DaMSTF: Domain Adversarial Learning Enhanced Meta Self-Training for Domain Adaptation

Anonymous ACL submission

Abstract

Self-training emerges as an important research line on domain adaptation. By taking the model's prediction as the pseudo labels of the unlabeled data, self-training bootstraps the model with pseudo instances in the target domain. However, the prediction errors of pseudo labels (label noise) challenge the performance of self-training. To address this problem, previous approaches only use reliable pseudo instances, i.e., pseudo instances with high prediction confidence, to retrain the model. Although these strategies effectively reduce the label noise, they are prone to miss the hard examples. In this paper, we propose a new self-training framework for domain adaptation, namely Domain adversarial learning enhanced Self-Training Framework (DaMSTF). Firstly, DaMSTF involves meta-learning to estimate the importance of each pseudo instance, so as to simultaneously reduce the label noise and preserve hard examples. Secondly, we design a meta constructor for constructing the meta validation set, which guarantees the effectiveness of the meta-learning module by improving the quality of the meta validation set. Thirdly, we find that the meta-learning module suffers from the training guidance vanishment and tends to converge to an inferior optimal. To this end, we employ domain adversarial learning as a heuristic neural network initialization method, which can help the meta-learning module converge to a better optimal. Theoretically and experimentally, we demonstrate the effectiveness of the proposed DaMSTF. On the cross-domain sentiment classification task, DaMSTF improves the performance of BERT with an average of nearly 4%.

1 Introduction

Domain adaptation, which aims to adapt the model trained on the source domain to the target domain, attracts much attention in Natural Language Processing (NLP) applications(Du et al., 2020; Chen et al., 2021; Lu et al., 2022). Since domain adaptation involves labeled data from the source domain and unlabeled data from the target domain, it can be regarded as a semi-supervised learning problem. From this perspective, self-training, a classical semi-supervised learning approach, emerges a prospective research direction on domain adaptation (Zou et al., 2019; Liu et al., 2021).

Self-training consists of a series of loops over the pseudo labeling phase and model retraining phase. In the pseudo labeling phase, self-training takes the model's prediction as the pseudo labels for the unlabeled data from the target domain. Based on these pseudo-labeled instances, self-training retrains the current model in the model retraining phase. The trained model can be adapted to the target domain by repeating these two phases. Due to the prediction errors, there exists label noise in pseudo instances, which challenges self-training approaches (Zhang et al., 2017).

Previous self-training approaches usually involve a data selection process to reduce the label noise, i.e., preserving the reliable pseudo instances and discarding the remaining ones. In general, higher prediction confidence implies higher prediction correctness, so existing self-training approaches prefer the pseudo instances with high prediction confidence (Zou et al., 2019; Shin et al., 2020). However, fitting the model on these easy pseudo instances cannot effectively improve the model, as the model is already confident about its prediction. On the contrary, pseudo instances with low prediction confidence can provide more information for improving the model, but contain more label noise at the same time.

To simultaneously reduce the label noise and preserve hard examples, we propose to involve in meta-learning to reweight pseudo instances. Within a learning-to-learn schema, the meta-learning module learns to estimate the importance of every pseudo instance, and then, allocates different in-

stance weights to different pseudo instances. Ideally, hard and correct pseudo instances will be assigned larger weights, while easy or error pseudo instances will be assigned smaller weights. To achieve this, the process in the meta-learning module is formulated as a bi-level hyperparameters optimization problem (Franceschi et al., 2018), where instance weights are taken as the hyperparameters and determined by a series of meta-training steps and meta-validation steps. In the meta-training step, the model is virtually updated on the meta-training set with respect to the current instance weights. In the meta validation step, we validate the virtually updated model with an unbiased meta validation set, and optimize the instance weights with the training guidance back-propagated from the validation performance.

According to the analysis in (Ren et al., 2018), a high-quality meta validation set, which is clean and unbiased to the test set, is important for the effectiveness of the meta-learning algorithm. To this end, we propose a meta constructor oriented to the domain adaptation scenario. At each self-training iteration, the meta constructor selects out the most reliable pseudo instances and inserts them into the meta validation set. Since the instances in the meta validation set are all from the target domain and vary along with the self-training iterations, the data distribution in the constructed meta validation set approximates the one in the target domain. Thus, the meta constructor reduces the bias of the meta validation set. On the other hand, selecting the most reliable pseudo instances can reduce the label noise, making the meta validation set cleaner.

Another challenge for the meta-learning module is the training guidance vanishment, referring to the gradient vanishment on hyperparameters. With a theoretical analysis, we attribute this problem to the gradient vanishment on the meta validation set. To this end, we introduce a domain adversarial learning module to perturb the model's parameters, thereby increasing the model's gradients on the meta validation set. In DaMSTF, we also interpret the domain adversarial learning module as a heuristic neural network initialization method. Before the model retraining phase, the domain adversarial learning module first initializes the model's parameters by aligning the model's feature space. For domain adaptation, the global optimal refers to the state where the model's parameters are agnostic to the domain information but discriminative to the task information. Thus, the training process in the domain adversarial learning module makes the model's parameters closer to the global optimal, serving as a heuristic neural network initialization.

