same coefficient. This may have a risk in transferring the noise and harm the performance, as pointed out in its sparse variant [22]. On the Amazon dataset, it is inferior to the TMH model by a large margin (though TMH leverages content information). In contrast, the memory and transfer components are both selective to extract useful information based on the attention mechanism. This may explain that our model is consistently the best at all settings.

There is a possibility that the noise from auxiliary domain and some irrelevance information contained in the unstructured text pose a challenge for exploiting them. This shows that the proposed model is more effective since it can select useful representations from the source network and attentively focus on important words to match preferences of users. In summary, the empirical comparison results demonstrate the superiority of the proposed TMH model to exploit the text content and source domain knowledge for recommendation.

5.3 Impact of Text and Source Domain

We have shown the effectiveness of the two memory and transfer components together in the proposed framework. We now investigate the contribution of each network to the TMH by eliminating the impact of text content and source domain from it in turn. 1) **TMH\M\T**: Eliminating the impact of both content and source information from TMH. This is a collaborative filtering recommender. Actually, it is equivalent to a single hidden layer MLP model. 2) **TMH\M**: Eliminating the impact of content information (MNet) from TMH. This is a novel cross-domain recommender which can adaptively select source items to transfer via the attentive weights. 3) **TMH\T**: Eliminating the impact of source information (TNet) from TMH. This is a novel hybrid filtering recommender which can attentively match word semantics with user preferences.

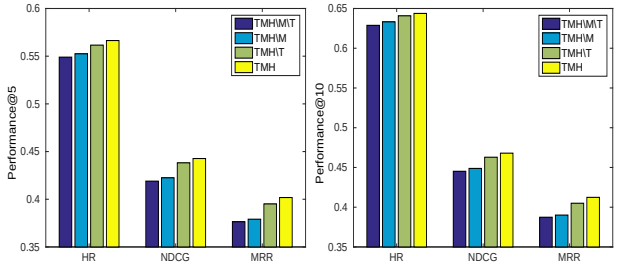

Figure 2: Contributions from unstructured text and cross-domain knowledge on Mobile Data.

The ablation analyses of TMH are shown in Figure 2 (Results on Amazon data are not shown due to space limit). The performance degrades when either memory or transfer modules is eliminated. This is understandable since we lose some information. In other words, the two components can extract useful knowledge to improve the recommendation performance. For example, TMH\T and TMH\M reduce 1.1% and 4.3% relative NDCG@10 performance, respectively, by comparing with TMH on the Mobile dataset (they are 8.5% and 16.1% on Amazon), suggesting that both memory and transfer networks learn essential knowledge for recommendation. On the two datasets, removing the memory component degrades the performance worse than removing the transfer component. This may be due to that the text content contains richer information or the source domain contains much more noise or both.

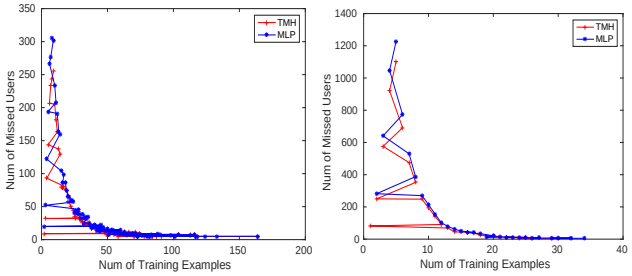

Figure 3: The Missed Hit Users (MHU) distribution (not normalized) over the number of training examples on the Mobile (left) and Amazon (right) datasets.

5.4 Improvement on Cold Users and Items

The cold-user and cold-item problems are common issues in recommender systems. When new users enter into a system, they have no history that can be exploited by the recommender system to learn their preferences, leading to the cold-user start problem. Similarly, when latest news are released on the Google News, there are no reading records that can be exploited by the recommender system to learn users' preferences on them, leading to the cold-item start problem. In general, it is very hard to train a reliable recommender system and make predictions for users and items that have few interactions. Intuitively, the proposed model can alleviate both the cold-user and cold-item start issues. TMH alleviates the cold-user start issue in the target domain by transferring his/her history from the related source domain. TMH alleviates the cold-item start issue by exploiting the associated text content to reveal its properties, semantics, and topics. We now investigate whether TMH indeed improves the performance over the cold users and items by comparing with the pure neural collaborative filtering method, MLP.

We analyse the distribution of Missed Hit Users (MHUs) of TMH and MLP (at cutoff 10). We expect that the cold users in MHUs can be reduced by using the TMH model. The more amount we can reduce, the more effective that TMH can alleviate the cold-user start issues. The results are shown in Figure 3 where the number of training examples can measure the "coldness" of a user. Naturally, the MHUs are most of the cold users who have few training examples. As we can see, the number of cold users in MHUs of MLP is higher than that of TMH. If the cold users are defined as those with less than seven training examples, then TMH reduces the number of cold users from 4,218 to 3,746 on the Amazon dataset, achieving relative 12.1% reduction. On the Mobile dataset, if the cold users are those with less than ten training examples (Mobile is denser than Amazon), then TMH reduces the number of cold users from 1,385 to 1,145 on the Mobile dataset, achieving relative 20.9% reduction. These results show that the proposed model is effective in alleviating the cold-user start issue. The results on cold items are similar and we omit them due to the page limit.

