Transfer Learning with Domain-aware Attention Network for Item Recommendation in E-commerce

Minghui Qiu[1] , **Bo Wang**[2] , **Cen Chen**[3] , **Xiaoyi Zeng**[2] and **Jun Huang**[1]
[1]Alibaba Group
[2]Taobao.com
[3]Ant Financial Services Group
{minghui.qmh, huangjun.hj}@alibaba-inc.com,
{boyi.wb, yuanhan}@taobao.com,
chencen.cc@antfin.com

Abstract

The booming of online shopping trends have reshaped the e-commerce industry, resulting in a growing number of e-commerce platforms over the decades. The underlying tasks for these e-commerce platforms are similar, it's thus beneficial to employ a unified cross-domain model to leverage knowledge from both platforms (i.e., kill two birds with one stone). In light of this, we propose a novel transfer learning method with domain-aware attention network, to help multiple platforms. More specifically, our model addresses the importance of features across domains via attention networks. Specific-shared feature embeddings are used to enhance feature representations for different domains, while a new adversarial regularizer is proposed to further helps learn domain-invariant and domain-specific features that are useful for all domains. To the best of our knowledge, our study is the first to build a unified model for transferring knowledge in the e-commerce setting. Extensive experiments on real-world datasets from e-commerce platforms show that our model outperforms the competing methods by a large margin and is able to help both data-rich and data-deficient domains. Meanwhile, our quantitative evaluation shows that our model can discover important features for different domains.

1 Introduction

The booming of online shopping has led to a growing number of e-commerce websites and APPs (e.g., Taobao.com, Amazon.com, jd.com and etc.). Although many e-commerce platforms employed differentiation strategy to target different user pool, they may still share a portion of overlapping users. For example, in Figure 1, two e-commerce platforms clearly target different users: one (platform 1) is notable for selling high-end/branded products, while the other is famous for economic/cheap items. The core task of these platform is item recommendation based on user query, profile and historical behaviours, which is similar and motivates us to build a unified model for these platforms. Meanwhile, we find that some platforms are with far less user click or purchase history than others. For example, platform 2's data size is less than 1% of platform 1's, thus makes it difficult to train a good model for platform 2 by only using its own data. Hence this further motivates to build a unified model to transfer knowledge over multiple platforms.

(a) Platform 1. (b) Platform 2.

Figure 1: Search results for 'SmartPhone' on different platforms.

To build such a cross-domain model, we ought to understand users' needs on different platforms. As shown in Figure 1, we search for the same keyword 'SmartPhone' in two big E-commerce APPs, one is Taobao[1] and the other is Qintao[2], the top purchased items are distinct. Since platform 1 targets high-end market, users tend to buy branded smartphones like 'Apple' and 'Huawei' in platform 1, while users in platform 2 tend to buy low-end brands like 'VIVI' or 'VIVL' in platform 2. Note that even for the same user, when looking for the same type of products on these two platforms, the item preference is different. Understanding the underlying difference between those two platforms, it's thus necessary to provide domain-aware recommendations accordingly. Naturally, we study cross-domain learning for this task to

[1]https://taobao.com
[2]https://qintao.taobao.com/

joint train a unified model with a data-rich domain and a data-insufficient domain, in the hope of improving both domains. Note that this is different from the typical transfer learning setting, which seeks to improve the target domain only, while our task here is to improve both source and target domains. Furthermore, a naive transferring model from platform 1 to platform 2 in Figure 1 will suffer from the negative transfer problem as the domain shift exists in the two domains.

To tackle this problem, we propose to jointly model user behaviors on different platforms. Our intuition is that different features shall play different importance in different domains. For example, a high-end brand will be more attractive in platform 1 instead of platform 2 as in Figure 1. To capture this, we propose a cross-domain attention network for modeling domain-specific feature importance.

