A Minimax Game for Instance based Selective Transfer Learning

Bo Wang[1], Minghui Qiu[1,2,*], Xisen Wang[1], Yaliang Li[1], Yu Gong[1], Xiaoyi Zeng[1], Jun Huang[1], Bo Zheng[1], Deng Cai[2], and Jingren Zhou[1]
[1] Alibaba Group, China [2] Zhejiang University, China
{boyi.wb,minghui.qmh,xisen.wxs,yaliang.li,gongyu.gy}@alibaba-inc.com
{yuanhan,huangjun.hj,bozheng,jingren.zhou}@alibaba-inc.com
dengcai@cad.zju.edu.cn

ABSTRACT

Deep neural network based transfer learning has been widely used to leverage information from the domain with rich data to help domain with insufficient data. When the source data distribution is different from the target data, transferring knowledge between these domains may lead to negative transfer. To mitigate this problem, a typical way is to select useful source domain data for transferring. However, limited studies focus on selecting high-quality source data to help neural network based transfer learning. To bridge this gap, we propose a general Minimax Game based model for selective Transfer Learning (MGTL). More specifically, we build a selector, a discriminator and a TL module in the proposed method. The discriminator aims to maximize the differences between selected source data and target data, while the selector acts as an attacker to selected source data that are close to the target to minimize the differences. The TL module trains on the selected data and provides rewards to guide the selector. Those three modules play a minimax game to help select useful source data for transferring. Our method is also shown to speed up the training process of the learning task in the target domain than traditional TL methods. To the best of our knowledge, this is the first to build a minimax game based model for selective transfer learning. To examine the generality of our method, we evaluate it on two different tasks: item recommendation and text retrieval. Extensive experiments over both public and real-world datasets demonstrate that our model outperforms the competing methods by a large margin. Meanwhile, the quantitative evaluation shows our model can select data which are close to target data. Our model is also deployed in a real-world system and significant improvement over the baselines is observed.

KEYWORDS

Instance-based Transfer Learning, GAN, Reinforcement Learning

ACM Reference Format:
Bo Wang, Minghui Qiu, Xisen Wang, Yaliang Li, Yu Gong, Xiaoyi Zeng, Jun Huang, Bo Zheng, Deng Cai, and Jingren Zhou. 2019. A Minimax Game for Instance based Selective Transfer Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330841

* Minghui Qiu is the corresponding author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330841

1 INTRODUCTION

Transfer Learning [24] has been widely adopted in various applications to leverage information from data-rich domains to help data-insufficient domains. Due to the representation learning ability of deep neural networks, neural architectures based transfer learning methods have gained increasing popularity recently, and they have shown to be effective for several applications [2, 20, 23, 31, 43, 46]. However, when the distribution of the source domain is largely different with the target domain, transferring knowledge between two domains should be carefully examined, otherwise negative transfer [28] can lead to unexpected results.

One typical way to alleviate negative transfer is to select a part of the source data for transferring, which aims to make source domain distribution close to the target domain. Although various data selection methods [17, 29] have been proposed, they may not be ideal for deep neural network based transfer learning. The main reason is that these data selection methods treat the transfer learning task as a sub-module that can continuously provide sufficient updates for the instance selection, which requires repetitive training [27]. Thus, most of these existing data selection methods can be time-consuming for deep neural network based transfer learning. Therefore, there is a great need to design an efficient instance based transfer learning method which can not only leverage the great representation learning ability of deep neural network but also introduce low computation cost when selecting instances for transferring. In the light of this need, in this paper we propose a general **M**inimax **G**ame based method for selective **T**ransfer **L**earning (MGTL). The proposed method consists of three components: a data selector, a discriminator, and a transfer learning module. The data selector selects high-quality source data to train the transfer learning module, and in turn, transfer learning module and the discriminator together play a game to guide the data selector. These three components are jointly trained in a batch fashion to improve the efficiency.

To be more specific, the discriminator in the proposed MGTL method aims to separate the selected source data from the target data, i.e., maximizing the difference between the selected source data and the target data. On the other hand, the data selector acts as an attacker to the discriminator by selecting instances close to the target data distribution to minimize the differences between the selected source data and target data. Meanwhile, the transfer learning module examines the benefits of incorporating the selected

source data and provide "reward" to the data selector. In this way, the interactions among the data selector, the discriminator and the transfer learning module play a minimax game to choose a set of source data that are close to the target domain for performance improvement.

To demonstrate the generality of the proposed MGTL method, we evaluate it on two different tasks: item recommendation and text retrieval [1]. Experiments on both public datasets and large-scale real-world datasets show that the proposed MGTL method outperforms the competing methods by a large margin, while the efficiency of MGTL is guaranteed. Qualitative evaluations show that the instances selected by MGTL are reasonable and insightful. Furthermore, we apply the proposed method to a real-world system, and the promising results confirm its usefulness.

In summary, the main contributions of this paper are:

- We propose MGTL, a Minimax Game based selective Transfer Learning method, to select high-quality source data to improve the transferring ability between different domains. To the best of our knowledge, this is the first study to consider minimax game for selective transfer learning.
- In contrast to selecting data instance by instance, the actions in the proposed MGTL method are sampled in batches to improve the training efficiency.
- Extensive experiments on two different tasks with both public datasets and large-scale real-world datasets show the effectiveness and efficiency of the proposed MGTL method.

2 MODEL

In this section, we present our Minimax Game based Transfer Learning method (MGTL) in details.

2.1 Task definition

Our model is designed to address the following supervised TL problem. We assume to have a set of labeled data instances from a source domain $\{x_i^s\}_{i=1}^{n_s}$, a set of labeled instances from a target domain $\{x_i^t\}_{i=1}^{n_t}$ [2]. We seek to build a model to transfer knowledge from the source domain to help a target domain. Due to the domain shift, naively transferring from the source domain to the target domain may not be ideal. We urge that data selection is necessary to prevent such negative transfer problem. The data selection process is expected to find a subset of data instances from the whole source data to train a new TL module $\mathcal{M}$ to perform better in the target domain. Note that our model is general for supervised TL tasks. The underlying TL module can be any DNN based TL methods.

2.2 Overview

We present an overview of our method in Fig. 1, our proposed framework consists of three components: the selector $\mathcal{S}$ would try to select data that is similar to the target domain distribution from the source domain and therefore could fool the discriminator $\mathcal{D}$, the discriminator would try to draw a clear distinction between the target domain data and the selected ones made by its opponent selector. The discriminator examines each selected data instances and gives an immediate reward for each data instance to guide the selector. Meanwhile, the selected data is used to update the TL module $\mathcal{M}$. The performance of $\mathcal{M}$ on a held-out validation set $\mathcal{V}$ can give another delayed reward to guide the selector.

