MTNet: A Neural Approach for Cross-Domain Recommendation with Unstructured Text

name
affiliation
email

***Abstract*—Collaborative filtering (CF) is the key technique for recommender systems (RSs). CF exploits user-item behavior interactions (e.g., clicks) only and suffers from the data sparsity issue. One solution is to integrate the content information such as product reviews and news titles, leading to hybrid filtering methods. Another solution is to transfer knowledge from a related source domain such as improving the movie recommendation with the knowledge of the book domain, leading to cross-domain methods where transfer learning is a key technique. In real life, no single service can satisfy a user's all information needs. Thus it motivates us to exploit information from both the content and across domains for RSs in this paper. We achieve this by developing approaches to capture the text content and to transfer cross-domain knowledge. We propose a novel neural model, MTNet ("M" for memory and "T" for transfer), for cross-domain recommendation with unstructured text in an end-to-end manner. MTNet can attentively extract useful content via a memory network (MNet) and can selectively transfer knowledge from across domains by a transfer network (TNet), a novel network. The principle underlying these two components is the neural attention mechanism. A shared layer of feature interactions is stacked on the top to couple the high-level representations learned from individual networks. On two real-world datasets, MTNet shows better performance in terms of three ranking metrics by comparing with various baselines, including single/cross domain, shallow/deep, and hybrid methods. We conduct thorough analyses to understand how the text content and transferred knowledge help the proposed model.**

I. Introduction

Recommender systems are widely used in various domains and e-commerce platforms, such as to help consumers buy products at Amazon, watch videos on Youtube, and read articles on Google News. Collaborative filtering (CF) is among the most effective approaches based on the simple 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 which can learn the latent factors for users and items are its main cornerstone [23], [24]. Recently, neural networks like multilayer perceptron (MLP) are used to learn the interaction function from data [20], [21]. 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 content information such as unstructured text, like the news articles and product reviews. These additional sources of information can alleviate the sparsity issue and are essential for recommendation beyond user-item interaction data. For application domains like recommending research papers and news articles, the unstructured text associated with the item is its text content [8], [9]. Other domains like recommending products, the unstructured text associated with the item is its user reviews which justify the rating behavior of consumers [10], [18]. Topic modelling and neural networks have been proposed to exploit the item content and lead to performance improvement. Memory networks are widely used in question answering and reading comprehension to perform reasoning [14]. The memories can be naturally used to model additional sources like the item content [15], or to model a user's neighborhood who consume the common items with this user [25].

Another solution is to transfer the knowledge from relevant domains and the cross-domain recommendation techniques address such problems [28]–[30]. In real life, a user typically participates several systems to acquire different information services. For example, a user installs applications in an app store and reads news from a website at the same time. It brings us an opportunity to improve the recommendation performance in the target service (or all services) by learning across domains. Following the above example, we can represent the app installation feedback using a binary matrix where the 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) [6] approach 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 [34], [36]. In terms of neural networks, given two activation maps from two tasks, cross-stitch convolutional networks (CSN) [37] learn linear combinations of both the input activations and feed these combinations as input to the successive layers' filters, and hence enabling the knowledge transfer between two domains.

Thus it motivates us to exploit information from both the content and cross-domain information for RSs in this paper. To capture text content and to transfer cross-domain knowledge, we propose a novel neural model, MTNet, for cross-domain recommendation with unstructured text in an end-to-end manner. MTNet can attentively extract useful content via a memory network (MNet) and can selectively transfer knowledge across domains by a transfer network (TNet), a novel network. A shared layer of feature interactions is stacked on the top to

couple the high-level representations learned from individual networks. On real-world datasets, MTNet shows the better performance in terms of ranking metrics by comparing with various baselines. We conduct thorough analyses to understand how the content and transferred knowledge help MTNet.

To the best of our knowledge, MTNet is the first deep model that transfers cross-domain knowledge for recommendation with unstructured text in an end-to-end learning. Our contributions are summarized as follows:

- The proposed MTNet exploits the text content and transfers the source domain using an attention mechanism which is trained in an end-to-end manner. It is the first deep model that transfers cross-domain knowledge for recommendation with unstructured text using attention based neural networks.
- The memory component (MNet) can attentively exploit the text content to match word semantics and user preferences. It is among a few recent works on adopting memory networks for hybrid recommendations.
- The transfer component (TNet) can selectively transfer source items with the guidance of target user-item interactions. It is a novel transfer network for cross-domain recommendation.
- The proposed model can alleviate the sparsity issue including cold-user and cold-item start, and outperforms various baselines in terms of ranking metrics on two real-world datasets.

The paper is organized as follows. We firstly introduce the problem formulation in Section II-A. Then we present the memory component (MNet) to exploit the text content in Section II-B and the transfer component (TNet) to transfer cross-domain knowledge in Section II-C respectively. We propose the neural model MTNet for cross-domain recommendation with unstructured text in Section II-D, followed by its model learning (Section II-E) and complexity analyses (Section II-F). In Section III, we experimentally demonstrate the superior performance of the proposed model over various baselines (Sec. III-B). We show the benefit of both transferred knowledge and text content for the proposed model in Section III-C. We can reduce the amount of cold users and cold items that are difficult to predict accurately (Section III-D) and hence alleviate the cold-start issues. We review related works in Section IV and conclude the paper in Section V.