Furthermore, our model is based on a transfer learning (TL) [Pan and Yang, 2010; Liu *et al.*, 2017b] framework, but seeks to improve both source and target domain. A typical framework uses a shared NN to learn shared features for both source and target domains [Mou *et al.*, 2016; Yang *et al.*, 2017]. This is improved by using shared-private model, i.e., a shared NN and domain-specific NNs, to derive shared and domain-specific features [Liu *et al.*, 2017b]. Recent studies [Ganin *et al.*, 2016; Taigman *et al.*, 2017; Liu *et al.*, 2017b; Shen *et al.*, 2018] consider adversarial networks to learn more robust shared features across domains. The study in [Shen *et al.*, 2018] shows using Wasserstein distance can help to learn domain-invariant features and achieve better performance than other adversarial losses. However, the base model in [Shen *et al.*, 2018] is a fully-shared model which may not be able to capture domain differences in our cross-domain setting. We then extend the method to the shared-private model. Our intuitions are: (i) shared features from both source domain and target domain should be close to each other, and (ii) domain specific features should be different than the shared features. To capture this, we consider three Wasserstein distances: D_c, D_s, and D_t. Here D_c captures the difference between source-shared features and target-shared features, D_s captures the difference between source domain features and source-shared features, and D_t captures the difference between target domain features and target-shared features. Experiments show this Wasserstein regularizer can help to find domain-invariant and domain-specific features.

Last but not least, the input features in our model are embedded with a distributed representation, a.k.a feature embeddings. We urge that only using shared feature representations [Liu *et al.*, 2017b; Chen *et al.*, 2018] may not be sufficient to capture the characteristic of features across domains. Hence we propose to embed each feature with both shared and domain-specific embeddings. The shared part captures the commonalities, while the domain-specific part captures the differences, which can be of a much smaller embedding dimension size to reduce model complexity.

In all, we summarize our contributions as follows:

- We propose a domain-aware attention network for modelling feature importance in multiple domains which are proved to be effective for all the domains. To the best of our knowledge, our study is the first to build a unified model for multiple e-commerce platforms;
- We propose a new type of adversarial loss to regularize the learnt shared and specific features w.r.t. both source and target domains.
- We evaluate our method on both public and industry datasets. Experiments show that our model outperforms the competing methods, and improves both source and target domains.
- A quantitative evaluation shows our model can discover important features for different domains which help to understand different user needs in these domains.

2 Model

Our model is designed to address the following general problem. Given an input $X_i^d = [f_i^0, f_i^1, \cdots, f_i^n]$, where n is feature length and $d \in \{src, tgt\}$ is the domain id. We seek to predict its label $y_i^d \in \{0, 1\}$. This setting is close to cross-domain Click-Through-Rate (CTR) prediction task. It's worth noting that we consider to jointly train a model for both src and tgt domains, i.e. we seek to build an unified model.

We present an overview of our model in Fig 3. In a nutshell, our model first obtains shared and domain-specific representations for each feature by looking up its feature embeddings. These are fed into attention layers to obtain domain-specific feature representations. These representations are fed into a domain-specific neural networks (NN) layer and a shared NN layer to obtain high-level representations. The output layers are applied on the hidden representations to generate output labels. To encourage learning domain-invariant features, we study adversarial training [Liu *et al.*, 2017a] in our method. Below we present our model in details.

2.1 Cross-Domain Neural Attention Network with Adversarial Training (CNAN-AT)

The base model mainly contains two major components: attention layer and neural network (NN) layers. The former obtains a gated representations of input features and the latter obtain abstract features from the representations.

Embedding layer.

For an input $X_i = [f_i^1, f_i^2, \cdots, f_i^n]$, our model first lookup feature embedding $e^j \in \mathbb{R}^{K \times 1}$ for each feature f_i^j ($j \in [1, n]$). Each feature embedding e^j is a combination of two types of embeddings: one is shared feature embedding e_c^j, the other is domain-specific (source- or target- specific) embeddings (e_0^j or e_1^j). The dimension of domain-specific embedding is far less than that of shared embedding to reduce model complexity. Note that such domain-specific embeddings can help to enhance each feature representation by modeling its domain-specific properties, which is shown to be helpful in our experiments. If the input is from source domain, we have $e^j = e_c^j \oplus e_0^j$, otherwise $e^j = e_c^j \oplus e_1^j$, where $\oplus$ is a concatenation operator.