Let θ, ϕ, φ be the parameters correspond to the selector $\mathcal{S}$, the discriminator $\mathcal{D}$, and the TL module $\mathcal{M}$ respectively. Formally, we define the objective as follows:

$$J = J_1(\text{minimax game objective}) + J_2(\text{TL module objective}), \quad (1)$$

$$J_1 = \min_{\theta} \Big\{ \max_{\phi} [\mathbb{E}_{d_i \sim p_{tgt}(d)} \log \mathcal{D}(d_i) + \mathbb{E}_{d_i \sim p_\theta(d)} \log(1 - \mathcal{D}(d_i))] + \frac{1}{|\mathcal{V}|} \underbrace{\sum_{i=1}^{|\mathcal{V}|} [L'(y_i, \mathcal{M}_{p_\theta}(d_i)) - L'(y_i, \mathcal{M}(d_i))]}_{\text{Reward}} \Big\} \quad , \quad (2)$$

$$J_2 = \min_{\varphi} \left\{ \mathbb{E}_{d_i \sim p_{tgt}(d) \cup p_\theta(d)} L(y_i, \mathcal{M}(d_i)) \right\} \quad , \quad (3)$$

where $p_\theta(d)$ refers to the selected source instances by the selector, and $p_{tgt}(d)$ refers to the target data instances. $\mathcal{D}(d)$ measures the probability of a given instance d being relevant to the target domain, which is discussed in the next section. The reward term is based on the TL module $\mathcal{M}$ performance on the held-out validation set $\mathcal{V}$, where we use the loss difference L' between the previous module $\mathcal{M}$ and the current updated module $\mathcal{M}_{p_\theta}$ as reward. Note that here we refer to TL module $\mathcal{M}_{p_\theta}$ as the updated module based on the selected source instances based on p_θ. The last term J_2 correspond to the TL module loss, where it seeks to minimize the prediction loss $L(y_i, \mathcal{M}(d_i))$, on both the selected source instances and target data instances.

Empirically, we find it is beneficial to consider shared feature embeddings between the discriminator and TL module as illustrated in Fig. 1. On the one hand, this helps to reduce module size as the majority of parameters are in the feature embedding layer. On the other hand, we hope the projection of data in the embedding space is both discriminative from the discriminator point of view and helpful for the TL task. Below we describe the modules in details.

2.3 Discriminator

The objective for the discriminator $\mathcal{D}$ is to maximize the log-likelihood of correctly distinguishing the target and selected source instances. It can be modeled as a binary domain classification problem, where label 0 refers to the data instance is from the source domain, and label 1 from the target domain. We define the discriminator as a sigmoid function of the discriminative score as:

$$\mathcal{D}(d) = \sigma(f_\phi(d)) = \frac{\exp(f_\phi(d))}{1 + \exp(f_\phi(d))}. \quad (4)$$

With the target instances, and the source instances sampled from the current selector $p_\theta(d)$, we can optimize the discriminator as:

$$\phi^* = \arg\max_{\phi} [\mathbb{E}_{d_i \sim p_{tgt}(d)} \log \mathcal{D}(d_i) + \mathbb{E}_{d_i \sim p_\theta(d)} \log(1 - \mathcal{D}(d_i))], \quad (5)$$

where if the function f_ϕ is differentiable with respect to ϕ, the above can be solved by stochastic gradient descent.

[1] The core task of text retrieval is text matching, hence we use these two words "text retrieval" and "text matching" interchangeably in this paper.

[2] typically the source domain data size is larger than the target domain, i.e. $n_s >> n_t$.

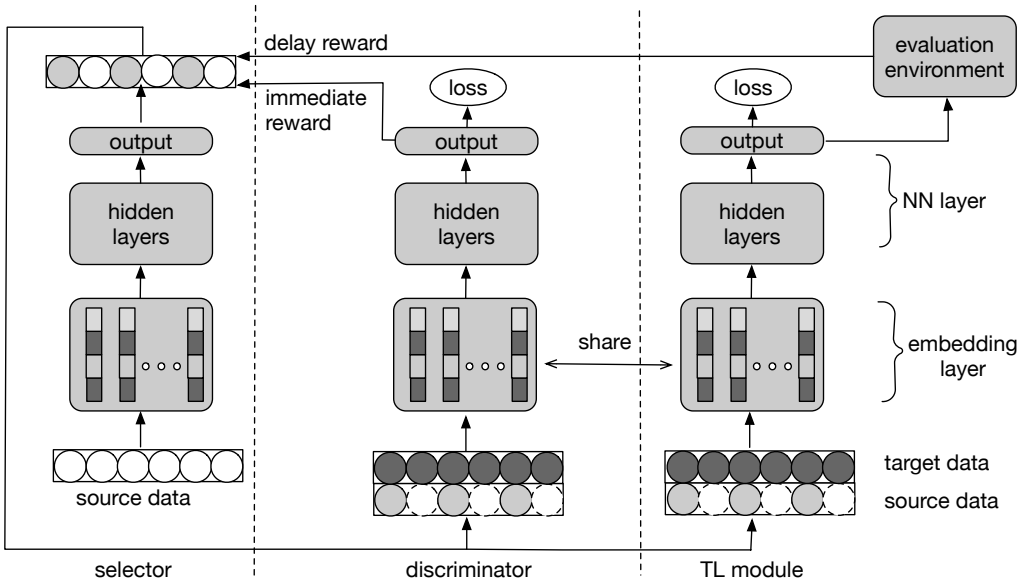

Figure 1: A Minimax Game based selective Transfer Learning method (MGTL). The discriminator provide immediate rewards to guide the selector and the TL module evaluates on a validation data (a.k.a. evaluation environment) and provides additional delayed reward to help the selector.

2.4 Data selector

The selector $p_\theta(d)$ intends to optimize two objectives: (i) to fit the target domain distribution $p_{tgt}(d)$ by selected different set of source instances, (ii) to improve the TL module on the target task. The former is to find source instances close to target domain to fool the discriminator, and the latter is to find source instances that can be used to update the TL module to improve the performance on a validation dataset. Specifically, while keeping the discriminator $f_\phi(d)$ fixed after its maximization in Eq. 5, we use the following formula to update θ for the selector.

$$\begin{aligned}\theta^* &= \arg\min_\theta \Big\{[\mathbb{E}_{d_i\sim p_{tgt}(d)} \log \mathcal{D}(d_i) + \mathbb{E}_{d_i\sim p_\theta(d)} \log(1-\mathcal{D}(d_i))] \\ &\quad + \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}[L'(y_i, \mathcal{M}_{p_\theta}(d_i)) - L'(y_i, \mathcal{M}(d_i))]\Big\} \\ &= \arg\max_\theta \Big\{ \underbrace{\mathbb{E}_{d_i\sim p_\theta(d)} - \log(1-\mathcal{D}(d_i)))}_{\text{Immediate Reward}} \quad + \\ &\quad \underbrace{\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}[L'(y_i, \mathcal{M}(d_i)) - L'(y_i, \mathcal{M}_{p_\theta}(d_i))]}_{\text{Delayed Reward}} \Big\}. \end{aligned} \tag{6}$$

Note that, the sampling of source instances is discrete which cannot be directly optimized by stochastic gradient descent based approach. We thus optimize the selector using reinforcement learning [41, 45], where we use the discriminator scores and TL module performance differences as "rewards" to update the selector. The former can be treated as a type of immediate reward and the latter provides a delayed reward to the selector, detailed as follows.