Our contributions can be summarized as follows:

- We propose a new self-training framework to realize domain adaptation, named Domain adversarial learning enhanced Meta Self Training Framework (DaMSTF), which involves meta-learning to simultaneously reduce the label noise and preserve hard examples.
- We propose a meta constructor to construct the meta validation set, which guarantees the effectiveness of the meta-learning module.
- We theoretically point out the training guidance vanishment problem in the meta-learning module and propose to address this problem with a domain adversarial learning module.
- Theoretically, We analyze the effectiveness of the DaMSTF in achieving domain adaptation. Experimentally, we validate the DaMSTF on two popular models, i.e., BERT for the sentiment analysis task and BiGCN for the rumor detection task, with four benchmark datasets.

2 Problem Formulation

We denote the set that involves all instances in the source domain as $\mathbb{D}_S$, and denote the set that contains all instances in the target domain as $\mathbb{D}_T$. From $\mathbb{D}_S$, we can obtain a labeled dataset for training, i.e., $D_S = \{(x_i, y_i)\}_{i=1}^{N}$. In text classification tasks, the input x_i is a text from the input space $\mathcal{X}$, the corresponding label y_i is a C-dimensional one-hot label vector, i.e., $y_i \in \{0,1\}^C$, where C is the number of classes. Based on D_S, we learn a hypothesis, $h : \mathcal{X} \rightarrow \{0,1\}^C$. Since D_S comes from $\mathbb{D}_S$ (i.e., $D_S \subseteq \mathbb{D}_S$), the learned hypothesis h usually performs well on $\mathbb{D}_S$. When we transfer the hypothesis h from $\mathbb{D}_S$ to $\mathbb{D}_T$, h may perform poorly due to the domain shift. The goal of *domain adaptation* is to adapt the hypothesis h to $\mathbb{D}_T$.

In general, unlabeled text in the target domain is available (Gururangan et al., 2020). We denote the unlabeled target domain dataset as $D_T^u = \{(x_m)\}_{m=1}^{U}$, where $x_m \in \mathcal{X}$ is a text input. In some cases, we can even access an *in-domain dataset*, i.e., a small set of labeled data in the target domain, which is denoted as $D_T^l = \{(x_j, y_j)\}_{j=1}^{L}$ ($x_i \in \mathcal{X}$ and $y_i \in \{0,1\}^C$). When $D_T^l = \emptyset$, the task is a case of *unsupervised domain adaptation* (Wilson and Cook, 2020). Otherwise, the

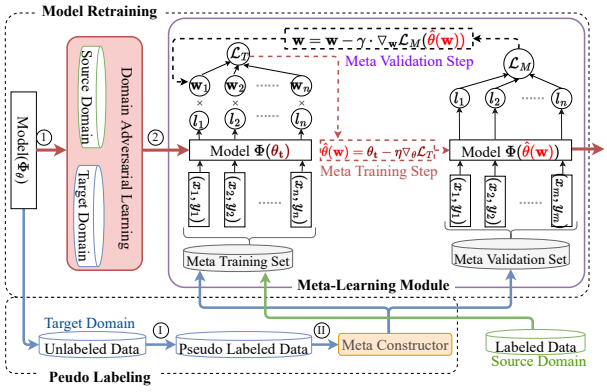

Figure 1: An overview of the DaMSTF. Red arrows indicate the training process of the model, while blue and green arrows indicate the data flow.

task is a case of *semi-supervised domain adaptation* (Saito et al., 2019).

3 Methodology

3.1 Model Overview

DaMSTF inherits the basic framework of self-training, which consists of iterations over the "Pseudo Labeling" phase and the "Model Retraining" phase. To achieve domain adaptation, self-training simultaneously optimizes the model's parameters and the pseudo labels with Eq. (1).

$$\min_{\theta,\hat{\mathbf{Y}}_T} \mathcal{L}_{st}(\theta,\hat{\mathbf{Y}}_T) = \sum_{(x_k,y_k)\in D_S} \mathcal{E}(\Phi(x_k;\theta),y_k) + \sum_{x_i\in D_T^u} \mathcal{E}(\Phi(x_i;\theta),\hat{y}(x_i)) \quad (1)$$

where $\hat{\mathbf{Y}}_T = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|D_T^u|}]^T$ denotes the pseudo label set of the unlabeled target domain data, Φ_θ denotes the model under the hypothesis (h), and θ denotes the model's parameters.

In the pseudo labeling phase, DaMSTF predicts the unlabeled data in the target domain, and the predictions are taken as pseudo labels. Then, these pseudo instances are sent to the meta constructor. For the instances with high prediction confidence, the meta constructor uses them to expand the meta validation set. For the remaining ones, the meta constructor uses them to construct the meta-training set.

In the model retraining phase, DaMSTF first trains the model in the domain adversarial training module to align the feature space. Then, the model is trained in the meta-learning module. Afterward, DaMSTF backs to the pseudo labeling phase to start another self-training iteration.

Fig. 1 shows the structure of DaMSTF, and Algorithm 1 presents the corresponding pseudo-code.