II. The Proposed MTNet Model

We describe the proposed MTNet model in this section. MTNet models user preferences in the target domain by exploiting the text content and transferring knowledge from a source/auxiliary domain. MTNet learns high-level representations for unstructured text and source domain items such that the learned representations can estimate the conditional probability of that whether a user will like an item. This is done with a memory network (Sec. II-B) and a novel transfer network (Sec. II-C), coupled by the shared embeddings on the bottom and an interaction layer on the top (Sec. II-D). The entire network can be trained efficiently to minimize a binary cross-entropy loss by back-propagation (Sec. II-E). We begin by describing the recommendation problem and the model formulation before introducing the network architecture.

A. Problem and Model Formulation

We have a target domain (e.g., news domain) user-item interaction matrix $\boldsymbol{R}_T \in \mathbb{R}^{m \times n_T}$ and a source domain (e.g., app domain) matrix $\boldsymbol{R}_S \in \mathbb{R}^{m \times n_S}$ where $m = |\mathcal{U}|$ and $n_T = |\mathcal{I}_T|$ ($n_S = |\mathcal{I}_S|$) is the size of users $\mathcal{U}$ and target items $\mathcal{I}_T$ (source items $\mathcal{I}_S$). Note that the users are shared and hence we can transfer knowledge across domains. We use u to index users, i to target items and j to source items. The entry $r_{ui} \in \{0, 1\}$ is one if the user u interacted with the item i and zero otherwise. Let $[j]^u = (j_1, j_2, ..., j_s)$ be the s-sized source items that user u has interacted with in the source domain.

The target domain also has the content information (e.g., product reviews). Denote by d_{ui} the content text corresponding to user u and item i. It is a sequence of words $d_{ui} = (w_1, w_2, ..., w_l)$ where each word w comes from a vocabulary $\mathcal{V}$ and $l = |d_{ui}|$ is the length of the text document.

For the task of item recommendation, the goal is to generate a ranked list of items for each user based on her history records, i.e., top-N recommendations. We hope improve the recommendation performance in the target domain with the help of both the content and source domain information.

Denote by the vector $\boldsymbol{r}_u = (r_{u1}, r_{u2}, ..., r_{un_T})$ the interactions of user u. The proposed MTNet models the probability of his/her each observation conditioned on this user, the content text, and the interacted source items:

$$\hat{r}_{ui} \triangleq p(r_{ui} = 1 | u, d_{ui}, [j]^u). \tag{1}$$

The equation sharpens the intuition behind the MTNet model, that is, the conditional probability of whether user u will like the item i can be determined by three factors: 1) his/her individual preferences, 2) the corresponding content text (d_{ui}), and 3) his/her behavior in a related source domain ($[j]^u$). The likelihood function of the entire matrix $\mathbf{R}_T$ is then defined as:

$$p(\mathbf{R}_T) = \Pi_u \Pi_i p(r_{ui} | u, d_{ui}, [j]^u). \tag{2}$$

The proposed MTNet is a neural network to learn the conditional probability in an end-to-end manner (see Fig. 1):

$$\hat{r}_{ui} = f(u, i, d_{ui}, [j]^u | \Theta_f), \tag{3}$$

where f is the network function and Θ_f are model parameters.

The model consists of a memory network $\mathbf{o}_{ui} = f_M(u, i, d_{ui} | \Theta_M)$ to model unstructured text (Sec. II-B) and a transfer network $\mathbf{c}_{ui} = f_T(i, [j]^u | \Theta_T)$ to transfer knowledge from the source domain (Sec. II-C). A shared feature interaction layer $f_S(\mathbf{o}_{ui}, \mathbf{z}_{ui}, \mathbf{c}_{ui} | \Theta_S)$ where $\mathbf{z}_{ui}$ is the non-linear representations of the (u, i) interaction, is stacked on the top of the learned high-level representations from individual networks (Sec. II-D).

B. MNet: Modeling Unstructured Text

We introduce the memory component, MNet, to exploit the content information, i.e., unstructured text. MNet is a variant of memory augmented neural network which can learn high-level representations of unstructured text with respect to the given user-item interaction. The attention mechanism inherent in the memory component can determine which words are highly relevant to the user preferences.

The MNet consists of one internal memory matrix $\mathbf{A} \in \mathbb{R}^{L \times 2d}$ where L is the vocabulary size (typically $L = 8000$) 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 $d_{ui} = (w_1, w_2, ..., w_l)$ corresponding to the (u, i) interaction, we form the memory slots $\boldsymbol{m}_k \in \mathbb{R}^d$ by mapping each word w_k into an embedding vector with matrix $\mathbf{A}$, where $k = 1, ..., l$ and the length of the longest document is the memory size l_{max}. We form a preference vector $\boldsymbol{q}$ corresponding to the given document d_{ui} and the user-item interaction (u, i) where each element $\boldsymbol{q}_k$ encodes the relevance of user u to these words given item i as:

$$q_k = \boldsymbol{x}_u^T \boldsymbol{m}_k^{(u)} + \boldsymbol{x}_i^T \boldsymbol{m}_k^{(i)}, \tag{4}$$

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 $\boldsymbol{m}_k^{(i)}$. The $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$ are the user and item embeddings obtained by embedding matrices $\boldsymbol{P} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{Q} \in \mathbb{R}^{n_T \times d}$ respectively. On the right hand of the above equation, the first term captures the matching between preferences of user u and word semantics, for example, the user is a machine learning researcher and he/she may be more interested in the words such as "optimization" and "Bayesian" than those of "history" and "philosophy". The second term computes the support of item i to the words, for example, the item is a machine learning related article and it may support more the words such as "optimization" and "Bayesian" than those of "history" and "philosophy". Together, the content-based/associative addressing scheme can determine the internal memories with highly relevance to the target user u regarding the words d_{ui} given the specific item i.

