Cross-Domain Sentiment Classification with Target Domain Specific Information

Minlong Peng, Qi Zhang*, Yu-gang Jiang, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{mlpeng16,qz,ygj,xjhuang}@fudan.edu.cn

Abstract

The task of adopting a model with good performance to a target domain that is different from the source domain used for training has received considerable attention in sentiment analysis. Most existing approaches mainly focus on learning representations that are domain-invariant in both the source and target domains. Few of them pay attention to domain specific information, which should also be informative. In this work, we propose a method to simultaneously extract domain specific and invariant representations and train a classifier on each of the representation, respectively. And we introduce a few target domain labeled data for learning domain-specific information. To effectively utilize the target domain labeled data, we train the domain-invariant representation based classifier with both the source and target domain labeled data and train the domain-specific representation based classifier with only the target domain labeled data. These two classifiers then boost each other in a co-training style. Extensive sentiment analysis experiments demonstrated that the proposed method could achieve better performance than state-of-the-art methods.

1 Introduction

Sentiment classification aims to automatically predict sentiment polarity of user generated sentiment data like movie reviews. The exponential increase in the availability of online reviews and recommendations makes it an interesting topic in research and industrial areas. However, reviews can span so many different domains that it is difficult to gather annotated training data for all of them. This has motivated much research on cross-domain sentiment classification which transfers the knowledge from label rich domain (source domain) to the label few domain (target domain).

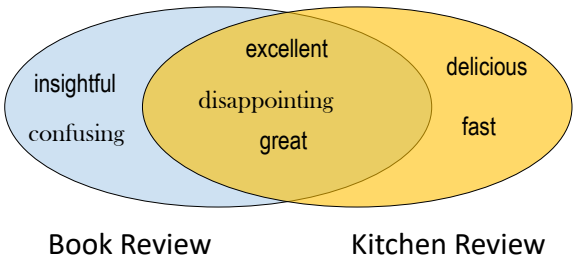

Figure 1: Top indicators extracted with logistic regression for Book and Kitchen domains. The overlap between the two ellipses denotes the shared features between these two domains.

In recent years, the most popular cross-domain sentiment classification approach is to extract domain invariant features, whose distribution in the source domain is close to that in the target domain. (Glorot et al., 2011; Fernando et al., 2013; Kingma and Welling, 2013; Aljundi et al., 2015; Baochen Sun, 2015; Long et al., 2015; Ganin et al., 2016; Zellinger et al., 2017). And based on this representation, it trains a classifier with the source rich labeled data. Specifically, for data of the source domain $\mathbf{X}_s$ and data of the target domain $\mathbf{X}_t$, it trains a feature generator $G(\cdot)$ with restriction $P(G(\mathbf{X}_s)) \approx P(G(\mathbf{X}_t))$. And the classifier is trained on $G(\mathbf{X}_s)$ with the source labels $\mathbf{Y}_s$. The main difference of these approaches is the mechanism to incorporate the restriction on $G(\cdot)$ into the system. The major limitation of this framework is that it losses the domain specific information. As depicted in Figure1, even if it can perfectly extract the domain

*Corresponding author.

invariant features (e.g., excellent), it will loss some strong indicators (e.g., delicious, fast) of the target Kitchen domain. We believe that it can achieve greater improvement if it can effectively make use of this information.

Thus, in this work, we try to explore a path to use the target domain specific information with as few as possible target labeled data. Specifically, we first introduce a novel method to extract the domain invariant and domain specific features of target domain data. Then, we treat these two representations as two different views of the target domain data and accordingly train a domain invariant classifier and a target domain specific classifier, respectively. Because the domain invariant representation is compatible with both source data and target data, we train the domain invariant classifier with both source and target labeled data. And for the target domain specific classifier, we train it with target labeled data only. Based on these two classifiers, we perform co-training on target unlabeled data, which can further improve the usage of target data in a bootstrap style.

In summary, the contributions of this paper include: (i) This is the first work to explore the usage of target domain specific information in cross-domain sentiment classification task. (ii) We propose a novel to extract the domain specific representation of target domain data, which encodes the individual characteristics of the target domain.