Algorithm 1 DaMSTF

Require: labeled source dataset D_S, unlabeled target dataset D_T^u, in-domain dataset D_T^l
Pretrain θ on D_S, $D_M \leftarrow D_T^l$
while the termination criteria is not met **do**
  Compute pseudo label $\hat{\mathbf{Y}}_T$ on D_T^u
  $H = -\hat{\mathbf{Y}}_T * log(\hat{\mathbf{Y}}_T)$
  Sort the D_T^p with respect to H in ascending order, and denote the first $\mathcal{K}$ data as D_E, the remaining data as D_T^{tr}
  $D_M = D_T^l \cup D_E$
  DomainAdversarial($D_S \cup D_T^u, \theta_F, \vartheta$)
  MetaLearning($D_S \cup D_T^{tr}, \theta, \mathbf{w}$)
end while
function MetaLearning($D, \theta, \mathbf{w}$)
  for training batch $\mathcal{B}$ in D **do**
    for t=1 $\rightarrow \mathcal{T}_M$ **do**
      Compute $\hat{\theta}(\mathbf{w}^t)$ via Eq. (3)
      Compute weight $\mathbf{w}^{t+1}$ via Eq. (7)
    end for
    $\mathbf{w}^* \leftarrow \mathbf{w}^{\mathcal{T}_M}$, update θ with $Eq.$ (8)
  end for
  return θ, $\mathbf{w}$
end function
function DomainAdversarial(D, θ_F, ϑ)
  for training batch $\mathcal{B}$ in D **do**
    for t=1 $\rightarrow \mathcal{T}_D$ **do**
      $\vartheta = \vartheta - \eta_1 \nabla_\vartheta \mathcal{L}_{DA}(\theta_F, \vartheta, \mathcal{B})$
    end for
    for t=1 $\rightarrow \mathcal{T}_G$ **do**
      $\theta_F = \theta_F + \eta_2 \nabla_\theta \mathcal{L}_{DA}(\theta_F, \vartheta, \mathcal{B})$
    end for
  end for
  return θ, ϑ
end function

3.2 Meta-Learning Module

As described in Fig. 1, the meta-learning module involves a series of loops over the "Meta Training" step and "Meta Validation" step to optimize the hyper-parameters and the model parameters.

Meta Training. The training batch in the meta training phase, i.e., $\mathcal{B} = \{(x_1,y_1),(x_2,y_2),\ldots\}$, merges the labeled data from the source domain with the pseudo labeled data from the target domain. The supervision on the pseudo instances is the pseudo-label, and the supervision on the labeled instances is the ground-truth label. We compute the risk loss on the training batch with Eq. (2):

$$\mathcal{L}_T(\theta,\mathbf{w}^t,\mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{x_i,y_i\in\mathcal{B}} \mathbf{w}_i^t \mathcal{E}(\Phi(x_i;\theta),y_i) \quad (2)$$

where $|\mathcal{B}|$ is the size of $\mathcal{B}$, $\mathcal{E}$ is the loss function, $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{|\mathcal{B}|}$ are the extra hyperparameters introduced in the meta-learning module, i.e., a set of instance weights indicating the importance of each training example. In the meta training step, we derive a virtual update on the model with Eq. (3):

$$\hat{\theta}(\mathbf{w}^t) = \theta - \eta \nabla_\theta \mathcal{L}_T(\theta,\mathbf{w}^t,\mathcal{B}) \quad (3)$$

where η is the learning rate.

Meta Validation After being virtually updated in the meta training phase, the model is validated on the meta validation set D_M with Eq. (4):

$$\mathcal{L}_M(\hat{\theta}(\mathbf{w}^t)) = \frac{1}{|D_M|} \cdot \sum_{x_j, y_j \in D_M} \mathcal{E}(\Phi(x_j; \hat{\theta}(\mathbf{w}^t)), y_j) \quad (4)$$

where $\mathcal{E}$ is the loss function, $|D_M|$ is the size of the meta validation set. By backpropagating the performance on the meta validation set, we derive the *training guidance* for updating the instance weights on the training batch as below:

$$\frac{\partial \mathcal{L}_M(\hat{\theta}(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}_M(\hat{\theta}(\mathbf{w}))}{\partial \hat{\theta}(\mathbf{w})} \cdot \frac{\partial \hat{\theta}(\mathbf{w})}{\partial \mathbf{w}} \quad (5)$$

To reduce the computation cost, we take the method in (Liu et al., 2018; Chen et al., 2021) to approximate the *training guidance*, as below:

$$\frac{\partial \mathcal{L}_M(\hat{\theta}(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\nabla_{\mathbf{w}} \mathcal{L}_T(\theta^+, \mathbf{w}^t, \mathcal{B}) - \nabla_{\mathbf{w}} \mathcal{L}_T(\theta^-, \mathbf{w}^t, \mathcal{B})}{2\epsilon} \quad (6)$$

where ϵ is a small scalar coefficient for approximation, θ^+ and θ^- are computed by the following:

$$\theta^+ = \theta + \epsilon \cdot \frac{\partial \mathcal{L}_M(\hat{\theta}(\mathbf{w}))}{\partial \hat{\theta}(\mathbf{w})}, \quad \theta^- = \theta - \epsilon \cdot \frac{\partial \mathcal{L}_M(\hat{\theta}(\mathbf{w}))}{\partial \hat{\theta}(\mathbf{w})}$$

Based on the computed training guidance, we obtain the optimal instance weights (marked as $\mathbf{w}^*$) with gradient descent algorithm, as described in Eq. (7). Further, we update θ with Eq. (8):

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \gamma \cdot \frac{\partial \mathcal{L}_M(\hat{\theta}(\mathbf{w}))}{\partial \mathbf{w}} \quad (7)$$

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta \mathcal{L}_T(\theta, \mathbf{w}^*, \mathcal{B}) \quad (8)$$

After the above process is completed on the training batch $\mathcal{B}$, another training batch will be selected to start the meta-learning phase again, as shown in lines 15-21 in Algorithm 1.