Actually we can compact the above two terms with a single vector dot product by concatenating the embeddings of the user and the item into $\boldsymbol{x}_{ui} = [\boldsymbol{x}_u, \boldsymbol{x}_i]$:

$$q_k = \boldsymbol{x}_{ui}^T \boldsymbol{m}_k, \ k = 1, 2, ..., l. \tag{5}$$

The neural attention mechanism can adaptively learn the weighting function over the words to focus on a subset of them. Traditional combination of words predefines a heuristic weighting function such as average or weights with tf-idf scores. Instead, we compute the attentive weights over words for a given user-item interaction to infer the importance of each word's unique contribution:

$$p_k = Softmax(q_k) = \frac{\exp(\beta q_k)}{\sum_{k'=1}^{l} \exp(\beta q_{k'})}, \tag{6}$$

which produces a probability distribution over the words in d_{ui}. The neural attention mechanism allows the memory component to focus on specific words while to place little importance on other words which may be less relevant. The parameter β is introduced to stabilize the numerical computation when the exponentials of the softmax function are very large and it also can amplify or attenuate the precision of the attention like a temperature [4] where a higher temperature (i.e., a smaller β) produces a softer probability distribution over words. We set $\beta = d^{-\frac{1}{2}}$ by scaling along with the dimensionality [5].

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

$$\boldsymbol{o}_{ui} = \sum\nolimits_k p_k \boldsymbol{c}_k, \tag{7}$$

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 $\mathbf{C}$. The external memories allows the storage of long-term knowledge pertaining specifically to each word's role in matching the user preference. In other words, the content-based addressing scheme identifies important words in a document acting as a key to retrieval the relevant values stored in the external memory matrix $\boldsymbol{C}$ via the neural attention mechanism. The attention mechanism adaptively weights the words according to the specific user and item. The final output $\boldsymbol{o}_{ui}$ represents a high-level, summarized information extracted attentively from the text content involved with the relations between the user-item interaction (u, i) and the corresponding words d_{ui}.

C. TNet: Transferring Cross-Domain Knowledge

We introduce the transfer component, TNet, to exploit the source domain knowledge. TNet is a novel network which can selectively transfer knowledge for cross-domain recommendation. The central idea is to learn adaptive weights over source domain items specific to the given target user-item interaction during the knowledge transfer.

Given the source items $[j]^u = (j_1, j_2, ..., j_s)$ 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 underlying observations can be illustrated in an example of improving the movie recommendation by transferring knowledge from the book domain. When we predict the preference of a user on the movie "The Lord of the Rings," the importance of her read books such as "The Hobbit," and "The Silmarillion" may be much higher than those such as "Call Me by Your Name".

The similarities between target item i and source items can be computed by their dot products:

$$a_j^{(i)} = \boldsymbol{x}_i^T \boldsymbol{x}_j, \ j = 1, ..., s, \tag{8}$$

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. For example, the similarity of target movie $i =$ "The Lord of the Rings," with the source book $j =$

"The Hobbit" may be larger than that with the source book j' = "Call Me by Your Name" (given a user u).

We normalize similarity scores to be a probability distribution over source items:

$$\alpha_j = Softmax(a_j^{(i)}), \tag{9}$$

and then the transfer vector is a weighted sum of the corresponding source item embeddings:

$$\boldsymbol{c}_{ui} = ReLU(\sum\nolimits_j \alpha_j \boldsymbol{x}_j), \tag{10}$$

where we introduce non-linearity on the transfer vector by activation function rectified linear unit (ReLU). Empirically we found that the activation function $ReLU(x) = max(0, x)$ works well due to its non-saturating nature and suitability for sparse data.

The transfer vector $\boldsymbol{c}_{ui}$ is a high-level representation, summarizing the knowledge from the source domain as the output of the TNet. TNet can selectively transfer representations from the corresponding embeddings of source items with the guidance of the target user-item interaction.

Remark I Memory networks are proposed to address the task of question answering where the memories are a short story text and the query/input is a question related to the story for which the answer can be reasoned by the network. We can think of the recommendation with unstructured text as a question answering problem: the question to be addressed is to ask how likely a user prefers an item where the text content is analogue to the story text and the query is analogue to the given user-item interaction.

Remark II The computational processes of MNet and TNet are similar. We firstly compute attentive weights over a collection of objects (content words in MNet and source items in TNet). Then we summarize the high-level representation as the output (the text representation $\boldsymbol{o}_{ui}$ in MNet and the transfer vector $\boldsymbol{c}_{ui}$ in TNet), weighted by the attentive probabilities which are computed by a content-based addressing scheme.