2.4.1 Immediate reward. As illustrated in Fig. 2, the selected data are fed into the discriminator to examine their discriminativeness

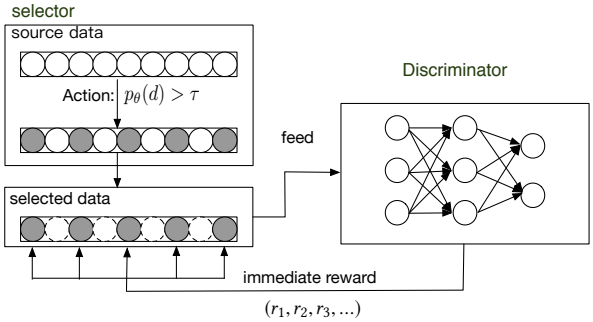

Figure 2: An illustration of immediate reward.

scores which are served as rewards to guide the selection process. From Eq. 6, we can update θ according to the immediate rewards by keeping the discriminator $f_\phi(d)$ fixed after its maximization in Eq. 5. Let d_i be the i-th data instance sampled based on the current selector $p_\theta(d)$, N and K be the data size and selected data size (usually $K << N$), we update the parameters θ as follows:

$$\begin{aligned}\theta^* &= \arg\max_\theta \mathbb{E}_{d_i\sim p_\theta(d)} - \log(1-\mathcal{D}(d_i)), \\ &= \arg\max_\theta \frac{1}{N}\sum_{i=1}^{N} p_\theta(d_i)\log(1+\exp(f_\phi(d_i))), \\ &= \theta + \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta p_\theta(d_i)\log(1+\exp(f_\phi(d_i))), \\ &= \theta + \mathbb{E}_{d\sim p_\theta(d)}[\nabla_\theta \log p_\theta(d_i)\log(1+\exp(f_\phi(d_i)))], \\ &\simeq \theta + \frac{1}{K}\sum_{i=1}^{K}[\nabla_\theta \underbrace{\log p_\theta(d_i)}_{\text{Action}} \underbrace{\log(1+\exp(f_\phi(d_i)))}_{\text{Reward}}]. \end{aligned} \tag{7}$$

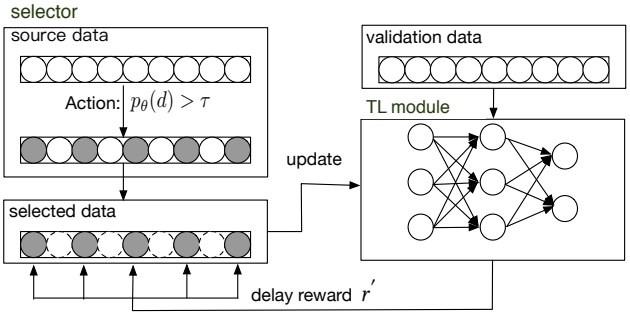

Figure 3: An illustration of delayed reward from TL module.

This is a typical policy gradient method setting [33, 41]. The term $\log(1 + \exp(f_\phi(d_i)))$ serves as the immediate reward r_i for policy $p_\theta(d)$ taking an action d_i in the source environment. Note that, in theory, we need to sample data instances from all the source data which is very time-consuming. To speed up the training, we update the module in batches, where we selected a subset of source instance from a batch B at a time. The subset is selected based on a predefined threshold τ:

$$\text{Selected subset} = \{d_i | p_\theta(d_i) > \tau\}_{i=1}^{B}. \tag{8}$$

2.4.2 Delayed reward. Furthermore, we consider additional reward to update the selector. Recall that we seek to find high quality source data to improve the TL module. The scenario is illustrated in Fig 3, the selector takes actions $p_\theta(d) > \tau$ to select data from source domain and form a new batch of source data χ_b. The τ is a parameter need to be tuned. We use χ_b to update the TL module. Then we get the new evaluation results $\frac{1}{\mathcal{V}} \sum_i^{\mathcal{V}} L'(y_i, \mathcal{M}_{p_\theta}(d_i))$ using validate data $\mathcal{V}$. As in Eq. 6, the delayed reward r_b is defined as:

$$r_b = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{\mathcal{V}} [L'(y_i, \mathcal{M}(d_i)) - L'(y_i, \mathcal{M}_{p_\theta}(d_i))], \tag{9}$$

where $L'(y_i, \mathcal{M}(d_i))$ is the previous evaluation results which can be stored beforehand, and $L'(y_i, \mathcal{M}_{p_\theta}(d_i))$ is the evaluation results after module updates.

In contrast to conventional reinforcement learning, where one action is sampled based on one state and obtaining one reward from the environment, our actions are sampled in batch and obtaining one reward for each batch. The delayed reward is set to evaluate whether the sampled batch is helpful to the TL module, that is, whether the sampled data provide positive updates to the TL module to improve its performance on the target task. Here we use a validation set to mimic the target task. To evaluate the TL module, we can use differentiable evaluation metrics such as Logloss or Accuracy, or non-differentiable metrics like Area under the ROC curve (AUC). In our experiment, we tested all these three metrics: Logloss, ACC, and AUC, and find the performances are close. For simplicity, we use Logloss as our final reward term, i.e. $r_b = \text{Logloss} - \text{Logloss}_{p_\theta}$, where $\text{Logloss}_{p_\theta}$ refers to Logloss after updated the module using data from p_θ. Formally, for each batch in an episode, the accumulated reward is defined as:

$$r'_b = r_b + \gamma * r_{b+1} + \gamma^2 * r_{b+2} + \cdots + \gamma^{n-b} * r_n. \tag{10}$$

where r_n is the last batch reward and γ is the discount factor.