Attention layer.

For each feature f_i^j ($j \in [1, n]$), its feature embedding $e_i^j \in \mathbb{R}^{K \times 1}$ is fed into an attention layer and neural network lay-

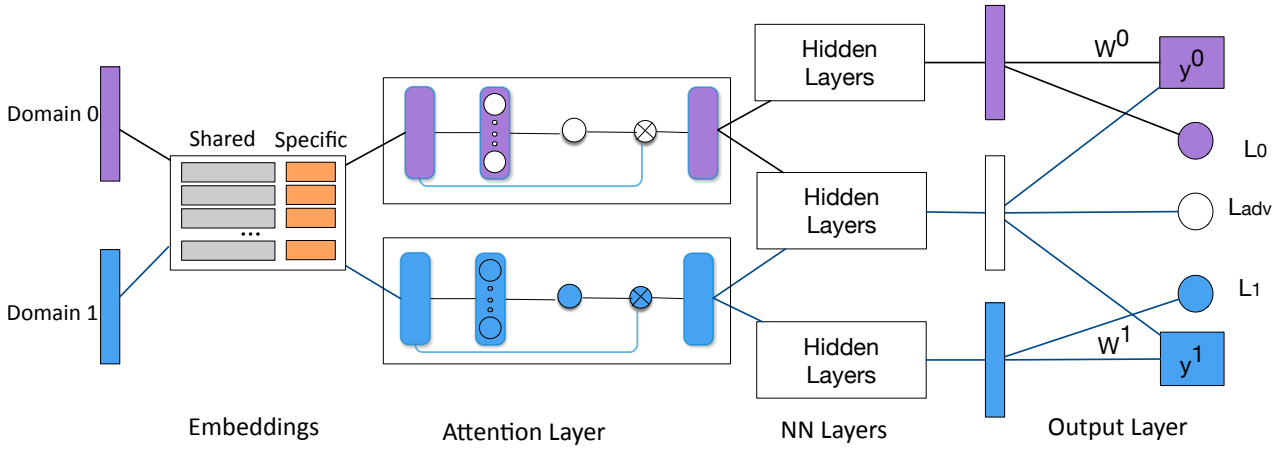

Figure 2: Cross-domain Neural Attention Networks with Adversarial Training (CNAN-AT). Each feature in our model has its specific and shared embeddings, which is fed into attention layer to get a new feature representation.

ers to obtain a gated representation $g^j \in \mathbb{R}^{K\times 1}$. Below we describe the underlying attention layer.

We present our attention layer in Fig. 3, and describe the process as follows:

$$\begin{aligned} h_0 &= \sigma(W^0 e_i^j + b^0), \\ h_1 &= \sigma(W^1 h_0 + b^1), \\ a^j &= \sigma(W^2 h_1 + b^2), \\ g_i^j &= a_i^j \times e_i^j, \end{aligned} \tag{1}$$

where a_i^j is a scalar that weights the importance of embeddings e_i^j.

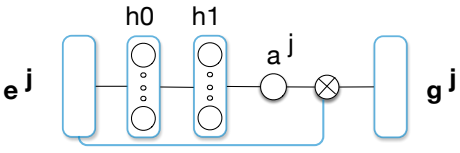

Figure 3: An attention unit for feature embeddings e^j.

We then concatenate all the weighted embeddings to form the output $x_i^g = g_i^1 \oplus g_i^2 \oplus \cdots \oplus g_i^n$.

NN layers.

The NN layers are used to extract abstract features from the above gated representations. We adopt Deep & Cross network [Wang *et al.*, 2017] to efficiently learn feature interactions. Specifically, we process the above output x_i^g as follows:

$$\begin{aligned} x_i^{g1} &= x_i^g (x_i^g)^\top W_1^g + x_i^g + b_1^g, \\ x_i^{g2} &= x_i^g (x_i^{g1})^\top W_2^g + x_i^{g1} + b_2^g, \\ h_i^{g2} &= \text{FCN}(x_i^g) \\ o_i^g &= h_i^{g2} \oplus x_i^{g2}, \end{aligned} \tag{2}$$

where x_i^{g2} and h_i^{g2} are two-layer cross network output and two-layer Fully-Connected Network (FCN) output respectively, W^g and b^g are weights and biases.