D. MTNet: The Proposed Neural Model

The architecture for the proposed MTNet model is illustrated in Figure 1 as a multi-layer feedforward neural network. The input layer specifies embeddings of a user u, a target item i, and the corresponding source items $[j]^u = (j_1, ..., j_s)$. The content text d_{ui} is modelled by the memories in the MNet to produce a high-level representation $\boldsymbol{o}_{ui}$. The source items are transferred into the transfer vector $\boldsymbol{c}_{ui}$ with the guidance of (u, i) in the TNet. These computational pathes are introduced in the above Sec. II-B and Sec. II-C respectively.

We now propose the MTNet model. Firstly, we use a single hidden layer network to learn a nonlinear representation for the user-item interaction:

$$\boldsymbol{z}_{ui} = ReLU(\boldsymbol{W}\boldsymbol{x}_{ui} + \boldsymbol{b}), \tag{11}$$

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.

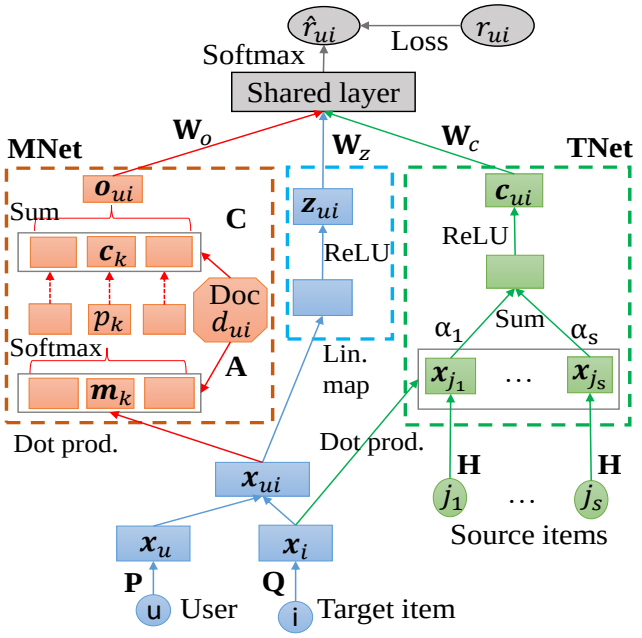

Fig. 1. The Proposed MTNet Architecture. The memory network (MNet) is to model unstructured text. The transfer network (TNet) is to transfer cross-domain knowledge. The middle part is to learn the user-item interaction function. The shared layer is to learn feature interactions from the outputs of individual networks.

TABLE I
MODEL PARAMETERS OF MTNET.

Parameter	Dimension	Description
$\boldsymbol{P}$	$m \times d$	User embedding matrix
$\boldsymbol{Q}$	$n_T \times d$	Target item embedding matrix
$\boldsymbol{H}$	$n_S \times d$	Source item embedding matrix
$\boldsymbol{A}$	$L \times 2d$	Internal memory matrix
$\boldsymbol{C}$	$L \times 2d$	Internal memory matrix
$\boldsymbol{W}, \boldsymbol{b}$	$2d \times d$, d	Linear mapping weight and bias for the user-item interaction
$\boldsymbol{W}_o, \boldsymbol{W}_z, \boldsymbol{W}_c$	$d \times d$	Linear mapping for outputs of individual networks
$\boldsymbol{h}$	$3d$	Weight of the shared layer

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:

$$\hat{r}_{ui} = \frac{1}{1 + \exp(-\boldsymbol{h}^T \boldsymbol{y}_{ui})}, \tag{12}$$

where $\boldsymbol{h}$ is the parameter and the joint representation:

$$\boldsymbol{y}_{ui} = [\boldsymbol{W}_o \boldsymbol{o}_{ui}, \boldsymbol{W}_z \boldsymbol{z}_{ui}, \boldsymbol{W}_c \boldsymbol{c}_{ui}], \tag{13}$$

is concatenated from the linear mapped outputs of individual networks.

E. Model 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:

$$\mathcal{L} = -\sum\nolimits_{(u,i)\in\mathcal{S}} r_{ui}\log\hat{r}_{ui} + (1-r_{ui})\log(1-\hat{r}_{ui}), \quad (14)$$

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.

This objective function has a probabilistic interpretation and is the negative logarithm likelihood of the following likelihood function:

$$L(\Theta|\mathcal{S}) = \prod\nolimits_{(u,i)\in\boldsymbol{R}_T^+} \hat{r}_{ui} \prod\nolimits_{(u,i)\in\boldsymbol{R}_T^-} (1-\hat{r}_{ui}), \quad (15)$$

where the model parameters are:

$$\Theta = \{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{H}, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{W}_o, \boldsymbol{W}_z, \boldsymbol{W}_c, \boldsymbol{h}\}. \quad (16)$$

Comparing with Eq.(2), instead of modeling all zero entries (i.e., the whole target matrix $\boldsymbol{R}_T$), we learn from only a small subset of such unobserved entries and treat them as negative samples by picking them randomly during each optimization iteration (i.e., the negative sampling technique).

The objective function can be optimized by stochastic gradient descent (SGD) and its variants like adaptive moment method (Adam) [1]. The update equations are:

$$\Theta^{new} \leftarrow \Theta^{old} - \eta\frac{\partial L(\Theta)}{\partial\Theta}, \quad (17)$$

where η is the learning rate. Typical deep learning library like TensorFlow (https://www.tensorflow.org) provides automatic differentiation and hence we omit the gradient equations $\frac{\partial L(\Theta)}{\partial\Theta}$ which can be computed by chain rule in back-propagation (BP).