3.3 Meta Constructor

In previous studies, the meta validation set is constructed by collecting a set of labeled data that have the same distribution as the test set (Ren et al., 2018; Shu et al., 2019). However, such practice is not acceptable in domain adaptation, as we are not aware of the data distribution of the target domain during the training phase.

To this end, we propose a meta constructor to construct a meta validation set that approximates the target domain. Specifically, we select the reliable instances from the pseudo-labeled data as the instances in the meta validation set. To evaluate the reliability of each of the pseudo instances, we compute their prediction entropy via Eq. (9):

$$H(x_i) = -\sum_{c=1}^{C} (\Phi(c|x_i; \theta) \cdot log(\Phi(c|x_i; \theta))) \quad (9)$$

where $\Phi(c|x_i; \theta)$ is the probability of the instance x_i belongs to the c_{th} category.

In general, a lower prediction entropy indicates a higher prediction correctness (Nguyen et al., 2020). Thus, we first sort the D_T^p (pseudo labeled dataset) in ascending order according to their prediction entropy. Then, the top-ranked $\mathcal{K}$ instances, denoted as D_E, are selected as the validation instances, and the remaining pseudo samples, denoted as D_T^{tr}, are preserved in the meta training set.

In the semi-supervised domain adaptation, we take the in-domain dataset to initialize the meta validation dataset and use D_E to expand the meta validation set along with the self-training iterations. In the unsupervised domain adaptation, where the in-domain dataset is empty, we directly take D_E as the meta validation set. The above process is detailed in lines 2-8 of Algorithm 1.

Here, meta constructor is an important knot that combines meta-learning and self-training. On the one hand, traditional machine learning approaches cannot exploit the pseudo instances with high prediction entropy, due to the inherent label noise. In this case, the meta constructor uses them to construct the meta training set, as the meta-learning module is tolerant to the label noise in the meta-training set. On the other hand, pseudo instances with low prediction entropy cannot provide extra information for improving the model but contain less label noise. In this case, the meta constructor uses them to validate the model, i.e., uses them to construct or expand the meta validation set, which can improve the quality of the meta validation set.

3.4 Domain Adversarial Learning

As theoretically explained in § 4.1, the training guidance would not be indicative if the model's gradient on the validation instance is negligible. The presence of domain adversarial learning can prevent the gradient vanishment on the meta validation set, thereby preventing the training guidance vanishment. On the other hand, domain adversarial learning can explicitly align the feature space along with the self-training iterations.

To present the details in the domain adversarial learning module, we divide the model $\Phi(\bullet; \theta)$ into two parts: the feature extraction layer $\Phi_F(\bullet; \theta_F)$ and the task-specific layer $\Phi_c(\bullet; \theta_c)$. Usually, θ_c is the parameters of the last layer in the model, whose output is the prediction probability of each

category. The prediction process in the model is:

$$\Phi(x_i;\theta) = \Phi_c(\Phi_F(x_i;\theta_F);\theta_c) \quad (10)$$

Following Ganin et al. (2016), we introduce an extra domain discriminator to discriminate the instances' domains, i.e., $\varphi(\bullet;\vartheta)$, where ϑ is the parameters. On a training batch $\mathcal{B}$, the risk loss for domain adversarial learning is:

$$\mathcal{L}_{DA}(\theta_F,\vartheta,\mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{x_i,d_i\in\mathcal{B}} \mathcal{E}(\varphi(\Phi_F(x_i;\theta_F);\vartheta),d_i) \quad (11)$$

where d_i is a one-hot vector representing the domain of x_i, $\mathcal{E}$ is the cross-entropy function. The specific training process of the proposed domain adversarial learning module is depicted in Algorithm 1, lines 25-35.

4 Theoretical Analysis

This section first introduces the training guidance vanishment problem and then explains the effectiveness of DaMSTF in achieving domain adaptation. The proofs of Theorem 1 and Theorem 2 are detailed in Appendix. A and Appendix. B.

4.1 Training Guidance Vanishment

Theorem 1. *Let $\mathbf{w}_i$ be the weight of the training instance i, denoted as (x_i, y_i), in $\mathcal{B}$, the gradient of $\mathbf{w}_i$ on $\mathcal{L}_M$ can be represented by the similarity between the gradients on training instance i and the gradients on the meta validation set:*

$$\frac{\partial L_M(\hat{\theta}(\mathbf{w}))}{\partial \mathbf{w}_i} = -\frac{\eta}{|\mathcal{B}|}\cdot[\frac{1}{|D_M|}\sum_{j=1}^{|D_M|}\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j)^T]\cdot\vec{\mathbf{g}}_{\theta}(x_i,y_i)$$

where $\frac{1}{|D_M|}\sum_{j=1}^{|D_M|}\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j)^T$ is the gradients of $\hat{\theta}$ on D_M, $\vec{\mathbf{g}}^i_{\theta}(x_i,y_i)$ is the gradients of θ on the training instance i, η is the learning rate in Eq. (3)

According to Theorem 1, $\frac{\partial L_M(\hat{\theta}(\mathbf{w}))}{\partial \mathbf{w}_i}$ is not indicative for every training instance if the model's gradient on the meta validation set (i.e., $\frac{1}{|D_M|}\sum_{j=1}^{|D_M|}\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j)$) is very small, which we named as the *training guidance vanishment* problem. In DaMSTF, the meta-learning module is challenged by the training guidance vanishment problem from the following aspects.