Finally, we use REINFORCE algorithm [41] to update the selector, combined both the above delayed reward and immediate reward in Eq. 7, we have the following formula:

$$\theta \leftarrow \theta + \alpha \frac{1}{K} \sum_{i=1}^{K} \nabla_\theta \log p_\theta(d_i) R_i \tag{11}$$

$$R_i = \underbrace{\lambda \log(1 + \exp(f_\phi(d_i)))}_{\text{Immediate Reward}} + \underbrace{(1-\lambda) r'_b}_{\text{Delayed Reward}}. \tag{12}$$

The above update formula is quite intuitive: an increase in $D(d_i)$, i.e. an instance that is closer to the target domain, will lead to an increase of immediate reward as $\log(1 + \exp(f_\phi(d_i))) = -\log(1 - D(d_i))$. And the increased reward will promote $p_\theta(d_i)$ which makes the selector tends to have higher probability to choose d_i. Meanwhile, an increase of delayed reward r_b will increase $p_\theta(d_i)$ as well, which means a higher gain of training TL module in the validation set will enhance the current selection behaviors.

In RL terminology, the source data instances serve as states S, the selection process on the source data instances are actions A, the immediate and delayed reward serve as rewards R. Given tuples of (S, A, R), we can update the RL method.

It is worth mentioning that unlike Generative Adversarial Networks [13, 18] where the generator tries to generate features or new samples to fool the discriminator, our selector, which is close to the generator, tries to directly "generate" known samples from the source domain. And different from the existing TL methods that learn shared feature space for both source and target domain [12, 22, 23, 34], our method seeks to select distribution similar data to improve the TL module in the target domain.

2.5 TL module

Let $\{d_i\}_{i=1}^{N'}$ be the selected source data and the full target data. As shown in Eq. 3, the objective for the TL module is to minimize the cross-entropy loss (in the case of classification problems) of $\{d_i\}_{i=1}^{N'}$ with regard to their labels $\{y_i\}_{i=1}^{N'}$.

Let $J = \frac{1}{N'} \sum_{i=1}^{N'} L(y_i, \mathcal{M}_{p_\theta}(d_i))$, we optimize the TL module parameters φ as follows.

$$\varphi^* = \varphi - \beta \frac{1}{N'} \sum_{i=1}^{N'} \nabla_\varphi L(y_i, \mathcal{M}_{p_\theta}(d_i)). \tag{13}$$

where β is a weighting factor. If the TL module $\mathcal{M}$ is differentiable w.r.t. φ, the above can be solved by stochastic gradient descent.

Note that the TL module is task specific. To examine the generality of our method, we evaluate it on two important tasks, namely item recommendation and text retrieval. For the item recommendation task, the base TL module is a fully-shared module with a shared hidden layer and two domain-specific output layers [3]. For the text retrieval task, the base TL module uses Decomposable Attention module (DAM) [25] as the shared hidden layer, the output layers are also domain specific. The details are in the experiment section.

[3] In case of multi-task learning, we can use multiple domain-specific output layers here

Algorithm 1: Inference of our Minimax Game based Tranfer Learning (MGTL).

Input : Selector $p_\theta(d)$, discriminator $p_\phi(d)$, TL module $p_\varphi(d)$, source data $\mathcal{V}_s$, target data $\mathcal{V}_t$ and validation data $\mathcal{V}$.

Initialize θ, ϕ, φ with random weights
Pre-train $p_\phi(d)$ using $\mathcal{V}_s$ and $\mathcal{V}_t$
Pre-train $p_\varphi(d)$ using $\mathcal{V}_t$
repeat:
for *each batch* $\{X_b^s, X_b^t\}$ *in an episode* **do**
  Sample $X_b^{s\prime}$ from X_b^s based on $p_\theta(d) > \tau$
  for *discriminator* **do**
    Obtain immediate reward r_i using Eq. (7)
    Update $p_\phi(d)$ using Eq. (5) with $\{X_b^{s'}, X_b^t\}$
  end
  for *TL module* **do**
    Update $p_\varphi(d)$ using Eq. (13) with $\{X_b^{s'}, X_b^t\}$
    Obtain delayed reward r_b using Eq. (9)
  end
  Store $(S_b, A_b, r_b, \{r_i\}_{i=1}^b)$ in episode history H
end
for *each tuple* $(S_b, A_b, r_b, \{r_i\}_{i=1}^b)$ *in the history H* **do**
  for *selector* **do**
    Obtain the future total reward R_i using Eq. (12)
    Update $p_\theta(d)$ using Eq. (11)
  end
end

2.6 Inference

The selector, discriminator, and TL module are learned jointly as they interact with each other closely during training. The overall training procedure is in Algorithm 1. Our model is trained in batches. For each batch, the selector selects data from the source batch X_b^s based on $p_\theta(\cdot)$. The discriminator provides immediate rewards to the selected source data $X_b^{s'}$, and then update its module to try to separate the source batch $X_b^{s'}$ from target batch X_b^t. Meanwhile, the TL module uses X_b^t and $X_b^{s'}$ to update its parameters and in turn, provides delayed reward to guide the selector. We have test different batch sizes for X_b^s and X_b^t, in general it is better if we set a larger original source batch size than target batch size, i.e. $|X_b^s| > |X_b^t|$. Empirically, we set source batch size as two times of target batch size, i.e. $|X_b^s| = 2|X_b^t|$.

Note that our model is trained in batches where each batch data updates the TL and discriminator modules. This means each update has a potential effect on the future model performance. This process is close to Markov Decision Process (MDP) where each step has a potential effect on the future return. Thus it is beneficial to organize the batches into different steps in an episode, where model performance in future steps can provide future rewards to the current step. An ideal case is to treat all the data as an episode. However in practice, the source data is very large which makes the episode too long. As a trade-off, we set a maximum episode length in our experiments. We find to use 50 to 500 batches for an episode is generally good for our datasets.

3 EXPERIMENTS

To examine the generality of our proposed model, we evaluate it on two different real-world applications namely item recommendation and text Matching. Specifically, we examine these questions:
Q1: Does our method perform well on item recommendation and text Matching task?
Q2: Does our method help to improve the efficiency of the task?
Q3: Are our model performances sensitive to model hyper-parameters?
Q4: Are the selected data instances reasonable and interpretable?
Q5: Is the method helpful for an online production system?

3.1 Datasets

Item Recommendation: We collected item recommendation data from two E-commerce websites (platform A and B) [4]. We have collected one-week user click-through data from these platforms, where 52000M data are from platform A and 350M data from B. Clearly for platform A, it has sufficient data to train a model, while for platform B, it may not have enough data. And if all the data of the platform A is used for transfer learning, it is obviously not cost-effective. Thus, it is more appropriate to select data that is more helpful in improving the performance of the target domain. We present data statistics and sample features for our dataset in Tab. 4 and 5 in Appx. A.
Text Matching: We use two open text matching datasets to evaluate our method: MultiNLI [40] as the source domain data and SciTail [19] as the target domain data. Each sample in the dataset is a sentence pair that contains a premise, a hypothesis, and a relation label of ENTAILMENT or NEUTRAL. We adopt the same dataset setting from this study [27].