2.2 Joint Training

Our joint model considers a joint training setting with two domains involved. We leave the exploration of more domains to our future work. Similar to [Liu *et al.*, 2017b], we use a shared network and two domain-specific networks to derive shared features O^c and domain-specific features O^0 and O^1 respectively. Note that, for the shared attention layer in shared network, the parameters $W = W^c$ are shared with both domains. For the domain-specific attention layer in domain-specific networks, the parameters $W = W^d$ ($d \in [0, 1]$) are specific to different domains. The predictions $\hat{y}^0$ and $\hat{y}^0$ are defined as:

$$\begin{aligned} \hat{y}^0 &= \sigma(W_c^0 O^c + W^0 O^0 + b^0), \\ \hat{y}^1 &= \sigma(W_c^1 O^c + W^1 O^1 + b^1), \end{aligned}$$

where W_c^1, W_c^0, W^0, and W^1 are the weights for two domains, b^0 and b^1 are the biases for two domains.

We can then define the domain-specific loss $Loss_d$ as follows::

$$Loss_d = -\frac{1}{n_d} \sum_{j=1}^{n_d} \frac{1}{2} [y_j^d \log \hat{y}_j^d + (1 - y_j^d) \log(1 - \hat{y}_j^d)].$$

2.3 Adversarial Loss

Note that the above method does not force the shared features to be domain-invariant. It's possible that some domain-specific features are mixed with the shared features as the model itself doesn't prevent this. Recent studies [Ganin *et al.*, 2016; Taigman *et al.*, 2017; Liu *et al.*, 2017b; Chen *et al.*, 2018; Shen *et al.*, 2018] consider to apply adversarial loss on shared features to prevent domain-specific features from creeping into shared feature space. Following [Liu *et al.*, 2017b; Chen *et al.*, 2018], we use an adversarial loss L_{adv} on the shared hidden representations to encourage the shared features learned to be indiscriminate across two domains:

$$L_{adv} = \frac{1}{n} \sum_{i=1}^{n} \sum_{d=0}^{1} p(d|O^c) \log p(d|O^c).$$

where d_i is the domain label and $p(d_i|\cdot)$ is the domain probability from a domain discriminator, defined below.

The main idea of domain discriminator is to predict the domain label d on the hidden representations. Let D be the domain discriminator which is a fully connected layer with weights W^c and bias vector b^c, we have:

$$D(d \mid O^c) = \text{softmax}(W^c O^c + b^c),$$

Hence we have $p(d_i|O^c) = D(d = d_i \mid O^c)$.

Furthermore, to encourage the specific feature space to be discriminate between different domains, we consider applying domain discrimination losses on the two specific feature spaces [Chen *et al.*, 2018]. We define L_0 and L_1 for domain 0 and domain 1 as follows:

$$L_0 = -\frac{1}{n_0}\sum_{i=1}^{n_0} \log p(d_i = 0|O^0).$$

$$L_1 = -\frac{1}{n_1}\sum_{i=1}^{n_1} \log p(d_i = 1|O^1).$$

where O^0 and O^1 are hidden representations for domain 0 and domain 1 respectively.

In all, we obtain a combined loss for our model:

$$\mathcal{L} = \sum_{d=0}^{1} Loss_d + \frac{\lambda_1}{2} L_{adv} + \frac{\lambda_2}{2} L_0 + \frac{\lambda_3}{2} L_1 + \frac{\lambda_4}{2} ||\Theta||_F^2.$$

where Θ denotes model parameters. Note that the first term is the classification loss (cross entropy loss), the second is adversarial loss, the third and fourth are discrimination losses, and the last is the regularization. To minimize the above the objective, our model can help to make the shared features learned in our model to be more domain-invariant, and the specific features learned to be more domain-specific.