Firstly, the meta validation set is much smaller than the meta training set, so the model converges faster on the meta validation set than that on the meta training set. Considering the optimization on neural networks is non-convex, the model can converge to an inferior optimal if it converges too early on the meta validation set. In this case, the model's gradient on the meta validation set is very small, which results in the training guidance vanishment.

Secondly, the instances in D_E are the ones with small prediction entropy. Since the supervision for the pseudo instances is exactly the model's predictions, lower prediction entropy results in lower risk loss. Then, the gradients back-propagated from the risk loss are negligible, which also results in the training guidance vanishment.

4.2 Theoretical Explanation of DaMSTF

The *disagreement* and $H\Delta H$-distance were first proposed in Ben-David et al. (2010) and have been widely applied to analyze the effectiveness of domain adaptation approaches (Saito et al., 2019; Du et al., 2020). For any two different hypotheses h_1 and h_2, *disagreement* $\epsilon_D(h_1,h_2)$ quantifies the discrepancy of their different predictions on a specific dataset D. When h_2 is an ideal hypothesis that can correctly map all instances in D, $\epsilon_D(h_1,h_2)$ also represents the *error rate* of the hypothesis h_1 on dataset D, abbreviated as $\epsilon_D(h_1)$. $H\Delta H$-distance is a metric for evaluating the divergence of the data distribution between two datasets, which is only relevant to the input space of the datasets.

Based on the *disagreement* and $H\Delta H$-distance, we conclude Theorem 2 for the proposed DaMSTF.

Theorem 2. *Assume there exists an ideal hypothesis, denoted as h^*, which correctly maps all instances in the target domain to their groud-truth labels. In the self-training iteration t, let $\epsilon_{D^l_T}(h^t)$ and $\epsilon_{D_E}(h^t)$ be the error rate of the hypothesis h^t on D^l_T and D_E, respectively. Then, the error rate of the hypothesis h^t on the target domain is upper bounded by:*

$$\epsilon_{\mathbb{D}_T}(h^t) \le \epsilon_{D^l_T\cup D_E}(h^t) + \frac{1}{2}d_{H\Delta H}(\mathbb{D}_T, D^l_T\cup D_E) + \rho\cdot\epsilon_{D_E}(h^*,h^{t-1})$$

where $\rho = \frac{|D_E|}{|D^l_T|+|D_E|}$ is a coefficient related to the size of D^l_T and D_E, $\epsilon_{D^l_T\cup D_E}(h^t)$ is the error rate of the hypothesis h^t on the union of D^l_T and D_E.

Based on Theorem 2, we demonstrate the effectiveness of DaMSTF from the following aspects.

Firstly, expanding the meta validation set can decrease the second term in Theorem 2, i.e., $\frac{1}{2}d_{H\Delta H}(\mathbb{D}_T, D^l_T\cup D_E)$, thereby reducing the upper bound of $\epsilon_{\mathbb{D}_T}(h^t)$. Since $H\Delta H$-distance is irrelevant to the label, the annotation errors in D_E have no effect on computing the $H\Delta H$-distance between D_E and other datasets, so $d_{H\Delta H}(\mathbb{D}_T, D^l_T\cup D_E)$ is smaller than $d_{H\Delta H}(\mathbb{D}_T, D^l_T)$. Furthermore, as D_E varies in each self-training iteration, the DaMSTF can leverage the diversity of

the unlabeled data in the target domain. Thus, $d_{H\Delta H}(\mathbb{D}_T, D_T^l \cup D_E)$ is close to $d_{H\Delta H}(\mathbb{D}_T, D_T^u)$ in the whole training process.

Secondly, by selecting examples that have the lowest prediction entropy, the error rate on D_E is much lower than that of the expected error rates on D_T^p, formally, $\epsilon_{D_E}(h^*, h^{t-1}) < \epsilon_{D_T^p}(h^*, h^{t-1})$. In other words, the data selection process in the meta constructor reduces the third term in Theorem 2,i.e., $\rho \cdot \epsilon_{D_E}(h^*, h^{t-1})$.

5 Experiments

To validate the effectiveness of the DaMSTF, we conduct experiments on two popular models: BERT (Devlin et al., 2019) and BiGCN (Bian et al., 2020). BERT is a pre-trained language model based on the transformer, which has achieved great success on NLP tasks. BiGCN is a neural network based on GCN (Graph Convolution Network (Kipf and Welling, 2017)), which is an important baseline for rumor detection tasks.

Dataset On the rumor detection task, we conduct experiments with the dataset TWITTER (Zubiaga et al., 2016). The instances in the TWITTER dataset are collected with five topics, "Charlie Hebdo#", "Germanwings Crash#", "Ferguson#", "Ottawa Shooting", and "Sydney Siege" (abbreviated as "Cha.", "Ger.", "Fer.", "Ott.", and "Syd."). Thus, we categorized the instances into five domains. On the sentiment classification task, we conduct experiments with the dataset Amazon (Blitzer et al., 2007). We follow the method in (He et al., 2018) to preprocess the Amazon dataset, and the resultant dataset consists of 8,000 instances from four domains: books, dvd, electronics, and kitchen. More statistics about the TWITTER dataset and the Amazon dataset can be found in Appendix D.