3.2 Baseline models

We compare our full model, denoted as MGTL, with these baselines:

- **Src-only.** A model that uses only source data. This is to examine whether source data is helpful for other domains;
- **Tgt-only [25].** A model that uses its own data for training;
- **TL Method.** A typical transfer learning method [22, 23] that uses a shared encoder for both source and target domains, and domain-specific output layers for each domain. This model is also known as fully-shared model, a variant of our model without using data selection.
- **Ruder and Plank [29].** A recently proposed instance selection method with Bayesian optimization.
- **RTL [27].** A recent proposed RL based instance selection method for transfer learning.

3.3 Model performance on item recommendation and text matching (Q1)

The detailed model settings is presented in Appx. B. We evaluate our model performance on the task of item recommendation and text matching in terms of ACC and AUC. As shown in Tab. 1, a few important observations are as follows. First, Src-only performs worse than Tgt-only which means source data is close to but different from target data, i.e. there is domain shift between

[4] Platform A is Taobao (https://taobao.com),
Platform B is Qintao (https://qintao.taobao.com/).

source domain and target domain. In this case, simply using data from source domain for target domain is not good. Second, the TL method improves the Tgt-only model which means the TL method can help to leverage information from source domain to help the target domain. Furthermore, the instance selection method in [29] can further improve the TL model, which means its necessary to selected high-quality source data for the TL model. Last but not least, our method achieves better performance than the method in [29]. The method in [29] treats TL module as a sub-module which needs to be repeated trained to provide sufficient updates to the Bayesian optimization based selection method. Our selector, discriminator and TL module are trained jointly and thus the selector is updated more efficiently and effectively. Our method has better performance than the RTL method which further shows our immediate rewards and delayed rewards are helpful for our task. In all, the comparison with other competing methods shows the advantage of our proposed instance selection policy. Meanwhile, we find our method helps more for the text matching data. A potential reason is the transfer learning based method generally helps more for domains with less data.

Table 1: Evaluations of our method on the item recommendation and text matching datasets. † means the statistically significant difference over the strongest baseline with $p < 0.05$ measured by the student's paired t-test. Note that the metric AUC is an overall measurement which is not suitable for t-test.

Methods	Item Recommendation		Text Matching	
	ACC	AUC	ACC	AUC
Src-only	0.8905	0.6913	0.7112	0.7087
Tgt-only	0.9013	0.7031	0.7300	0.7663
TL Method	0.9097	0.7212	0.7453	0.8044
Ruder and Plank	0.9120	0.7328	0.7521	0.8062
RTL	0.9125	0.7331	0.7672	0.8163
MGTL	**0.9153**†	**0.7452**	**0.7782**†	**0.8247**

3.4 Training efficiency of MGTL (Q2)

To examine the training efficiency of MGTL, we train our MGTL by selecting data with different thresholds τ as in Eq. 8 in item recommendation datasets.

$\text{MGTL}_{0.0}$ - To train our MGTL with $\tau = 0.0$, i.e. use all the data. This uses the TL framework in [22, 23].

$\text{MGTL}_{0.2}$ - To train our MGTL with $\tau = 0.2$, i.e. selecting source data with threshold greater than 0.2.

$\text{MGTL}_{0.5}$ - To train our MGTL method by setting $\tau = 0.5$.

As shown in Tab. 2, we examine the number of selected data instances from source and training time to achieve a good AUC. Clearly, $\text{MGTL}_{0.0}$ takes much more data and time to reach a high AUC, this is because of the source data have a different distribution with the target data, adding all but not high-quality source data is not efficient and effective. With a larger threshold, i.e. choosing a higher quality of source data, the training process speeds up. Our model $\text{MGTL}_{0.5}$ takes far less time comparing to $\text{MGTL}_{0.0}$, with only around 1/2 of time and round 50% of data instances, it reaches a higher AUC. This shows our method is both effective and efficient for this task.

Table 2: Training efficiency of MGTL.

Model	Selected Data	Time(hour)	AUC
$\text{MGTL}_{0.0}$(TL Method)	7300M	2.82h	0.7212
$\text{MGTL}_{0.2}$	6500M	2.13h	0.7364
$\text{MGTL}_{0.5}$	4000M	1.28h	0.7452

We also evaluate the training efficiency of our model in text matching data. Similar to Table 2, our model $\text{MGTL}_{0.5}$ takes far less time comparing to $\text{MGTL}_{0.0}$, with only around 1/3 of time and round 35% of data instances, it reaches a higher AUC. This further echoes our finding that our method is both effective and efficient for this task.

3.5 Empirical analysis (Q3)

3.5.1 Model performance w.r.t. different steps and thresholds. We first examine our model performance w.r.t. different steps in item recommendation dataset in the left sub-figure of Fig. 4. We compare our method (MGTL) with a random baseline that selects the same amount of random source data instances (4000M) as in our model. Clearly, our MGTL achieves far better performance than the random baseline which shows the selected data instances are useful for our task. Meanwhile, selecting high-quality source data helps to converge faster as we find MGTL takes less time to achieve a good AUC than the random baseline. And the resulting AUC is also 0.028 higher than the random method.

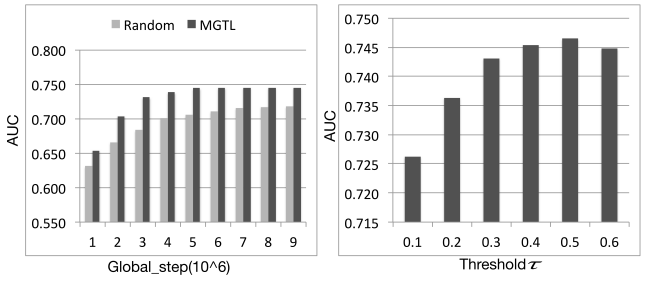

Figure 4: Model performance w.r.t. different thresholds τ.

Furthermore, in our experiments, we set the threshold τ as 0.5, we tune the threshold to examine our model performance in the right sub-figure of Fig. 4. We find that in general, setting a large threshold helps more for the model, which means selecting data closer to the target domain helps more for the task. However, larger threshold $\tau = 0.6$ does not necessary provides better performance than $\tau = 0.5$. The reason is that a larger threshold may result in a smaller set of selected source data, which may not help more for the target domain. We find setting $\tau = 0.5$ provides a reasonably good trade-off between quality and quantity of selected source instances.