2 Related Work

Domain adaptation aims to generalize a classifier that is trained on a source domain, for which typically plenty of labeled data is available, to a target domain, for which labeled data is scarce. In supervised domain adaptation, cross-domain classifiers are learnt by using labeled source samples and a small number of labeled target samples (Hoffman et al., 2014). A common practice is training the cross-domain classifiers with the labeled source data and then fine-tuning the classifier with the target labeled data (Pan and Yang, 2010). Meanwhile, some unsupervised and semi-supervised cross domain methods (Ganin et al., 2016; Louizos et al., 2015; Zellinger et al., 2017) are proposed by combining the transfer of classifiers with the match of distributions. These methods focus on extracting the domain-invariant features with the help of unlabeled data. Specifically, Ganin et al., (2016) incorporated an adversarial framework to perform this task. It trained the feature generator to minimize the classification loss and simutaneously deceive the discriminator, which is trained to distinguish the domain of the input data coming from. Louizos et al., (2015) used the Maximum Mean Discrepancy (Borgwardt et al., 2006) regularizer to constrain the feature generator to extract the domain invariant features. And similarly, Zellinger et al., (2017) proposed the central moment discrepancy (CMD) metric for the role of domain regularizer. The above methods either treat it no difference between domain specific information and domain invariant information or just ignore the domain specific information during in the process of learning adaptive classifiers.

One of the most related work is the DSN model (Bousmalis et al., 2016). It proposed to extract the domain specific and the domain invariant representations, simultaneously. However, It does not explored the usage of the domain specific information. Its classifier was still only trained on the domain invariant representation. This work differs from it in the following two aspects. First, we make use of the source and target unlabeled data to extract domain specific information, instead of relying on the orthogonality constraint between the extra representation and the domain invariant counterpart. It is achieved by forcing the distribution of the source examples and that of the target examples in the domain specific space to be different. We argue that this can avoid the potential problem of the orthogonality constraint in that the domain specific representation can be well predicted by the domain invariant representation, while simultaneously meeting the orthogonality constraint. For example, let $\boldsymbol{X} = (\boldsymbol{0}, \boldsymbol{Z})$ be the domain invariant representation and $\boldsymbol{Y} = (\boldsymbol{Z}, \boldsymbol{0})$ be the domain specific representation, then $\boldsymbol{X}$ can be uniquely determined by $\boldsymbol{Y}$, while in the meanwhile $\boldsymbol{X} \perp \boldsymbol{Y}$. Second, we apply a co-training framework to make use of the domain specific representation, rather than simply treating it as a regularizer for extracting the domain invariant representation.

Another related work is the CODA model (Chen et al., 2011). It also applied a co-training framework for semi-supervised domain adaptation. However, instead of dividing the feature space into domain invariant and domain specific

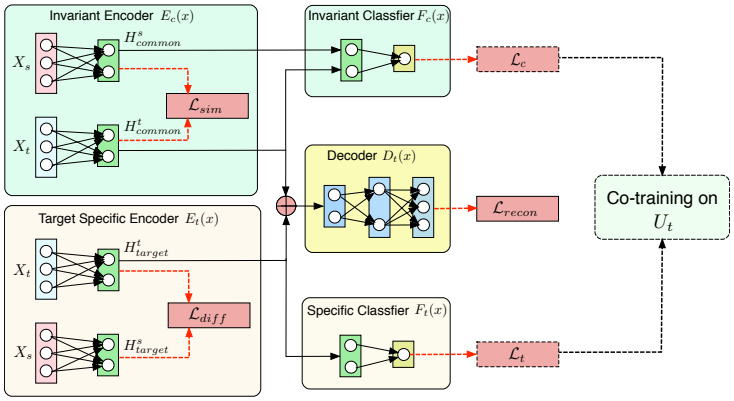

Figure 2: The general architecture of the proposed model. The source data $\boldsymbol{X}_s$ and target data $\boldsymbol{X}_t$ are mapped to a domain invariant representation and a target domain specific representation by feature maps E_c and E_t, respectively. In the space of the domain invariant representation, the distributions of source data $\boldsymbol{H}^s_{inv}$ and target data $\boldsymbol{H}^t_{common}$ are forced to be similar by minimizing a certain distance $\mathcal{L}_{sim}$. In contrast, in the space of the target domain specific representation, the distributions of source data $\boldsymbol{H}^s_{spec}$ and target data $\boldsymbol{H}^t_{spec}$ are forced to be different by minimizing the distance $\mathcal{L}_{diff}$. Based on the domain invariant representation, a classifier F_c is trained with the source rich labeled data and some of the target labeled data. In addition, based on the target domain specific representation, a classifier F_t is trained with the target labeled data only. These two classifiers teach each other in a co-training framework based on the target unlabeled data $\boldsymbol{U}_t$.

parts, it randomly separated the features space.