Comparing Methods Since the DaMSTF can be customized to both semi-supervised and unsupervised domain adaptation scenarios, the baselines contain both unsupervised and semi-supervised domain adaptation approaches. For the unsupervised domain adaptation, Out (Chen et al., 2021), DANN (Ganin et al., 2016) and CRST (Zou et al., 2019) are selected as the baselines, while In+Out (Chen et al., 2021), MME (Saito et al., 2019), BiAT (Jiang et al., 2020), and Wind (Chen et al., 2021) are selected as the baselines for the semi-supervised domain adaptation. Out and In+Out are two straightforward ways for realizing unsupervised and semi-supervised domain adaptation, where Out means the base model is trained on the out-of-domain data (i.e., labeled source domain data) and In+Out means the base model is trained on both the in-domain and the out-of-domain data. The core of DANN is an adversarial learning algorithm that takes the domain classification loss as an auxiliary loss. CRST is also a self-training method that uses a label regularization technique to reduce the label noise from mislabeled data. WIND is a meta-learning-based domain adaptation approach that optimizes the weights of different training instances. The difference between the WIND and DaMSTF lies in that, (i) WIND only use the labeled source data to construct the meta training set, while the meta training set in the DaMSTF contains both the labeled data from the source domain and the pseudo data from the target domain. (ii) WIND does not consider the training guidance vanishment problem and the bias between the test set (i.e., target domain) and the meta validation set.

Experiment Setup On the benchmark datasets, we conduct domain adaptation experiments on every domain. When one domain is taken as the target domain for evaluation, the rest domains are merged as the source domain. The unlabeled data from the target domain are used for training the model, and the labeled data from the target domain are used for testing and validating the model (with a ratio of 7:3). Notes that the TWITTER dataset does not contain extra unlabeled data, we take 70% of the labeled data on the target domain as the unlabeled data for training the model, and the rest will be preserved for testing and validating the model. The experiments on TWITTER are conducted on "Cha.", "Fer.", "Ott.", and "Syd."[1]. More impelementation details are provided in Appendix C and the codes are available in `https://github.com/anonymous/XXX.git`.

5.1 Results

To validate the effectiveness of the meta self-training, we conduct unsupervised and semi-supervised domain adaptation experiments on two benchmark datasets, i.e., BiGCN on TWITTER, and BERT on Amazon. Since the rumor detection task focuses more on the 'rumor' category, we evaluate different models by their F1 score in classifying the 'rumor' category. On the sentiment classification task, the prediction accuracy of dif-

[1]The labeled data in "Ger." domain is too scare to provide extra unlabeled data.

Table 1: F1 score on the TWITTER and WEIBO

Base Model (BiGCN)		Unsupervised domain adaptation				Semi-Supervised domain adaptation				
		Out	DANN	CRST	DaMSTF	In+Out	MME	BiAT	Wind	DaMSTF
TWITTER	Cha.	*0.561*	0.501	0.563	**0.635**	*0.586*	0.601	0.547	0.552	**0.649**
	Fer.	*0.190*	0.387	0.446	**0.524**	*0.200*	0.081	0.256	0.291	**0.629**
	Ott.	*0.575*	0.544	0.709	**0.753**	*0.599*	0.612	0.614	0.633	**0.843**
	Syd.	*0.438*	0.461	0.673	**0.717**	*0.424*	0.677	0.661	0.628	**0.731**
	Mean	*0.441*	0.473	0.598	**0.657**	*0.452*	0.493	0.520	0.526	**0.714**

Table 2: Macro-F1 score on the Amazon dataset

Base Model (BERT)	Unsupervised Domain Adaptation				Semi-Supervised Domain Adaptation				
	Out	DANN	CRST	DaMSTF	In+Out	MME	BiAT	Wind	DaMSTF
books	*0.882*	0.887	0.878	**0.931**	*0.890*	0.896	0.891	0.890	**0.947**
dvd	*0.831*	0.864	0.845	**0.917**	*0.882*	0.893	0.888	0.904	**0.935**
electronics	*0.871*	0.914	0.877	**0.925**	*0.918*	0.906	0.926	0.917	**0.941**
kitchen	*0.863*	0.922	0.868	**0.927**	*0.925*	0.93	0.934	0.933	**0.947**
Mean	*0.862*	0.897	0.867	**0.925**	0.904	0.906	0.910	0.911	**0.942**

ferent classes is equally important, so we take the macro-F1 score to evaluate different models. For semi-supervised domain adaptation, 100 labeled instances in the target domain are taken as the in-domain dataset. The experiment results are listed in Tab. 1, Tab. 2.

As shown in Tab. 1, Tab. 2, DaMSTF outperforms all baseline approaches on all benchmark datasets. On the rumor detection task, DaMSTF surpasses the best baseline approaches (CRST for unsupervised domain adaptation, WIND for semi-supervised domain adaptation) by nearly 5% on average. For the "Fer." domain, where most approaches perform worse than the Out and In+Out, DaMSTF still achieves an F1 value of 0.629, which is 40% higher than that of the In+Out. On the sentiment classification task, DaMSTF also outperforms other approaches. Under the unsupervised domain adaptation scenario, DaMSTF surpasses the best baseline approach (DANN on the Amazon dataset) by nearly 2% on average. Under the semi-supervised domain adaptation scenario, DaMSTF surpasses Wind, the best baseline approach on the Amazon dataset, by nearly 3% on average.