The results on the text matching dataset are similar, where the choice of threshold τ in the range of $[0.3, 0.5]$ has relatively better performance. In practice, the choice of threshold τ is related to the distribution of the source and target domain data: if two domains are different, a large threshold is better, otherwise a small threshold is sufficient.

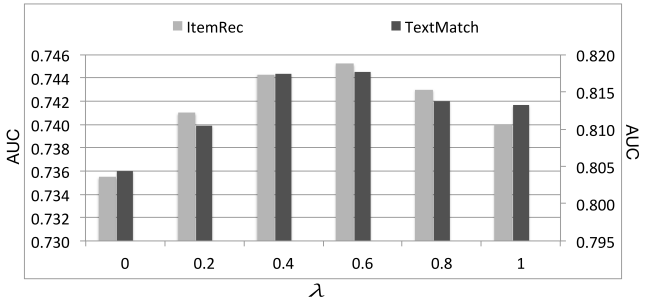

Figure 5: Model performance w.r.t. differet thresholds λ.

3.5.2 Model performance w.r.t. different hyper-parameter λ. Recall in Eq. 12, λ is to weight the importance of delayed reward and immediate reward. As in Fig. 5, we find only delayed reward($\lambda = 0$) is worse than using immediate reward ($\lambda = 1$). A possible reason is the instance level (or immediate) reward is better than batch level (or delayed) reward to the selector. In all, it is beneficial to consider both rewards as we find the best results are achieved by setting $\lambda = 0.6$.

3.6 Qualitative evaluation (Q4)

We perform qualitative evaluations of the selected data by our selection model.

3.6.1 Visualization of selected data. First, in the item recommendation dataset, we choose the distribution of two types of features to illustrate: price level and brand level. We divide all the items into 6 different price ranges, from the highest price level *price_level_6* to the lowest price level *price_level_1*. We then visualize the distribution of these price ranges in the original dataset, the selected dataset, and the target dataset respectively in Fig. 6. We find that the price distribution on the original data set is relatively uniform, while on the target data set, it is biased toward a lower price. It is because the source data is from a large e-commerce platform, covering users with a wide range of consumption levels. The target dataset mainly covers users who are looking for economic or cheap items. Thus the distributions are quite different. With the help of our model, the selected data instances are with a similar price distribution with the target domain. Similarly, take the brand distribution in Fig. 6 for example, our selected data instances are with a much similar brand distribution [5] with the target domain than the original source domain.

Furthermore, in the text matching dataset, we would also like to examine the feature (word) distribution of the selected, source, and target data. To give a better idea of the data distributions, we compute the Wasserstein distance between the term distributions of these data. The results are presented in Table 3.

We observe that $D_{Sel-Tgt}$ is smaller than $D_{Src-Tgt}$, which means that the source domain data selected by our method is closer to the target domain data and thus may contribute to the transfer learning process. And $D_{Drop-Tgt}$ is larger than $D_{Src-Tgt}$, which means that the source domain data dropped are not very similar

[5] The items are divided into 6 different brand ranges, from the most high-end brand level $brand\_level\_6$ to the lowest brand level $brand\_level\_1$.

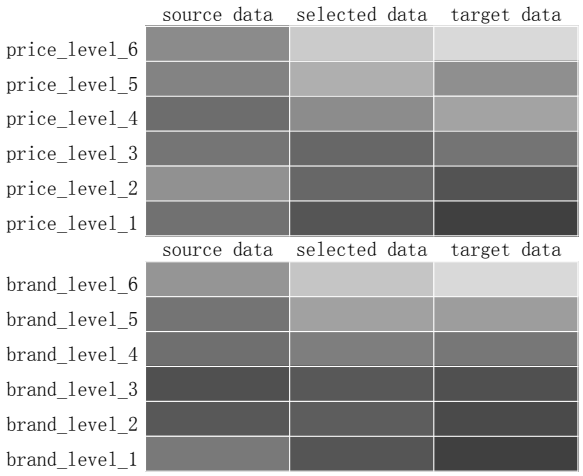

Figure 6: An illustration of price & brand distributions of different datasets.

Table 3: The Wasserstein distances between the term distributions of different domains.

	$D_{Src-Tgt}$	$D_{Sel-Tgt}$	$D_{Drop-Tgt}$
Wasserstein distances	3.256E-06	3.012E-06	3.340E-06

to the target domain data and thus may cause negative transfer. This further echoes our findings that our method is able to select data instances are with a much similar distribution with the target domain to help the transferring task.

In a nutshell, the data selected by our MGTL model are very insightful and the selection model behaviors are reasonable.

3.6.2 Visualization of the process of minimax game. The minimax game based selection model is the key for MGTL. Thus, it is important to know whether the distribution of the data selected by the selection model is similar to the target data at different training steps. To verify whether the selection model is working as we expected, we employ the t-SNE method to visualize the feature of the discriminate model output from two aspects.

Firstly, with the training goes on, we visualize the output features of discriminate model to verify whether the data selected from source domain are becoming more and more similar to the target domain. As in Fig. 7, the threshold τ of MGTL model is set to 0.5, and the blue and red points refer to the selected source data and target data respectively. As the training progresses, the features from different domains are getting closer and cannot be separated by discriminate model.

Secondly, when the training stage is finished, the selection model and discriminator model reach a dynamic equilibrium. We expect the different choices of threshold τ correspond to different range of discriminativeness of selected data. We change the threshold τ of MGTL model to select data with different discriminativeness, and visualize the output features of discriminate model in Fig 8. Again the blue and red points refer to the selected source data and target data. Take the similarity range [0,0.1] for example, the features

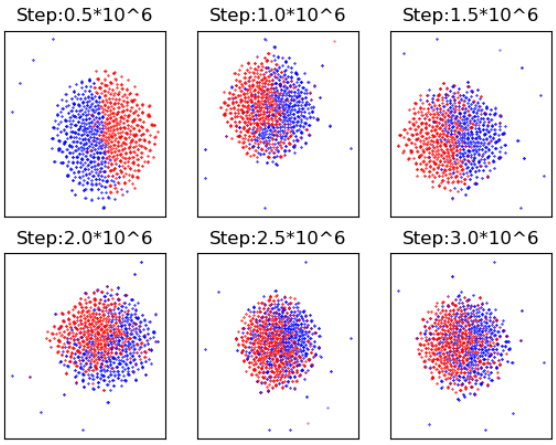

Figure 7: Selected data instances at different training steps.