3 Approach

We consider the following domain adaptation setting. The source domain consists of a set of n_s fully labeled points $D_s = \{(\mathbf{x}^s_1, \mathbf{y}^s_1), \cdots, (\mathbf{x}^s_{n_s}, \mathbf{y}^s_{n_s})\} \subset R^d \times \mathcal{Y}$ drawn from the distribution $P_s(\mathbf{X}, \mathbf{Y})$. And the target data is divided into n_l ($n_l \ll n_s$) labeled points $D^l_t = \{(\mathbf{x}^t_1, \mathbf{y}^t_1), \cdots, (\mathbf{x}^t_{n_l}, \mathbf{y}^t_{n_l}\} \subset R^d \times \mathcal{Y}$ from the distribution $P_t(\mathbf{X}, \mathbf{Y})$ and n_u ($n_u \gg n_l$) unlabeled points $D^u_t = \{(\mathbf{x}^t_{n_l+1}, \mathbf{y}^t_{n_l+1}), \cdots, (\mathbf{x}^t_{n_l+n_u}, \mathbf{y}^t_{n_l+n_u}\} \subset R^d$ from the marginal distribution $P_t(\mathbf{X})$. The goal is to build a classifier for the target domain data using the source domain data and a few labeled target domain data.

In the following section, we first introduce the CMD metric, which is used to measure the probability distribution discrepancy between two random variables. Then, we describe our method to extract the domain specific and domain invariant representations of target domain examples, using the CMD-based regularizer. Finally, we show how to combine these two representations using a co-training framework.

3.1 Central Moment Discrepancy (CMD)

The CMD metric was proposed by Zellinger et al.(2017) to measure the discrepancy between the probability distributions of two (high-dimensional) random variables. It is one of the state-of-the-art metrics and is used as a domain regularizer for domain adaptation. Here, we introduce its definition as a domain regularizer.

Definition 1 *(CMD regularizer). Let X and Y be bounded random samples with respective probability distributions p and q on the interval $[a, b]^N$. The CMD regularizer CMD_K is defined by*

$$CMD_K(X, Y) = \frac{1}{|b-a|} \parallel E(X) - E(Y) \parallel_2 + \frac{1}{|b-a|^k} \sum_{k=2}^{K} \parallel C_k(X) - C_k(Y) \parallel_2, \tag{1}$$

where $E(X) = \frac{1}{|X|} \sum_{x \in X} x$ is the empirical expectation vector computed on the sample X and

$$C_k(X) = \left(E(\prod_{i=1}^{N} (X_i - E(X_i))^{r_i} \right)_{r_i \geq 0, \sum_i^N r_i = k},$$

is the vector of all k^{th} order sample central moments of the coordinates of X.

An intuitive understanding of this metric is that if two probability distributions are similar, their central moment of each order should be close.

3.2 Extract Domain Invariant and Domain Specific Representations

In this work, we aim to extract a domain invariant representation, as well as a domain specific counterpart, for each target example. This makes our work different from most of the existing works, which only focus on the domain invariant representation. The general architecture of the proposed model is illustrated in Figure 2. Data are mapped into a domain invariant hidden space and target domain specific hidden space using two different mappers E_t and E_c, respectively:

$$\begin{aligned} \boldsymbol{H}^s_{spec} &= E_t(\boldsymbol{X}_s; \boldsymbol{\theta}^t_e) \\ \boldsymbol{H}^t_{spec} &= E_t(\boldsymbol{X}_t; \boldsymbol{\theta}^t_e) \\ \boldsymbol{H}^s_{inv} &= E_c(\boldsymbol{X}_s; \boldsymbol{\theta}^c_e) \\ \boldsymbol{H}^t_{inv} &= E_c(\boldsymbol{X}_t; \boldsymbol{\theta}^c_e). \end{aligned} \tag{2}$$