6 CONCLUSION

It is shown that the text content and the source domain knowledge can help improve the recommendation performance and they can be effectively integrated under a neural architecture. The sparse target user-item interaction matrix can be reconstructed with the knowledge from both kinds of information. The results demonstrate that our model outperforms the baseline that relies on the memory network only or relies on the transfer network only. In real-world services, data sources may belong to different providers. The data privacy is a big issue when we combine the multiple data sources. In our future work, it is worth developing new learning techniques to learn a combined model while protecting user privacy.

Acknowledgment The research has been supported by Hong Kong CERG projects 16211214/16209715/16244616, NSFC 61673202, and HKPFS PF15-16701.

REFERENCES

[1] T. Bansal, D. Belanger, and A. McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *ACM RecSys*, 2016.
[2] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, 2014.
[3] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. Cross-domain recommender systems. In *Recommender Systems Handbook*. 2015.
[4] R. Caruana. Multitask learning. *Machine Learning*, 1997.
[5] R. Catherine and W. Cohen. Transnets: Learning to transform for recommendation. In *ACM RecSys*, 2017.
[6] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *ACM SIGIR*, 2017.
[7] H.-T. Cheng, L. Koc, J. Harmsen, et al. Wide & deep learning for recommender systems. In *Workshop on Deep Learning for Recommender Systems*, 2016.
[8] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *ACM RecSys*, 2016.
[9] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 2004.
[10] G. Dziugaite and D. Roy. Neural network matrix factorization. *arXiv:1511.06443*, 2015.
[11] T. Ebesu, B. Shen, and Y. Fang. Collaborative memory network for recommendation systems. In *ACM SIGIR*, 2018.
[12] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *ACM SIGIR*, 2018.
[13] A. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*, 2015.
[14] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. Beyond the stars: improving rating predictions using review text content. In *WebDB*, 2009.
[15] Ming He, Jiuling Zhang, Peng Yang, and Kaisheng Yao. Robust transfer learning for cross-domain collaborative filtering using multiple rating patterns approximation. In *ACM WSDM*, 2018.
[16] R. He and J. McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. In *AAAI*, 2016.
[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, 2017.
[18] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
[19] G. Hu, Y. Zhang, and Q. Yang. Lcmr: Local and centralized memories for collaborative filtering with unstructured text. *arXiv preprint arXiv:1804.06201*, 2018.
[20] Guang-Neng Hu and Xin-Yu. Dai. Integrating reviews into personalized ranking for cold start recommendation. In *Pacific-Asia Knowledge Discovery and Data Mining*, 2017.
[21] Guang-Neng Hu, Xin-Yu Dai, Feng-Yu Qiu, Rui Xia, Tao Li, Shu-Jian Huang, and Jia-Jun Chen. Collaborative filtering with topic and social latent factors incorporating implicit feedback. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):23, 2018.
[22] Guangneng Hu, Yu Zhang, and Qiang Yang. Conet: Collaborative cross networks for cross-domain recommendation. *CIKM*, 2018.
[23] Haoran Huang, Qi Zhang, Xuanjing Huang, et al. Mention recommendation for twitter with end-to-end memory network. In *IJCAI*, 2017.
[24] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *ACM RecSys*, 2016.
[25] D. Kingma and J. Ba. Adam: A method for stochastic optimization. 2015.
[26] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.
[27] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, 2009.
[28] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item

co-occurrence. In *ACM RecSys*, 2016.
[29] Guang Ling, Michael R Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *ACM RecSys*, 2014.
[30] Bo Liu, Ying Wei, Yu Zhang, Zhixian Yan, and Qiang Yang. Transferable contextual bandit for cross-domain recommendation. In *AAAI*, 2018.
[31] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In *European conference on information retrieval*, 2014.
[32] Z. Lu, E. Zhong, L. Zhao, E. Xiang, W. Pan, and Q. Yang. Selective transfer learning for cross domain recommendation. In *SIAM International Conference on Data Mining*, 2013.
[33] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM RecSys*, 2013.
[34] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *IEEE CVPR*, 2016.
[35] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, 2008.
[36] S. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2010.
[37] W. Pan, N. Liu, E. Xiang, and Q. Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI*, 2011.
[38] Weike Pan, Evan W Xiang, Nathan N Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, 2010.
[39] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.
[40] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 2012.
[41] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.
[42] A. Singh and G. Gordon. Relational learning via collective matrix factorization. In *ACM SIGKDD*, 2008.
[43] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, 2015.
[44] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*, 2018.
[45] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *NIPS*, 2013.
[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.
[47] C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In *ACM SIGKDD*, 2011.
[48] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *ACM SIGKDD*, 2015.
[49] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*, 2018.
[50] Y. Wu, C. DuBois, A. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *ACM WSDM*, 2016.
[51] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *ACM SIGKDD*, 2017.
[52] D. Yang, J. He, H. Qin, Y. Xiao, and W. Wang. A graph-based recommendation across heterogeneous domains. In *ACM CIKM*, 2015.
[53] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *ACM SIGKDD*, 2016.
[54] Y. Zhang and Q. Yang. A survey on multi-task learning. *arXiv:1707.08114*, 2017.
[55] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *ACM SIGKDD*, 2017.
[56] L. Zheng, V. Noroozi, and P. Yu. Joint deep modeling of users and items using reviews for recommendation. In *ACM WSDM*, 2017.