5.2 Ablation Study

This subsection presents an ablation study to understand the effectiveness of the DaMSTF. As illustrated in § 3 and § 4.2, DaMSTF combines meta-learning and self-training via two strategies: (i) expanding the meta validation set with a meta constructor; (ii) preventing the training guidance vanishment problem with a domain adversarial module. Thus, we separately remove the above strategies from the DaMSTF, yielding three different variants, namely DaMSTF - *w/o E*, DaMSTF - *w/o D*, and DaMSTF - *w/o D, E*. Compared with DaMSTF, DaMSTF - *w/o E* does not select examples to expand the meta validation set, which means all pseudo instances are preserved to the meta training set. DaMSTF - *w/o D* removes the domain adversarial module from the DaMSTF. DaMSTF - *w/o D, E* removes both two strategies. Other experiment settings are the same as § 5.1. We summarize the results in Tab. 3, Tab. 4.

Table 3: Ablation Study on TWITTER

Domain	Cha.	Fer.	Ott.	Syd.	Mean
DaMSTF	0.649	0.629	0.843	0.731	0.713
- *w/o D*	0.585	0.401	0.782	0.724	0.623
- *w/o E*	0.600	0.542	0.694	0.685	0.630
- *w/o D, E*	0.569	0.352	0.633	0.631	0.547

Table 4: Ablation Study on the Amazon dataset

Domain	books	dvd	electronics	kitchen	Mean
DaMSTF	0.947	0.935	0.941	0.947	0.942
- *w/o D*	0.899	0.917	0.924	0.935	0.918
- *w/o E*	0.917	0.929	0.934	0.945	0.931
- *w/o D, E*	0.887	0.896	0.919	0.931	0.908

As shown in Tab. 3 and Tab. 4, both strategies are indispensable for the effectiveness of DaMSTF, and removing either strategy can result in performance degeneration. Removing the domain adversarial learning module (DaMSTF - *w/o D*) leads to an average decrease from 0.713 to 0.623 on the TWITTER dataset and from 0.942 to 0.918 on the Amazon dataset. Without expanding the meta validation set, DaMSTF - *w/o E* performs worse than DaMSTF on both the TWITTER dataset (0.630 vs. 0.731 on average) and the Amazon dataset(0.931 vs. 0.942 on average). After removing both strategies, DaMSTF suffers a severe performance deterioration on both benchmark datasets.

5.3 Effect of the unlabeled dataset size

As illustrated in § 4.2, the second term $d_{H\Delta H}(\mathbb{D}_T, D_T^l \cup D_E)$ is close to $d_{H\Delta H}(\mathbb{D}_T, D_T^u)$ in the whole training process. From this perspective, increasing the size of the unlabeled dataset can improve the performance. To validate this, we separately expose 0%, 5%, 10%, 20%, 30%, 40%, 50%,

60%, 70%, 80%, 90%, 100% of the unlabeled data during the training. These new unlabeled dataset are denote as $D_T^u(0\%), D_T^u(5\%), \ldots, D_T^u(100\%)$ respectively. The experiments are conducted on "Ott." Domain of TWITTER and the results are presented in Fig. 2.

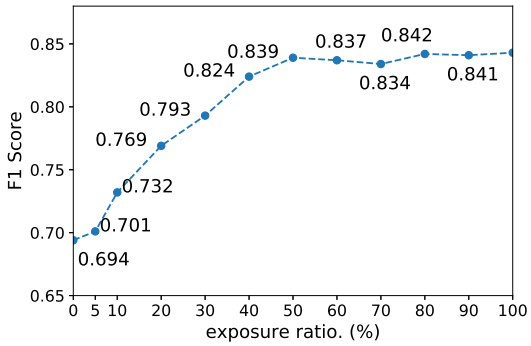

Figure 2: The impact of the size of D_T^u.

From Fig. 2, we observe that the model performs poorly when using a small proportion of the unlabeled data in the training process. For example, exposing $D_T^u(5\%)$ to the DaMSTF only achieves an F1 score of 0.701, which is 14.2% lower than the 0.843 achieved by exposing the $D_T^u(100\%)$. From 0% to 50%, increasing the exposure ratio consistently improves the F1 score. The improvements saturate after more than 50% of the unlabeled data are exposed, which can be explained by the law of large numbers in the statistic theory (Kraaikamp and Meester, 2005). An exposure ratio of 50% can be regarded as a large number for approaching the unlabeled dataset. Thus, $D_T^u(50\%)$ is close to $D_T^u(100\%)$ and $d_{H\Delta H}(\mathbb{D}_T, D_T^u(50\%))$ approximates $d_{H\Delta H}(\mathbb{D}_T, D_T^u(100\%))$, which leads to the performance saturation.

6 Related Work

Domain Adaptation Inspired by the taxonomy in Ramponi and Plank (2020), we categorize the domain adaptation approaches into two categories: *Feature-Alignment approaches* and *Data-Centric approaches*. *Feature-Alignment approaches* (Tzeng et al., 2014; Ganin et al., 2016; Saito et al., 2019) focus on aligning the feature space across domains. In contrast, *Data-Centric approaches* exploit the unlabeled data in the target domain or select the relevant data from the source domain. To select relevant data, Moore and Lewis (2010); Plank and van Noord (2011) design a technique based on topic models for measuring the domain similarity, while Chen et al. (2021) takes a meta-learning algorithm to implicitly measure the domain similarity. To exploit the unlabeled data, pseudo labeling approaches, including self-training (Zou et al., 2019), co-training (Chen et al., 2011), and tri-training (Saito et al., 2017), are widely applied and become an important direction.