Here, E_t refers to the domain invariant mapper and E_c is the target domain specific mapper. $\boldsymbol{\theta}^t_e$ and $\boldsymbol{\theta}^c_e$ denote their corresponding parameters. The subscript e denotes *encode*. Based on the hidden presentations $\boldsymbol{H}^t_{inv}$ and $\boldsymbol{H}^t_{spec}$, we build an auto-encoder for the target domain examples:

$$\hat{\boldsymbol{X}}_t = D_t(\boldsymbol{H}^t_{inv}, \boldsymbol{H}^t_{spec}; \boldsymbol{\theta}^t_d), \tag{3}$$

with respect to parameters $\boldsymbol{\theta}^t_d$, where the subscript d denotes *decode*. The corresponding reconstruction loss is defined by the mean square error:

$$\mathcal{L}_{recon} = \frac{1}{n_t} \sum_{i}^{n_t} \frac{1}{k} ||\boldsymbol{X}^i_t - \hat{\boldsymbol{X}}^{\,i}_t||^2_2, \tag{4}$$

where k is the dimension of the input feature vector, and $\boldsymbol{X}^i_t$ denotes the i^{th} example of the target domain data. Note that in this work, only target examples are passed to the auto-encoder because we only want to extract target domain specific information.

For E_c, we hope that it only encodes features shared by both the source and target domains. From the distribution view, we hope that the distributions of the mapped outputs, by E_c, of source and target data are similar. To this end, we apply the CMD regularizer onto the hidden representation of source data $\boldsymbol{H}^s_{inv}$ and that of target data $\boldsymbol{H}^t_{inv}$. The corresponding loss is defined by:

$$\mathcal{L}_{sim} = CMD_K(\boldsymbol{H}^s_{inv}, \boldsymbol{H}^t_{inv}). \tag{5}$$

Minimizing this loss will force the distribution of $\boldsymbol{H}^s_{inv}$ and $\boldsymbol{H}^t_{inv}$ to be similar, which in turn encourages E_c to encode domain invariant features.

And for the domain specific encoder E_t, we hope that it only encodes features dominated by the target domain. Ideally, these features should commonly appear in the target domain while hardly appear in the source domain. We argue that this can also be obtained by forcing the distribution of these features in the target domain to differ from that in the source domain, because the target specific auto-encoder D_t should filter out features that hardly appear in the target domain while commonly appear in the source domain. Based on this intuition, we apply a signal flipped CMD regularizer onto the mapped representation of source data $\boldsymbol{H}^s_{spec}$ and that of target data $\boldsymbol{H}^t_{spec}$. The corresponding loss is defined by:

$$\mathcal{L}_{diff} = -CMD_K(\boldsymbol{H}^s_{spec}, \boldsymbol{H}^t_{spec}). \tag{6}$$

Minimizing this loss encourages the distribution of $\boldsymbol{H}^s_{spec}$ to differ from that of $\boldsymbol{H}^t_{spec}$, which in turn encourage E_t to encode domain specific features.

3.3 Co-Training with Domain Invariant and Domain Specific Representations

The co-training algorithm assumes that the data set is presented in two separate views, and two classifiers are trained for each view. In each iteration, some unlabeled examples that are confidently predicted according to exactly one of the two classifiers are moved to the training set. In this way, one classifier provides the predicted labels to the unlabeled examples, on which the other classifier may be uncertain.

In this work, we treat the domain invariant representation and the domain specific representation as the two separate views of target domain examples. Based on the domain invariant representation, we train a domain invariant classifier, F_c, with respect to parameters $\boldsymbol{\theta}_c$. In addition, based on the domain specific representation, we train a domain specific classifier, F_t, with respect to parameters $\boldsymbol{\theta}_t$.

Because the distribution of the source examples is compatible with that of the target examples in

input:
L_s: labeled source domain examples
L_t: labeled target domain examples
U_t: unlabeled target domain examples
H^s_{inv}: Invariant representation of L_s
H^t_{inv}: Invariant representation of L_t
H^t_{spec}: Specific representation of L_t
repeat
  Train classifier F_c with L_s and L_t based on H^s_{inv} and H^t_{inv};
  Apply classifier F_c to label U_t;
  Select p positive and n negative the most confidently predicted examples U^c_t from U_t;
  Train classifier F_t with L_t based on H^t_{spec};
  Apply classifier F_t to label U_t;
  Select p positive and n negative the most confidently predicted examples U^t_t from U_t;
  Remove examples $U^c_t \cup U^t_t$ from U_t;
  Add examples $U^c_t \cup U^t_t$ and their corresponding labels to L_t;
until *best performance obtained on the developing data set*;

Algorithm 1: Co-Training for Domain Adaptation

the domain invariant hidden space, we use both the source rich labels and a few target labels to train the classifier F_c. To train the classifier F_t, only the target labels are used. The entire procedure is described in Algorithm 1.