F. Complexity Analysis

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 $L = 8000$. 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 [6] and one hidden layer neural CF approaches [21].

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. MTNet is scalable to the number of the training data. It can easily update when new data examples come, just feeding them into the training mini-batch. Thus, MTNet can handle the scalability and dynamics of items and users like in an online fashion. In contrast, the topic modeling related techniques have difficulty in benefitting from these advantages to this extent.

TABLE II
DATASETS AND STATISTICS.

Dataset	Domain	Statistics	Amount
Mobile News	Shared	#Users	15890
	Target	#News	84802
		#Reads	477685
		Density	0.035%
		#Words	612,839
		Avg. Words Per News	7.2
	Source	#Apps	14340
		#Installations	817120
		Density	0.359%
Amazon Men	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%

III. EXPERIMENTS

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

A. Experimental Settings

Dataset We evaluate on two real-world cross-domain datasets. The first dataset, **Mobile**[1], is provided by a large internet company, i.e., Cheetah Mobile (http://www.cmcm.com/en-us/). 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. 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%.

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 [12]. We use the two categories of Amazon Men and Amazon Sports as the cross-domain. The original ratings

[1]An anonymous version can be released later.

are from 1 to 5 where five stars indicate that the user shows a positive preference on the item while the one stars are 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. The average number of words per review is 32.9.

The statistics of the two datasets are summarized in Table II. As we can see, both datasets are very sparse and hence we hope 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 is short text of news titles (the number of average words per item is 7).

Evaluation Protocol For item recommendation task, the leave-one-out (LOO) evaluation is widely used and we follow the protocol in [21]. 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 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, defined as:

$$HR = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \delta(p_u \leq topK), \tag{18}$$

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 defined as:

$$NDCG = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \frac{\log 2}{\log(p_u + 1)}, \tag{19}$$

$$MRR = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \frac{1}{p_u}. \tag{20}$$

A higher value with lower cutoff indicates better performance.

Baselines We compare with various baselines, categorized as single/cross domain, shallow/deep, and hybrid methods.

Baselines	Shallow method	Deep method
Single-Domain	BPRMF	MLP
Cross-Domain	CDCF, CMF	MLP++, CSN
Hybrid	HFT, TextBPR	LCMR
Cross + Hybrid	CDCF++	MTNet (ours)

- **BPRMF**, Bayesian personalized ranking [3], is a latent factor model based on matrix factorization and pair-wise loss. It learns on the target domain only.
- **CDCF**, Cross-Domain CF with factorization machines (FM) [2], is a cross-domain recommender which extends FM [22]. It is a context-aware approach which applies factorization on the merged domains (aligned by the shared users). That is, the auxiliary domain is used as context. On the Mobile dataset, the context for a user in the target news domain is his/her history of app installations in the source app domain. The feature vector for the input is a sparse vector $\boldsymbol{x} \in \mathbb{R}^{m+n_T+n_S}$ where the non-zero entries are as follows: 1) the index for user id, 2) the index for target news id (target domain), and all indices for his/her installed apps (source domain).
- **CDCF++**: We extend the above CDCF model to exploit the text content. The feature vector for the input is a sparse vector $\boldsymbol{x} \in \mathbb{R}^{m+n_T+n_S+L}$ where the non-zero entries are augmented by the word features corresponding to the given user-item interaction. In this way, CDCF++ can learn from both the source domain and unstructured text information.
- **CMF**, Collective matrix factorization [6], is a multi-relation learning approach which jointly factorizes matrices of individual domains. Here, the relation is the user-item interaction. On Mobile, the two matrices are $\boldsymbol{A}$ = "user by news" and $\boldsymbol{B}$ = "user by app" respectively. The shared user factors $\boldsymbol{P}$ enable knowledge transfer between two domains. Then CMF factorizes matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ simultaneously by sharing the user latent factors: $\boldsymbol{A} \approx \boldsymbol{P}^T \boldsymbol{Q}_A$ and $\boldsymbol{B} \approx \boldsymbol{P}^T \boldsymbol{Q}_B$. It is a shallow model and jointly learns on two domains. This can be thought of a non-deep transfer/multitask learning approach for cross-domain recommendation.
- **HFT**, Hidden Factors and hidden Topics [10], adopts topic distributions to learn latent factors from text reviews. It is a hybrid method.
- **TextBPR** extends the basic BPRMF model by integrating text content. It computes the prediction scores by two parts: one is the standard latent factors, same with the BPRMF; and the other is the text factors learned from the text content. It has two implementations, the VBPR model [12] and the TBPR model [11] which are the same in essence.
- **MLP**, multilayer perceptron [21], is a neural CF approach which learns the nonlinear interaction function using neural networks. It is a deep model learning on the target domain only.
- **MLP++**: We combine two MLPs by sharing the user embedding matrix, enabling the knowledge transfer between two domains through the shared users. It is a naive knowledge transfer approach applied for cross-domain recommendation.
- **CSN**, Cross-stitch network [37], is a deep multitask learning model originally proposed for visual recognition tasks. We use the cross-stitch units to stitch two MLP networks. It learns a linear combination of activation maps from two networks and hence benefits from each other. Comparing with MLP++, CSN enables knowledge

TABLE III
COMPARISON RESULTS OF DIFFERENT METHODS ON THE MOBILE DATASET. THE BEST BASELINES ARE MARKED WITH ASTERISKS AND THE BEST RESULTS ARE BOLDFACED.