3 Experiments

In this section, we conduct extensive experiments to examine the following questions:

- Does our base model performs well without considering transfer learning? (Q1)
- Does our proposed method help more comparing with other transfer learning methods? (Q2)
- Are the domain-specific attention weights learned by our model insightful? (Q3)

We will answer these questions after presenting our datasets and some fundamental experimental settings.

Datasets: We collected one-week user click-through data from three large E-commerce websites [3]. The data sizes are 56,000M, 4,900M and 350M for platform A, B and C respectively. Clearly for platform A, it has sufficient data to train a model, while for platform B and C, it may not have enough data to train a good model.

Features: We have continuous and categorical features, where for all the continuous features, we perform feature discretization to map them to categorical features. Here we mainly consider these types of features:

statistic - the set of statistic features w.r.t. items, e.g.: item historical sales, exposure information, etc.;

profile - item and user profiles, e.g.: item price, item title, user age, user education information etc.;

personal behavior - the set of features related to user behaviors such as click, purchase etc.

Labels: We treat user click or purchase behaviors as positive labels (y=1), and others as negative labels (y=0).

Settings: We set the shared feature embedding size as 64, the specific feature embedding size as 8. The attention layer is with a structure of $64 \times 32 \times 1$ for each domain. For the deep and cross network, the deep network is 512×256, the cross network is with two layers. We use relu as the activation functions for the NN layers and Adam as the optimizer. The learning rate is set as 0.001. All the methods in this paper are implemented with TensorFlow [4] and are trained with NVIDIA Tesla K40M GPUs. Our model is distributed trained in a cluster of 100 GPUs.

3.1 Comparison on base models (Q1)

We first evaluate our model by using different NN layers. We consider these variants:

- Deep-only: a classic two-layer fully-connected network with a structure of 512×256;
- Wide & Deep [Cheng, 2016]: an extension of the Deep-only model with a Wide layer;
- DeepFM [Guo *et al.*, 2017]: an extension of the Deep-only model with a FM layer;
- Deep & Cross [Wang *et al.*, 2017]: an extension of the Deep-only model with a cross layer, this is a degenerated version of our model without consider joint training.

Note that all these models have the same setting of shared and specific feature embeddings, and the attention layers.

As shown in Table 1, the observations are as follows. First, we find the platform A itself is able to train a good model as it has the higher AUC than other two platforms. Second, Wide & Deep model outperforms the Deep-only model which shows the necessity of consider wide features. Third, DeepFM shows better performance than Wide & Deep in platform B and C. For platform A, it has similar performance with Wide & Deep. This shows for domains with fewer data, the DeepFM helps more. Last but not least, our Deep & Cross setting has the best performance comparing to the baseline models. This shows the cross network is effective in learning feature interactions.

In this study, we use Deep & Cross network as our base NN layer, more NN methods such as [He and Chua, 2017; Xiao *et al.*, 2017] can also be used here. We leave the exploration of more effective NN methods as our future work.

3.2 Model performance on cross-domain learning (Q2)

We consider two scenarios of joint training: A & B, and A & C, the former refers to joint train a model for platform A

[3] Taobao, TMall and Qintao respectively.

[4] www.tensorflow.org

Table 1: Comparison on the base models that use different neural networks for learning abstract features.

AUC	A	B	C
Deep-only	0.7383	0.6833	0.6523
Wide & Deep	0.7479	0.6935	0.6689
DeepFM	0.7478	0.6998	0.6711
Deep & Cross	**0.7488**	**0.7183**	**0.6734**

and B, the latter for A and C. To evaluate the effectiveness of our cross-domain learning in these scenarios, we compare our full model with these baselines.

- **Src-only.** A model that uses only source data (from platform A). This is to examine whether platform A is helpful for other domains;
- **Self-train.** A model that uses its own data for training;
- **FineTune.** A typical TL method that trains on a source domain and finetune the networks in the target domain.
- **DANN.** The domain-adversarial neural network (DANN) in [Ganin *et al.*, 2016]. It's a fully-shared model with both source and target data with a gradient reversal layer as adversarial training. For fair comparison, we adopt Deep & Cross network in DANN.
- **CNAN**$_1$. A degenerated version of our model without using domain-specific embeddings.
- **CNAN**$_2$. A degenerated version of our model without using attention layer.
- **CNAN-AT.** This is our full model.