3.4 Model Learning

The training of this model is divided into two parts with one for the domain invariant classifier, F_c, and another one for the domain specific classifier, F_t. For F_c, the goal of training is to minimize the following loss with respect to parameters $\mathbf{\Theta} = \{\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_d, \boldsymbol{\theta}_c\}$:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{recon}(\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_e, \boldsymbol{\theta}^t_d) + \alpha\mathcal{L}_c(\boldsymbol{\theta}^c_e, \boldsymbol{\theta}_c) \\ + \gamma\mathcal{L}_{sim}(\boldsymbol{\theta}^c_e) + \lambda\mathcal{L}_{diff}(\boldsymbol{\theta}^t_e),\end{aligned} \tag{7}$$

where α, γ, and λ are weights that control the interaction of the loss terms. $\mathcal{L}(\theta)$ means that loss, $\mathcal{L}$, is optimized on the parameters θ during training. And $\mathcal{L}_c$ denotes the classification loss on the domain invariant representation, which is defined by the negative log-likelihood of the ground truth class for examples of both source and target domains:

$$\begin{aligned}\mathcal{L}_c = \frac{1}{n_s + l_t}\sum_{i=1}^{n_s} -Y^i_s \log F_c(Y^i_s|E_c(L^i_s)) \\ + \frac{1}{n_s + l_t}\sum_{i=1}^{l_t} -Y^i_t \log F_c(Y^i_t|E_c(L^i_t)),\end{aligned} \tag{8}$$

where Y^i_s is the one-hot encoding of the class label for the i^{th} source example, Y^i_t is that for the i^{th} labeled target example, and l_t denotes the dynamic number of target labeled data in each iteration.

For F_t, the goal of training is to minimize the following loss with respect to parameters $\mathbf{\Theta} = \{\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_e, \boldsymbol{\theta}^t_d, \boldsymbol{\theta}_t\}$:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{recon}(\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_e, \boldsymbol{\theta}^t_d) + \beta\mathcal{L}_t(\boldsymbol{\theta}^t_e, \boldsymbol{\theta}_t) \\ + \gamma\mathcal{L}_{sim}(\boldsymbol{\theta}^c_e) + \lambda\mathcal{L}_{diff}(\boldsymbol{\theta}^t_e),\end{aligned} \tag{9}$$

where γ and λ are the same weights as those for the classifier F_c, and β is the weight that controls the portion of classification loss, $\mathcal{L}_t$, on the domain specific representation, which is defined by the negative log-likelihood of the ground truth class for examples of the target domain only:

$$\mathcal{L}_t = \frac{1}{l_t}\sum_{i=1}^{l_t} -Y^i_t \log F_t(Y^i_t|E_t(L^i_t)) \tag{10}$$

4 Experiment

4.1 Dataset

Domain adaptation for sentiment classification has been widely studied in the NLP community. The major experiments were performed on the benchmark made of reviews of Amazon products gathered by Blitzer et al. (2007). This data set[1] contains Amazon product reviews from four different domains: Books, DVD, Electronics, and Kitchen appliances from Amazon.com. Each review was originally associated with a rating of 1-5 stars. For simplicity, we are only concerned with whether or not a review is positive (higher than 3 stars) or negative (3 stars or lower). Reviews are encoded in 5,000 dimensional *tf-idf* feature vectors of bag-of-words unigrams and bigrams. From this data, we constructed 12 cross-domain binary classification tasks. Each domain adaptation task consists of 2,000 labeled source examples,