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
MTNet	**.5664**	**.4427**	**.4018**	**.6438**	**.4680**	**.4124**	**.6983**	**.4820**	**.4163**
Improvement of MTNet	2.04%	2.42%	2.51%	1.75%	2.32%	2.47%	0.81%	1.86%	2.34%

transfer also in the hidden layers besides the lower embedding matrices. This is a deep transfer learning approach for cross-domain recommendation.

- **LCMR**, Local and Centralized Memory Recommender [15], is a deep model for collaborative filtering with unstructured Text. The local memory module is similar to our MNet except that we only have one layer. This is a deep hybrid method.

Implementation For BPRMF, we use LightFM's implementation[2] which is a popular CF library. For CDCF and CDCF++, we adapt the official libFM implementation[3]. For CMF, we use a Python version reference to the original Matlab code[4]. For HFT and TextBPR, we use the code released by their authors[5]. The word embeddings used in the TextBPR are pre-trained by GloVe[6]. 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[7]. The MLP++ and CSN are implemented based on MLP. The LCMR model is similar to our MNet model and thus implemented in company. 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 reference in the original paper [21]. For CSN, it requires that the number of neurons in each hidden layer is the same and the configuration is $[64] * 4$ (equals $[64 \rightarrow 64 \rightarrow 64 \rightarrow 64]$). We investigate several typical configurations $\{16, 32, 64, 80\} * 4$. The dimension of embeddings is $d = 75$.

[2]https://github.com/lyst/lightfm
[3]http://www.libfm.org
[4]http://www.cs.cmu.edu/~ajit/cmf/
[5]http://cseweb.ucsd.edu/~jmcauley/
[6]https://nlp.stanford.edu/projects/glove/
[7]https://github.com/hexiangnan/neural_collaborative_filtering

B. Comparisons of Different Recommender Systems

In this section, we report the recommendation performance of different methods and discuss the findings. The comparison results are shown in Table III and Table IV respectively on the Mobile and Amazon datasets where the last row is the relative improvement of ours vs the best baseline. We have the following observations.

Firstly, we can see that our proposed neural models are better than all baselines on the two datasets at each setting, including the base 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 MTNet 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 that on the Mobile dataset, obtaining average 9.56% relative improvements over the best CSN baseline, though the Amazon is sparser than the Mobile (see Table II). One explanation is that the relatedness of the Men and Sports domains is closer than that 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 explanation is that the text content contains richer information on the Amazon dataset. As it is shown in Table II, the average words in the product reviews are longer that in the news titles. This will benefit all hybrid methods including HFT, TextBPR, and LCMR.

The hybrid TextBPR model composes a document representation by averaging the words's embeddings. This can not distinguish the important words to match the user preferences.

TABLE IV
COMPARISON RESULTS OF DIFFERENT METHODS ON THE AMAZON DATASET. THE BEST BASELINES ARE MARKED WITH ASTERISKS AND THE BEST RESULTS ARE BOLDFACED.

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
MTNet	**.2575**	**.1796**	**.1550**	**.3490**	**.2077**	**.1666**	**.4443**	**.2311**	**.1727**
Improvement of MTNet	10.04%	6.90%	6.01%	15.63%	9.43%	7.34%	12.65%	10.52%	7.60%

This may explain that it has difficulty in improving the recommendation performance when integrating text content. For example, it can not consistently outperform the pure CF method, MLP. The cross-domain 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 [7]. On the Amazon dataset, it loses to the proposed model by a large margin (though MTNet 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 propose 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 the important words to match preferences of users.

In summary, the empirical comparison results demonstrate the superiority of the proposed neural model to exploit the text content and source domain knowledge for recommendation.

C. The Impact of Unstructured Text and Auxiliary 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 MTNet by eliminating the impact of text content and source domain from it in turn:

- **MTNet\M\T**: Eliminating the impact of both content and source information from MTNet. This is a collaborative filtering recommender. Actually, it is equivalent to a single hidden layer MLP model.
- **MTNet\M**: Eliminating the impact of content information (MNet) from MTNet. This is a novel cross-domain recommender which can adaptively select source items to transfer.
- **MTNet\T**: Eliminating the impact of source information (TNet) from MTNet. This is a novel hybrid filtering recommender which can attentively select content words to exploit.