Table 2: Comparison with different cross-domain models. Domain A is jointly train with domain B and C respectively.

AUC	A & B		A & C	
	A	B	A	C
Src-only (A)	0.7488	0.6892	0.7456	0.6621
Self-train	0.7488	0.7183	0.7456	0.6734
FineTune	0.7435	0.7382	0.7398	0.6869
DANN	0.7489	0.7356	0.7459	0.6873
CNAN$_1$	0.7472	0.7368	0.7442	0.6910
CNAN$_2$	0.7445	0.7340	0.7385	0.6866
CNAN-AT	**0.7501**	**0.7414**	**0.7489**	**0.7011**

Table 3: Comparison with different cross-domain models. Domain A is jointly train with domain B and C respectively.

AUC	Scenario A		Scenario B	
	Taobao	Tmall	Taobao	Qintao
Src-only (A)	0.7488	0.6892	0.7456	0.6621
Self-train	0.7488	0.7183	0.7456	0.6734
FineTune	0.7435	0.7382	0.7398	0.6869
DANN	0.7489	0.7356	0.7459	0.6873
CNAN$_1$	0.7472	0.7368	0.7442	0.6910
CNAN$_2$	0.7445	0.7340	0.7385	0.6866
CNAN-AT	**0.7501**	**0.7414**	**0.7489**	**0.7011**

As in Table 3, a few important observations are as follows. First, Src-only performs worse than Self-train which means simply use data from platform A for B or C is not good. This shows platform A is close to but different from platform B or C, i.e. there is domain shift between A and B/C; Second, the FineTune method does help the target domains as an improvement is observed in domain B and C comparing with Self-train. However, it may not has a good performance in the source domain, e.g. the FineTune method has degenerated performance in A & B for platform A (0.7435 vs. 0.7488), similarly in A & C. This is reasonable as the model is fine-tuned by the target data at a later stage which may not be ideal for source domain; Third, despite the domain shift, DANN is able to leverage knowledge from the source domain and boost performance; Furthermore, the fact that our CNAN outperform the degenerated model CNAN$_1$ shows the importance of domain-specific embeddings; and our CNAN outperform CNAN$_2$ shows the importance of attention layers; Last, our model shows better performance than DANN and other baselines, this shows the advantage of our proposed framework.

Meanwhile, our model improves more for B and C comparing to A. This is reasonable as platform A has the largest data set where using its own data is quite good. But still our method can improves A which shows the benefits of joint training. We also find our model improve more on the setting of 'A & C' than 'A & B', a potential reason is that domain C has less data than B, thus benefit more from the source domain. To examine this, we choose different percentage of target data to evaluate our model performance.

We further evaluate our model performance when more target data are given. As shown in Table 4, with the growing amount of target data utilized in training (10% to 100%), our model has a smaller improvement over the Self-train model (6.9% to 4.1%). This shows our model helps more for target domains with less sufficient data. This is possibly because that when target domain has sufficient data, joint training may not be that helpful.

Table 4: Model performance w.r.t. different target data size.

% Data	10%	40%	70%	100%
Self-train	0.6014	0.6224	0.6678	0.6734
CNAN-AT	0.6432	0.6642	0.6988	0.7011
Improvement	6.9%	6.7%	4.6%	4.1%

3.3 Qualitative evaluation (Q3)

We choose five types of our features to illustrate the attention weights learned.

- item statistics: the set of statistic features of the item, e.g.: N days sales (N=1,3,7,15,30).
- query+item statistics: the set of query-item interaction statistic features,e.g.: N days sales of item specific to a certain query (N=1,3,7,15,30).
- item price: price information.
- item brand: brand information.
- personal: the set of features related to user behavior, e.g: user click behavior.