[1] https://www.cs.jhu.edu/ mdredze/datasets/sentiment/

S→T	Supervised Learning		Unsupervised Transfer		Semi-supervised Transfer			
	SO	ST	CMD	DSN	CMD-ft	DSN-ft	CODA	CoCMD (p-value)
B→D	81.7±0.2	81.6±0.4	82.6±0.3	82.8±0.4	82.7±0.1	82.7±0.6	81.9±0.4	**83.1**±0.1(.003)
B→E	74.0±0.6	75.8±0.2	81.5±0.6	81.9±0.5	82.4±0.6	82.3±0.8	77.5±2.0	**83.0**±0.6(.061)
B→K	76.4±1.0	78.2±0.6	84.4±0.3	84.4±0.6	84.7±0.5	84.8±0.9	80.4±0.8	**85.3**±0.7(.039)
D→B	79.5±0.3	80.0±0.4	80.7±0.6	80.1±1.3	81.0±0.7	81.1±1.2	80.6±0.3	**81.8**±0.5(.022)
D→E	75.6±0.7	77.0±0.3	82.2±0.5	81.4±1.1	82.5±0.7	81.3±1.2	79.4±0.7	**83.4**±0.6(.019)
D→K	79.5±0.4	80.4±0.6	84.8±0.2	83.3±0.7	84.5±0.9	83.8±0.8	82.4±0.5	**85.5**±0.8(.055)
E→B	72.3±1.5	74.7±0.4	74.9±0.6	75.1±0.4	76.2±0.6	76.3±1.4	73.6±0.7	**76.9**±0.6(.094)
E→D	74.2±0.6	75.4±0.4	77.4±0.3	77.1±0.3	77.7±0.7	77.1±1.1	75.9±0.2	**78.3**±0.1(.079)
E→K	85.6±0.6	85.7±0.7	86.4±0.9	87.2±0.7	86.7±0.3	87.1±0.9	86.1±0.4	**87.3**±0.4(.093)
K→B	73.1±0.1	73.8±0.3	75.8±0.3	76.4±0.5	76.4±0.5	76.2±0.3	74.3±1.0	**77.2**±0.4(.016)
K→D	75.2±0.7	76.6±0.9	77.7±0.4	78.0±1.4	78.8±0.4	78.5±0.5	77.5±0.4	**79.6**±0.5(.039)
K→E	85.4±1.0	85.3±1.6	86.7±0.6	86.7±0.7	**87.3**±0.3	87.2±0.4	86.4±0.5	87.2±0.4(.512)

Table 1: Average prediction accuracy with 5 runs on target domain testing data set. The left group of models refer to previous state-of-the-art methods and the right group of models refer to the proposed model and some of its variants. We list the p-values of the T-test between CoCMD and CMD-ft for more intuitive understanding.

2,000 unlabeled target examples, and 50 labeled target examples for training. To fine-tune the hyper-parameters, we randomly select 500 target examples as developing data set, leaving 2,500-5,500 examples for testing. All of the compared methods and CoCMD share this setting.

4.2 Compared Methods

CoCMD is systematically compared with: 1) neural network classifier without any domain adaptation trained on labeled source data only (SO); 2) neural network classifier without any domain adaptation trained on the union of labeled source and target data (ST); 3) unsupervised central moment discrepancy trained with labeled source data only (CMD) (Zellinger et al., 2017); 4) unsupervised domain separation network (DSN) (Bousmalis et al., 2016); 5) semi-supervised CMD trained on labeled source data and then fine-tuned on labeled target data (CMD-ft); 6) semi-supervised DSN trained on labeled source data and then fine-tuned on labeled target data (DSN-ft); 7) semi-supervised Co-training for domain adaptation (CODA) (Chen et al., 2011).

4.3 Implementation Detail

CoCMD was imeplented with a similar architecture to that of Ganin et al., (2016) and Zellinger et al., (2017), with one dense hidden layer with 50 hidden nodes and sigmoid activation functions. The classifiers consist of a softmax layer with two dimensional outputs. And the decoder was implemented with a multilayer perceptron (MLP) with one dense hidden layer, tanh activation functions, and relu output functions.

Model optimization was performed using the RmsProp (Tieleman and Hinton, 2012) update rule with learning rate set to 0.005 for all of the tasks.Hyper-parameter K of the CMD regularizer was set to 3 for all of the tasks, according to the experiment result of Zellinger et al. (2017). For the hyper-parameters α, β, γ, and λ, we took the values that achieve the best performance on the developing data set via a grid search $\{0.01, 0.1, 1, 10, 100\}$. However, instead of building grids on α, β, γ, and λ all at the same time, we first fine-tuned the values of α and β with the values of γ and λ fixed at 1. After that, we fine-tuned the values of γ and λ with α and β fixed at the best values obtained at last step. Though, this practice may miss the best combination of these hyper-parameters, it can greatly reduce the time consuming for fine-tuning and still obtain acceptable results. And for each iteration of the co-training, we set $p = n = 5$.