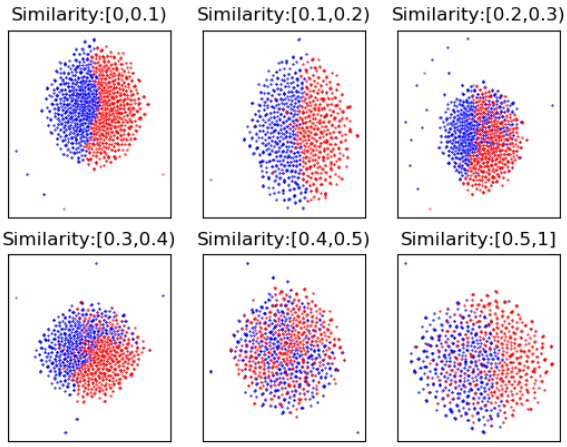

Figure 8: Discriminativeness of selected data at different thresholds when training stage is finished.

from different domains are almost completed separated. Because the data selected by this threshold is not similar to the target data. And with the threshold increases, the selected data are more and more inseparable from the target domain data.

In all, the interactions between the selector and discriminator help our model to select high-quality source data instances that are close to the target domain.

3.7 Online evaluations (Q5)

Last but not least, we apply our method on the platform Qintao to leverage information from platform Taobao. We compare our full model with a baseline model which is a degenerated version of our model that does not consider source instance selection. We have run 7-day A/B testing on the click-through-rate of the models. Our method outperforms the online model, a degenerated version of our model without considering data selection, yielding 3%+ improvement on average. This brings to an increase of 1.5%+ in terms of total gross merchandise volume, which is considered to be a big improvement for the platform.

4 RELATED WORK

Transfer Learning (TL). TL has been widely studied in the past years [24]. A widely used TL method is supervised domain adaptation, which assumes to have enough labeled data from a source domain and a little labeled data from a target domain [8]. This is different from the partial transfer learning [3] where source label space subsumes target label space. Supervised domain adaptation methods have two categories: feature-based and instance-based. The former seeks to find a shared feature space to reduce the divergence between the distribution of the source and the target domains [1, 35]. With the popularity of deep neural networks, there is a growing number of studies working on using neural network based methods. In these methods, the shared feature space is learned by projecting both domains into a common latent space using neural network [22, 23]. And the adversarial training can be also considered to improve the TL model performance [12, 20, 32, 34]. However they may have the negative transfer problem when the source data distribution is largely different from the target data.

The latter category select or re-weight the source domain training samples so that data from the source domain and the target domain would share a similar data distribution [7, 39]. Although various data selection methods [4, 26, 29] were proposed for TL settings, they may not be ideal for deep TL modules [38]. As these methods treat the TL module as a sub-module that needs to be trained repetitively to provide sufficient updates to the data selector. The complex nature of neural architectures makes the training process less cost-effective. In this paper, we propose a general minimax game based method for selective transfer learning. We build three components: a data selector, a discriminator, and a TL module in our model. The interactions between these components play a minimax game to find useful source data to help the target domain.

Generative Adversarial Network (GAN). Our minimax game setting is close to GAN[13]. In a typical GAN setting, a generator learns a generative network G and discriminator D, in which G generates samples from the generator distribution, and D learns to determine whether a sample is a real data or generated data. And the modules G and D are trained by a min-max game like loss function. There are some recent studies on using GANs for information retrieval, e.g. IRGAN [36] and Adversarial Personalized Ranking (APR) [16]. However they both focus on a single domain or dataset, and our work is the first proposed to utilize GAN framework to explore the transferring ability between different domains for ranking tasks in IR.

Furthermore, the generator in GAN usually uses reinforcement learning (RL) method to update. Reinforcement learning algorithms can be categorized into two types: value based methods [21, 30] and policy based methods [41]. We experiment with policy based methods to update the data selector in our method. Recently, RL has been applied to data selection in many tasks [9–11, 42]. However, there is a lack of RL methods for DNN based selective transfer learning. Furthermore, our method experiments two types of rewards in the RL method, one is from the discriminator and the other from TL

module, these two together help our module to find high-quality source data for transfer learning.

Item Search and Text Retrieval. For item search, the core task is to rank items according to user and item features. DNNs are widely used for this task, typical methods include: Deep-only method [6], Wide & Deep [5], DeepFM [14, 15], and Deep&Cross [37]. In our method, we adapt Deep-only based module as our hidden representation layer. For text retrieval, the core problem is to measure the similarity between two pieces of texts. Typical models for such tasks include ABCNN [44] and Decomposable Attention module (DAM) [25]. We use the DAM as our base module, as it has an effective performance with remarkable efficiency. Note that our method is general as any of the above methods can be used as our base module. We leave the exploration of more methods as our future work.

5 CONCLUSION

In this work, we proposed a minimax game based transfer learning method for selective transfer learning. Our method has a selector, a discriminator and a transfer learning module, and they play a minimax game to find useful source data. We evaluate our method on two real-world tasks namely item recommendation and text matching to examine its generality. Extensive experiments over both public datasets and real-world datasets show that our model outperforms the competing methods by a large margin. We have also deployed the proposed method in a large e-commerce platform and observed significant improvement over the baseline models.

REFERENCES

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *NIPS*.

[2] John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *ACL* (2007).

[3] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. 2017. Partial Transfer Learning with Selective Adversarial Networks. *CoRR* (2017).

[4] Minmin Chen, Kilian Q. Weinberger, and John C. Blitzer. 2011. Co-training for Domain Adaptation. In *NIPS*.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *CoRR* abs/1606.07792 (2016).

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys '16*. 191–198.

[7] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for Transfer Learning. In *ICML*. 193–200.

[8] Hal Daume III. 2007. Frustratingly Easy Domain Adaptation. In *ACL*.

[9] Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2017. Learning What Data to Learn. *CoRR* (2017).

[10] Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to Active Learn: A Deep Reinforcement Learning Approach. In *EMNLP*.

[11] Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement Learning for Relation Classification From Noisy Data. In *AAAI*.

[12] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial Training of Neural Networks. *J. Mach. Learn. Res.* 17, 1 (Jan. 2016), 2096–2030.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

[14] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* abs/1703.04247 (2017).

[15] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of SIGIR*. 355–364.

[16] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 355–364.

[17] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard. Scholkopf. 2006. Correcting Sample Selection Bias by Unlabeled Data. *In NIPS*. (2006).

[18] Ferenc Huszar. 2015. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv:1511.05101*. (2015).

[19] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A Textual Entailment Dataset from Science Question Answering. In *AAAI*.

[20] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. *In Proceedings of ACL*. (2017).

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* (2015).

[22] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *ACL*.

[23] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications?. In *EMNLP*.

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

[25] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.

[26] Yash Patel, Kashyap Chitta, and Bhavan Jasani. [n. d.]. Learning Sampling Policies for Domain Adaptation. *CoRR, abs/1805.07641, 2018*. ([n. d.]).