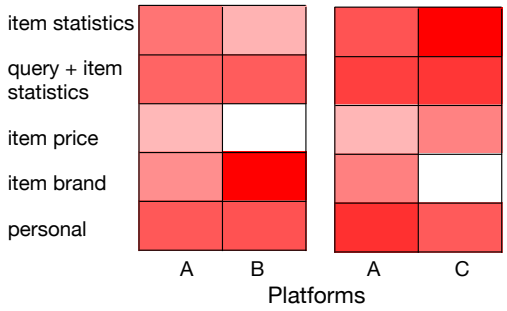

Figure 4: Visualization of attention weights learned by our model at different domains. The redder the cell is, the higher the attention weight is. The attention weights on the left are learned from joint training of platform A and B, the right are from platform A and C.

We visualize the attention weights of these types of features w.r.t different domains (platforms) in Fig. 4. Note that for each type of features, we average all the related feature attention weights. A few important observations are as follows. First, we find that the same features have different attention weights in different domains. This shows that 'domain shift' exist in the datasets. Second, since platform B and C target different user groups, we find the attention weights are significant different. For the high-end platform B, item brand and penalization features play more important role than item price. While for the low-end platform C, the item price is more important than item brand. Last, since the platform A targets a wide range of users, which sits at the middle of B and C, the attention weights of different features are more equally distributed.

In a nutshell, the attention weights learned by our model are very insightful and can give us a glimpse of user needs at different platforms.

4 Related Work

Item Recommendation Item recommendation is the core task for E-commerce search. This task is close to click-through-rate (CTR) prediction task as the core problem is to evaluate the click or purchase probability of an item for a given user query. The task is a large-scale problem in industry as usually millions of items and features are used. Due to the large feature space and data size, a number of methods are used to avoid extensive feature enginneering, typical methods include: FM [Rendle, 2010], Wide & Deep [Cheng, 2016], DeepFM [Guo *et al.*, 2017], and Deep & Cross [Wang *et al.*, 2017]. The FM based methods [Rendle, 2010] embed sparse features with low-dimensional dense vectors and learns feature interactions based on the vectors. Still these models are considered as shallow and may have large computational cost by extending to higher orders [Blondel *et al.*, 2016]. With the popularity of deep neural networks (DNN), there are growing number of studies working on using DNN to learn high-degree feature interactions [Guo *et al.*, 2017; Cheng, 2016; Wang *et al.*, 2017]. The Wide & Deep model from Google [Cheng, 2016] combines the cross features in wide part with the abstract features from deep part to improve the deep-only and wide-only models. To avoid the feature crossing in the wide part, DeepFM [Guo *et al.*, 2017] uses the power of factorization machines in the 'wide' part. A recent study in [Wang *et al.*, 2017] shows a cross network can help to learn high-degree feature interactions. By combining the cross network with the deep network, the resulting Deep & Cross method shows to have better performance than FM, Deep-only, and Wide & Deep method. Hence we adopt this method as our hidden representation layer. Note that this can be replaced with other models. Different from the original Deep & Cross network, our method jointly learns domain-specific and shared embeddings together with attention layers to learn domain-specific feature representations.

Joint Training. We study cross-domain learning in our task, which is close to transfer learning [Pan and Yang, 2010] or multi-task learning [Zhang and Yang, 2017]). There are generally two lines of studies for transfer learning. The first line of work is supervised domain adaptation, which assumes to have enough labeled data from a source domain and also a little labeled data from a target domain [Daume III, 2007]. And the latter assumes to only have labeled data from source domain but may also have some unlabeled data from a target domain [Blitzer *et al.*, 2006; Yu and Jiang, 2016].

Our study is close to the first line of work. For studies in this line, a majority of recent studies are to find a shared feature space which can reduce the divergence between the distribution of the source and the target domains [Argyriou *et al.*, 2007; Wang and Mahadevan, 2008]. In our study, we mainly use deep neural network (DNN) models to learn shared feature representations, as DNN models are shown to be effective for learning transferable features in many NLP tasks [Yosinski *et al.*, 2014; Huang *et al.*, 2017b; Huang *et al.*, 2017a].