4.4 Result

Table 1 shows the average classification accuracy of our proposed model and the baselines over all 12 domain adaptation tasks. We can first observe that the proposed model CoCMD outperforms the compared methods over almost all of the

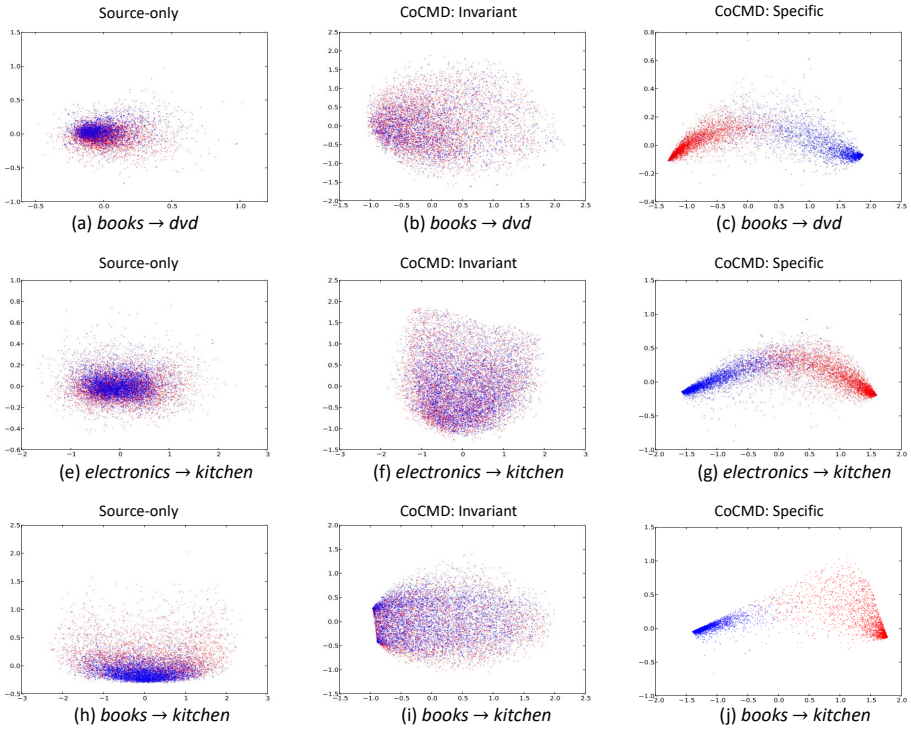

Figure 3: The distribution of source and target data in the hidden space of different representations. The red points denote the source examples and the blue ones denote the target examples. The pictures of each row correspond to the $B \rightarrow D$, $E \rightarrow K$, and $B \rightarrow K$ task. The pictures of each column correspond to the hidden space, H_c^e, of the source-only model, the domain invariant representation, and the target specific representation of the proposed model.

12 tasks except for the K→E task. And by comparing the results of CMD-based methods and DSN-based methods, we can find out that just extracting the domain specific information but not making further usage does not offer much improvement to the adaptation performance for sentiment classification task. This approves the necessary to explore the usage of domain specific information.

If organizing the domain B and D into a group and organizing the domain E and K into another group, we can observe that the domain adaptation methods achieve greater improvement on the standard classifiers over cross-group tasks (e.g., B → K) than over within-group tasks (e.g., B → D). Similar observation can also be observed by comparing ST with SO, CMD with CMD-ft, and DSN with DSN-ft. The possible explanation is that domains within the same group are more close. Thus adapting over within group tasks is easier than adapting over cross group tasks, if without any domain adaptation regularizer. In addition, we can also observe that CoCMD achieve relatively greater improvement on CMD baseline over the cross-group tasks that over the within-group tasks. We argue that this is because domains in the same group contain relatively less domain individual characteristic. While for domains cross the groups, the domain specific information usually takes a larger share of all of the information. Because the additional part of our proposed method compared to the CMD baseline, is built on the domain specific information, the improvement should be relatively less for within-group tasks. Further analysis of the proposed model in the next section empirically proves this explanation.

4.5 Model Analysis

In this section, we look into how similar two domains are to each other in the space of domain invariant representation and domain specific representation.