[27] Chen Qu, Feng Ji, Minghui Qiu, Liu Yang, Zhiyu Min, Haiqing Chen, Jun Huang, and W. Bruce Croft. 2019. Learning to Selectively Transfer: Reinforced Transfer Learning for Deep Text Matching. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*.

[28] Michael T. Rosenstein, Zvika Marx, Leslie Pack Kael-bling, and Thomas G.Dietterich. 2005. To Transfer or Not To Transfer. *NIPS Workshop on Inductive Transfer* (2005).

[29] Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian Optimization. *In EMNLP*. (2017).

[30] Gavin A. Rummery and Mahesan Niranjan. 1994. *OnLine Q-Learning Using Connectionist Systems*. Technical Report. University of Cambridge.

[31] Tobias Schnabel and Hinrich SchuĬLtze. 2014. FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *TACL, 2:15âĂŞ26*. (2014).

[32] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein Distance Guided Representation Learning for Domain Adaptation. In *AAAI*. AAAI Press.

[33] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning - An Introduction*. MIT Press.

[34] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial Discriminative Domain Adaptation. *CoRR* abs/1702.05464 (2017).

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

[36] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of SIGIR*. ACM, 515–524.

[37] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*. 12:1–12:7.

[38] Tianyang Wang, Jun Huan, and Michelle Zhu. 2018. Instance-based Deep Transfer Learning. In *WACV*.

[39] Junfeng Wen, Chun-Nam Yu, and Russell Greiner. 2014. Robust Learning Under Uncertain Test Distributions: Relating Covariate Shift to Model Misspecification *(ICML'14)*. JMLR.org, II–631–II–639.

[40] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL*.

[41] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning (1992)* (1992).

[42] Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced Co-Training. In *NAACL*.

[43] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. *In ICLR* (2017).

[44] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *TACL* (2016).

[45] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *In AAAI*. (2017).

[46] Fuzhen Zhuang, Lang Huang, Jia He, Jixin Ma, and Qing He. 2017. Transfer Learning with Manifold Regularized Convolutional Neural Network, Gang Li, Yong Ge, Zili Zhang, Zhi Jin, and Michael Blumenstein (Eds.). Springer International Publishing, Cham, 483–494.

A DATASETS

The data statistics are presented in Tab. 4. For item search, we collected one-week user click-through data from two E-commerce websites, namely Taobao(https://taobao.com) and Qintao(https://qintao.taobao.com). These two platforms share similar users and items features, but data distributions are different. Platform A is with much larger data instances than platform B, thus serving as the source domain.

For text matching, we use two open text matching datasets to evaluate our method: MultiNLI [40] and SciTail [19]. MultiNLI is a large crowd-sourced corpora for textual entailment recognition, it has diverse text sources and thus is more suitable to serve as the source domain in a TL setting. SciTail is a recently released textual entailment dataset in the science domain. The SciTail dataset is smaller than MultiNLI but with more diversity in terms of linguistic variations. We adopt the same dataset setting from this study [27]. Note that each sample in MultiNLI is a sentence pair that contains a premise, a hypothesis, and a relation label of ENTAILMENT, NEUTRAL, or CONTRADICTION. However, the labels in SciTail only consists of Entailment and Neutral. Therefore, we remove the Contradiction samples from MultiNLI in order to use it as source domain.

Table 4: One E-commerce dataset and one text matching dataset.

Data	Item Recommendation		Text Matching	
	A (Taobao)	B (Qintao)	MultiNLI	SciTail
# instances	52000M	350M	262K	24K
# fields	254		50	
# features	10,00M		190K	

The sample features for our dataset are in Tab. 5. We mainly consider these five types of features: user profile, item profile, query statistics, item statistics, and user behaviors. Each type of feature contains multiple "fields".

B MODEL SETTINGS

For the item recommendation dataset, we treat user click or purchase behaviors as positive labels (y=1), and others as negative labels (y=0). We have continuous and categorical features, where for all the continuous features, we perform feature discretization to map them to categorical features.

We set the shared feature embedding size among the discriminative module and TL module as 128. The discriminative is a fully-connected network with a structure of $512 \times 256 \times 64 \times 1$. The TL module is a fully-shared module whose shared hidden layers are with a structure of $512 \times 256 \times 128$, and two domain specific output layers are with a structure of 64×1. As for the selection module, we use a feature embedding with size 128, the hidden layers are with a structure of $512 \times 256 \times 64 \times 1$. The source batch size is set as 2048, and the target batch size is 1024.

For the text matching dataset, we set the word embedding size of 300. The word embeddings are initialized with GloVe and finetuned in our model. The TL module uses Decomposable Attention Model (DAM) as base module [25]. The hidden layer size for DAM is 200. The two domain specific output layers are with a structure of 32×1. The selection module, its word embeddings are with a hidden size of 300 and initialized using GloVe. The hidden layers are with a structure of $64 \times 32 \times 1$. The discriminator is a fully-connected network with a structure of $512 \times 256 \times 1$. The source batch size is set as 256, and the target batch size is 128.

Table 5: Sample features in our item recommendation dataset. PV and IPV refer to page view and item page view respectively.

Category	Feature	Dimension	Type	#
User profile	user id	5×10^8	one-hot	1
	gender	2	one-hot	1
	age level	6	one-hot	1
	...	...	...	...
Item profile	item id	$\sim 2 \times 10^8$	one-hot	1
	price level	6	one-hot	1
	brand level	6	one-hot	1
	...	...	...	...
Query statistic	search PV 1 day	~ 100	one-hot	1
	search PV 7 days	~ 100	one-hot	1
	search PV 14 days	~ 100	one-hot	1
	...	...	...	...
Item statistic	IPV 1 day	~ 100	one-hot	1
	IPV 7 days	~ 100	one-hot	1
	IPV 14 days	~ 100	one-hot	1
	...	...	...	...
User behavior sequence	click items	$\sim 2 \times 10^8$	multi-hot	~ 50
	cart items	$\sim 2 \times 10^8$	multi-hot	~ 50
	pay items	$\sim 2 \times 10^8$	multi-hot	~ 50
	...	...	...	...

The target dataset is randomly splitted into three parts, where 80% of target dataset is using for training, 10% is for testing, and the last 10% is for validation.

We use ReLU as the activation functions for the NN layers and Adam as the optimizer. The learning rate is set as 0.001. The threshold τ is set as 0.5. The λ is set to 0.6. The γ is set to 0.75. All the methods in this paper are implemented with TensorFlow [6] and are trained with NVIDIA Tesla K40M GPUs.

[6] www.tensorflow.org