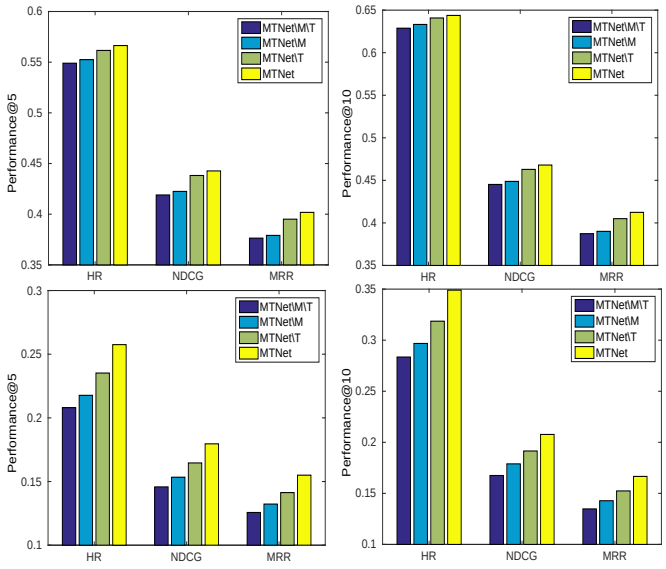

Fig. 2. Contributions from Unstructured Text (MNet) and Cross-Domain Knowledge (TNet) on the Mobile (Top) and Amazon (Bottom) Datasets.

The ablation analyses of MTNet and its components are shown in Figure 2. The performance degrades when either memory or transfer modules are 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, MTNet\T and MTNet\M respectively reduce 1.1% and 4.3% relative NDCG@10 performance by comparing with MTNet 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 evaluated two datasets, removing the memory component degrades performance worse than that of removing the transfer component.

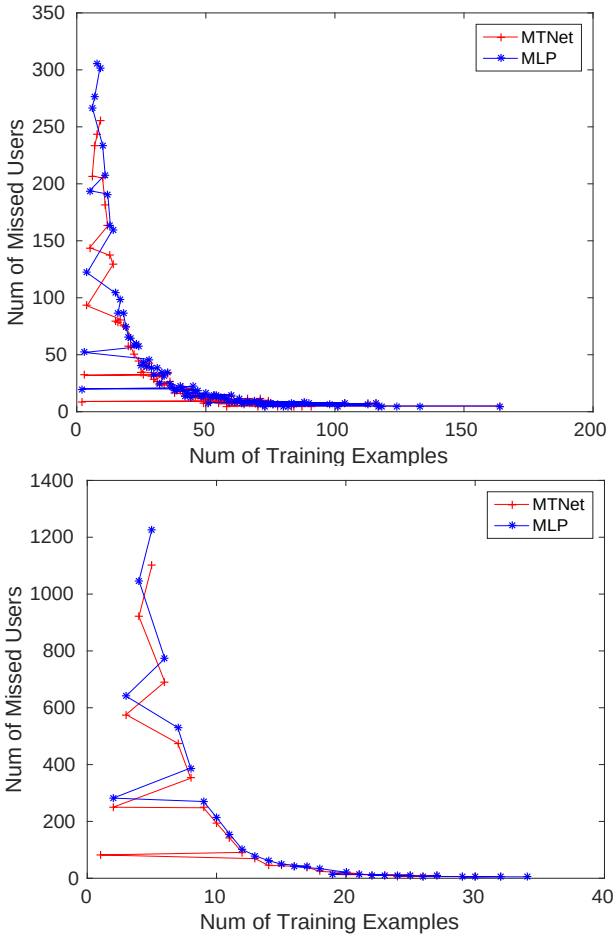

Fig. 3. The Missed Hit Users Distribution (not normalized) Over the Number of Training Examples on the Mobile (Top) and Amazon (Bottom) Dataset.

D. Improvement Over 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. MTNet alleviates the cold-user start issue in the target domain by transferring his/her history from the related source domain. MTNet alleviates the cold-item start issue by exploiting the associated text content to reveal its properties, semantics, and topics. We now investigate that MTNet 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 MTNet and MLP (at cutoff 10). We expect that the cold users in MHUs of MLP can be reduced by using the MTNet model. The more amount we can reduce, the more effective that MTNet 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 MTNet. If the cold users are defined as those with less than seven training examples, then MTNet 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 MTNet 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.

IV. Related Works

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. Collaborative filtering (CF) alleviates this issue by predicting user preferences based on the user-item interaction behavior, agnostic to the content [27]. Latent factor models learn feature vectors for users and items mainly based on MF [24] which has probabilistic interpretations [23]. FM can mimic MF [22]. Neural networks are proposed to push the learning of feature vectors towards non-linear representations, including the NNMF and MLP [20], [21]. The basic MLP architecture is extended to regularize the factors of users and items by social and geographical information [17]. Other neural approaches learn from the explicit feedback for rating prediction task [18], [19]. We focus on learning from the implicit feedback for top-N recommendation [16]. CF models, however, suffer from the data sparsity issue.

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 [8], [9], [13] and user reviews [10]–[12]. Memory networks can reason with an external memory [14]. Due to the capability of naturally learning word embeddings to address the problems of word sparseness and semantic gap, a memory module can be used to model item content [15] or the neighborhood of users [25]. We follow this research thread by using neural networks to attentively extract important information from the text content.

Cross-domain recommendation [28] is an effective technique to alleviate sparse issue. A class of methods are based on MF applied to each domain, including CMF [6] with its heterogeneous [29] variants, and codebook transfer [30]. Heterogeneous cross-domain [31], multiple source

domains [32], and multi-view learning [33] are also proposed. Transfer learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [34]. Similar to TL, the multitask learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [35], [36]. The cross-stitch network [37] enables information sharing between the two base networks. We follow this research thread by using neural networks to selectively transfer knowledge from the source items.