A-distance Study: Some of previous works proposed to make use of a proxy of the $\mathcal{A}$-distance (Ben-David et al., 2007) to measure

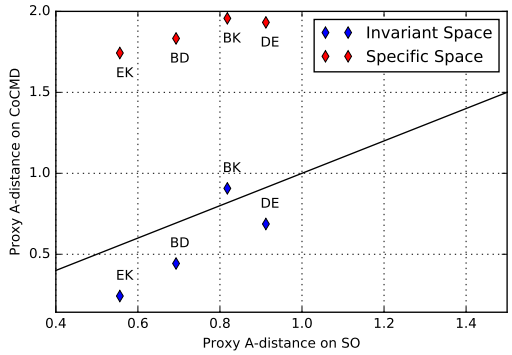

Figure 4: Proxy A-distance between domains of the Amazon benchmark for the 4 different tasks.

the distance of two domains. The proxy was defined by $2(1-2\epsilon)$, where ϵ is the generalization error of a linear SVM classifier trained on the binary classification problem to distinguish inputs between the source and target domains. Figure 4 shows the results of each pair of domains. We observe several trends: Firstly, the proxy A-distance of within-group domain pairs (i.e., BD and EK) is consistently smaller than that of the cross-group domain pairs (i.e., BK and DE) on all of the hidden spaces. Secondly, the proxy A-distance on the domain specific space is consistently larger than its corresponding value on the hidden space of SO model, as expected. While the proxy A-distance value on domain invariant space is generally smaller than its corresponding value on the hidden space of SO model, except for BK domain pair. A possible explanation is that the balance of classification loss and domain discrepancy loss makes there is still some target domain specific information in the domain invariant space, introduced by the target unlabeled data.

Visualization: For more intuitive understanding of the behaviour of the proposed model, we further perform a visualization of the domain invariant representation and the domain specific representation, respectively. For this purpose, we reduce the dimension of the hidden space to 2 using principle component analysis (PCA) (Wold et al., 1987). Due to space constraints we choose three tasks: two within-group tasks ($B \rightarrow D$ and $E \rightarrow K$) and a cross-group task ($B \rightarrow K$). For comparison, we also display the distribution of each domain in the hidden space of the SO model. The results are shown in Figure 3.

Pictures of the first column in Figure 3 show the original distribution of the source and target examples in the hidden space of SO model. As can be seen, there is a great overlap between the distributions of the domain *B* and the domain *D* domains and between the distributions of the domain *E* and the domain *K*. While there is quite a gap between the distribution of the domain *B* and the domain *K*. This strengthens our argument that within-group domains share relatively more common information than cross-group domains. Pictures of the second column show the distribution of the source and target examples in the domain invariant hidden space of the proposed model. From these pictures we can see that the distributions of the source and target data are quite similar in this presentation. This demonstrates the effectiveness of the CMD regularizer for extracting domain invariant representation. Pictures of the third column show the distribution of the source and target examples in the domain specific hidden space of the proposed model. As can be seen from these pictures, examples of the source and target domains are separated very well. This demonstrates the effectiveness of our proposed method for extracting domain specific information.

5 Conclusion

In this work, we investigated the importance of domain specific information for domain adaptation. In contrast with most of the previous methods, which pay more attention to domain invariant information, we showed that domain specific information could also be beneficially used in the domain adaptation task with a small amount of in-domain labeled data. Specifically, we proposed a novel method, based on the CMD metric, to simultaneously extract domain invariant feature and domain specific feature for target domain data. With these two different features, we performed co-training with labeled data from the source domain and a small amount of labeled data from the target domain. Sentiment analysis experiments demonstrated the effectiveness of this method.

6 Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural

Science Foundation of China (No. 61751201, 61532011, 61473092, and 61472088), and STCSM (No.16JC1420401,17JC1420200).

References

Rahaf Aljundi, Rémi Emonet, Damien Muselet, and Marc Sebban. 2015. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 56–63.

Jiashi Feng Kate Saenko Baochen Sun. 2015. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*. pages 137–144.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. volume 7, pages 440–447.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics* 22(14):e49–e57.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 343–351. http://papers.nips.cc/paper/6254-domain-separation-networks.pdf.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *Advances in neural information processing systems*. pages 2456–2464.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. 2013. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2960–2967.

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

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

Judy Hoffman, Erik Rodner, Jeff Donahue, Brian Kulis, and Kate Saenko. 2014. Asymmetric and category invariant feature transformations for domain adaptation. *International journal of computer vision* 109(1-2):28–41.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Mingsheng Long, Jianmin Wang, Jiaguang Sun, and S Yu Philip. 2015. Domain invariant transfer kernel learning. *IEEE Transactions on Knowledge and Data Engineering* 27(6):1519–1532.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2015. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830* .

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

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2(1-3):37–52.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811* .