A simple but widely used framework is referred to as *fine-tuning* approaches, which first use the parameters of the well-trained models on the source domain to initialize the model parameters of the target domain, and then fine-tune the parameters based on labeled data in the target domain [Yosinski *et al.*, 2014; Mou *et al.*, 2016]. A more effective method is to use a shared NN to learn shared features for both source and target domains [Mou *et al.*, 2016; Yang *et al.*, 2017]. Another approach is to use both a shared NN and domain-specific NNs to derive shared and domain-specific features [Liu *et al.*, 2017b]. Some recent studies [Ganin *et al.*, 2016; Taigman *et al.*, 2017; Liu *et al.*, 2017b] consider an adversarial networks to learn more robust shared features across domains. Inspired by this, our model is built on a we extend our multi-domain attention network with a specific shared model with adversarial training. A major difference between our model and transfer learning model is that we seek to improve both source and target domains.

To the best of our knowledge, our work is the first to study multi-domain attention network for cross-platform e-commerce search. Experiments show that our model achieves significantly better results than the existing approaches.

5 Conclusion

In this work, we studied transfer learning with domain-aware attention network for item recommendation in e-commerce.

We also proposed a new adversarial regularizer to learn domain invariant and domain-specific features. Experiments show that our model outperforms the competing methods by a large margin. In the future, we seek to extend our model to multiple tasks besides CTR prediction, e.g. user purchase prediction, add to cart prediction etc.

References

[Argyriou *et al.*, 2007] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[Blitzer *et al.*, 2006] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

[Blondel *et al.*, 2016] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3351–3359. Curran Associates, Inc., 2016.

[Chen *et al.*, 2018] Cen Chen, Yinfei Yang, Jun Zhou, Forrest Sheng Bao, and Xiaolong Li. Cross-domain review helpfulness prediction based on convolutional neural networks with auxiliary domain discriminators. In *NAACL18*, 2018.

[Cheng, 2016] et al Cheng, Heng-Tze. Wide & deep learning for recommender systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2016.

[Daume III, 2007] Hal Daume III. Frustratingly easy domain adaptation. In *ACL*, 2007.

[Ganin *et al.*, 2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. *CoRR*, abs/1703.04247, 2017.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 355–364, 2017.

[Huang *et al.*, 2017a] Xin Huang, Yuxin Peng, and Mingkuan Yuan. Cross-modal common representation learning by hybrid transfer network. *CoRR*, abs/1706.00153, 2017.

[Huang *et al.*, 2017b] Xin Huang, Yuxin Peng, and Mingkuan Yuan. MHTN: modal-adversarial hybrid transfer network for cross-modal retrieval. *CoRR*, abs/1708.04308, 2017.

[Liu *et al.*, 2017a] Haijing Liu, Yang Gao, Pin Lv, Mengxue Li, Shiqiang Geng, Minglan Li, and Hao Wang. Using argument-based features to predict and analyse review helpfulness. In *EMNLP*, pages 1358–1363, Copenhagen, Denmark, September 2017.

[Liu *et al.*, 2017b] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. In *ACL*, 2017.

[Mou *et al.*, 2016] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? In *EMNLP*, 2016.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, 2010.

[Shen *et al.*, 2018] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *AAAI*. AAAI Press, 2018.

[Taigman *et al.*, 2017] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *ICLR*, 2017.

[Wang and Mahadevan, 2008] Chang Wang and Sridhar Mahadevan. Manifold alignment using procrustes analysis. In *ICML*, 2008.

[Wang *et al.*, 2017] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 12:1–12:7, New York, NY, USA, 2017. ACM.

[Xiao *et al.*, 2017] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *CoRR*, abs/1708.04617, 2017.

[Yang *et al.*, 2017] Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. Transfer learning for sequence tagging with hierarchical recurrent networks. *ICLR*, 2017.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[Yu and Jiang, 2016] Jianfei Yu and Jing Jiang. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP*, 2016.

[Zhang and Yang, 2017] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv:1707.08114*, 2017.