V. Conclusion

It is shown that the text content and the source domain knowledge can help improve recommendation performance and can be integrated under a neural architecture. The sparse target user-item interaction matrix can be reconstructed with the knowledge guidance from both of the two kinds for information, alleviating the data sparse issue. We proposed a novel deep neural model, MTNet, for cross-domain recommendation with unstructured text. MTNet consists of a memory component which can attentively focus important words to match user preferences and a transfer component which can selectively transfer useful source items to benefit the target domain. MTNet shows better performance than various baselines on two real-world datasets under different settings. Additionally, we conducted ablation analyses to understand the contributions from the two components. We quantify the amount of missed hit cold users that MTNet can reduce by comparing with the pure CF method, showing that MTNet is able to alleviate the cold-start issue.

References

[1] Kingma, D. and Ba, J., 2014. Adam: A method for stochastic optimization. *ICLR*, 2015

[2] Loni, B., Shi, Y., Larson, M. and Hanjalic, A. Cross-domain collaborative filtering with factorization machines. In *European conference on information retrieval*, 2014

[3] Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In *Uncertainty in artificial intelligence*, 2009

[4] Hinton, G., Vinyals, O. and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. and Polosukhin, I. Attention is all you need. In *NIPS*, 2017

[6] Singh, A. and Gordon, G. Relational learning via collective matrix factorization. *ACM SIGKDD*, 2008

[7] Hu, G., Zhang, Y. and Yang, Q. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. *arXiv preprint arXiv:1804.06769*, 2018

[8] Wang, C. and Blei, D. Collaborative topic modeling for recommending scientific articles. *ACM SIGKDD*, 2011

[9] Bansal, T., Belanger, D. and McCallum, A. Ask the GRU: Multi-task learning for deep text recommendations. In *ACM Recommender Systems*, 2016

[10] McAuley, J. and Leskovec, J. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM Recommender systems*, 2013

[11] Hu, G. and Dai, X. Integrating reviews into personalized ranking for cold start recommendation. In *Pacific-Asia Knowledge Discovery and Data Mining*, 2017

[12] He, R. and McAuley, J. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *AAAI*, 2016

[13] Wang, H., Wang, N. and Yeung, D. Collaborative deep learning for recommender systems. In *ACM SIGKDD*, 2015

[14] Sukhbaatar, S., Weston, J. and Fergus, R. End-to-end memory networks. In *NIPS*, 2015

[15] Hu, G., Zhang Y., Yang, Q. LCMR: Local and Centralized Memories for Collaborative Filtering with Unstructured Text. *arXiv preprint arXiv:1804.06201*, 2018

[16] Wu, Y., DuBois, C., Zheng, A. and Ester, M. Collaborative denoising auto-encoders for top-n recommender systems. In *ACM WSDM*, 2016

[17] Yang, C., Bai, L., Zhang, C., Yuan, Q. and Han, J. Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation. In *ACM SIGKDD*, 2017

[18] Zheng, L., Noroozi, V. and Yu, P. Joint deep modeling of users and items using reviews for recommendation. In *ACM WSDM*, 2017

[19] Catherine, R. and Cohen, W. TransNets: Learning to Transform for Recommendation. In *ACM Recommender Systems*, 2017

[20] Dziugaite, G. and Roy, D. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015

[21] Chen, J., Zhang, H., He, X., Nie, L., Liu, W. and Chua, T. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *ACM SIGIR*, 2017

[22] Rendle, S. Factorization machines. In *IEEE ICDM*, 2010

[23] Mnih, A. and Salakhutdinov, R. Probabilistic matrix factorization. In *NIPS*, 2008

[24] Koren, Y., Bell, R. and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer, 42(8)*, 2009

[25] Ebesu, T., Shen, B., and Fang, Y. Collaborative Memory Network for Recommendation Systems. *ACM SIGIR*, 2018.

[26] Pazzani, M. and D. Billsus, D. Content-based recommendation systems. *The adaptive web*, 2007

[27] Deshpande, M. and Karypis, G. Item-based top-n recommendation algorithms. *ACM TOIS*, 2004.

[28] Cantador, I., Fernández-Tobí, I., Berkovsky, S. and Cremonesi, P. Cross-domain recommender systems. *Recommender Systems Handbook*, 2015

[29] Pan, W., Liu, N., Xiang, E., Yang, Q. Transfer learning to predict missing ratings via heterogeneous user feedbacks. *IJCAI*, 2011

[30] Li, B. Yang Q., and Xue, X. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, 2009

[31] Yang, D., He, J., Qin, H., Xiao, Y., and Wang, W. A graph-based recommendation across heterogeneous domains. In *ACM CIKM*, 2015

[32] Lu, Z., Zhong, E., Zhao, L., Xiang, E., Pan, W. and Yang, Q. Selective transfer learning for cross domain recommendation. In *SIAM SDM*, 2013

[33] Elkahky, A., Song, Y., and He, X. A multi-view deep learning approach for cross domain user modeling in recommendation systems. *WWW*, 2015

[34] Pan, S. and Yang, Q. A survey on transfer learning. *IEEE TKDE*, 2010.

[35] Caruana, R. Multitask Learning. *Machine Learning*, 1997.

[36] Zhang, Y. and Yang, Q. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017

[37] Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. *IEEE CVPR*, 2016