Meta-Learning Meta-learning is an emerging new branch in machine learning that focuses on providing better hyperparameters for model training, including but not limited to better initial model parameters, e.g., MAML (Finn et al., 2017), better learning rates, e.g., MetaSGD (Li et al., 2017), and better neural network architecture, e.g., DARTs (Liu et al., 2018). Recent studies revealed the prospect of providing better instance weights (Ren et al., 2018; Shu et al., 2019). Later, (Li et al., 2020; Chen et al., 2021; Wang et al., 2021) realize such meta-learning algorithms in natural language processing tasks. Similar to DaMSTF, Wang et al. (2021) also proposes to combine the meta-learning algorithm and the self-training approach, but their method focuses on the neural sequence labeling task rather than the domain adaptation task. Also, they do not consider the bias between the meta validation set and the test set, whereas reducing such bias is an important contribution of the DaMSTF. In addition, (Chen et al., 2021) takes the meta-learning to achieve domain adaptation via reweighting the instances in the source domain, whose differences to the proposed DaMSTF are discussed in § 5.

7 Conclusion

This paper proposes an improved self-training framework for domain adaptation, named DaMSTF. DaMSTF extends the basic framework for self-training approaches by involving a meta-learning module, which alleviates the label noise problem in self-training. To guarantee the effectiveness of the meta-learning module, we propose a meta constructor to improve the quality of the meta validation set, and propose a domain adversarial module to prevent the training guidance vanishment. Also, the domain adversarial learning module can align the feature space along with the self-training iterations. Extensive experiments on two popular models, BiGCN and BERT, verify the effectiveness of DaMSTF. The ablation studies demonstrate that the meta-learning module, the meta constructor, and the domain adversarial module are indispensable for the effectiveness of the DaMSTF. The limitation, ethical considerations, and social impacts of this paper are in Appendix E and F.

References

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. Machine learning, 79:151–175.

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 549–556.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In Proceedings of the annual meeting of the association of computational linguistics, pages 440–447.

Minmin Chen, Kilian Q Weinberger, and John C Blitzer. 2011. Co-training for domain adaptation. In Proceedings of the International Conference on Neural Information Processing Systems, pages 2456–2464.

Xiang Chen, Yue Cao, and Xiaojun Wan. 2021. Wind: Weighting instances differentially for model-agnostic domain adaptation. In Findings of the Annual Meeting of the Association for Computational Linguistics, pages 2366–2376.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, pages 4171–4186.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, and Jianxin Liao. 2020. Adversarial and domain-aware bert for cross-domain sentiment analysis. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 4019–4028.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International conference on machine learning, pages 1126–1135.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In Proceedings of the International Conference on Machine Learning, pages 1568–1577.

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, 17:2096–2030.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 8342–8360.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 3467–3476.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain ner using cross-domain language modeling. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2464–2474.

Pin Jiang, Aming Wu, Yahong Han, Yunfeng Shao, Meiyu Qi, and Bingshuai Li. 2020. Bidirectional adversarial training for semi-supervised domain adaptation. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 934–940.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations, pages 1592–1601.

FDC Kraaikamp and HLL Meester. 2005. A modern introduction to probability and statistics.

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2022. Unified named entity recognition as word-word relation classification. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 10965–10973.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few shot learning. CoRR, abs/1707.09835.

Zhenzhen Li, Jian-Yun Nie, Benyou Wang, Pan Du, Yuhan Zhang, Lixin Zou, and Dongsheng Li. 2020. Meta-learning for neural relation classification with distant supervision. In Proceedings of the ACM International Conference on Information & Knowledge Management, pages 815–824.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. In Proceedings of the International Conference on Learning Representations, pages 934–940.

Hong Liu, Jianmin Wang, and Mingsheng Long. 2021. Cycle self-training for domain adaptation. pages 22968–22981.

Menglong Lu, Zhen Huang, Binyang Li, Yunxiang Zhao, Zheng Qin, and DongSheng Li. 2022. Sifter: A framework for robust rumor detection. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 30:429–442.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In Proceedings of the Annual Meeting of the Association for Computational Linguistics(Short Papers), pages 220–224.

Tien Thanh Nguyen, Anh Vu Luong, Manh Truong Dang, Alan Wee-Chung Liew, and John McCall. 2020. Ensemble selection based on classifier prediction confidence. Pattern Recognition, 100:107104.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 1566–1576.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP - A survey. In Proceedings of the International Conference on Computational Linguistics, pages 6838–6855.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In Proceedings of the International Conference on Machine Learning, pages 4334–4343.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. 2019. Semi-supervised domain adaptation via minimax entropy. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 8050–8058.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In International Conference on Machine Learning, pages 2988–2997.

Inkyu Shin, Sanghyun Woo, Fei Pan, and In So Kweon. 2020. Two-phase pseudo label densification for self-training based domain adaptation. In European conference on computer vision, pages 532–548.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: learning an explicit mapping for sample weighting. In Proceedings of the International Conference on Neural Information Processing Systems, pages 1919–1930.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. CoRR, abs/1412.3474.

Jianyu Wang and Haichao Zhang. 2019. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In 2019 IEEE/CVF International Conference on Computer Vision, pages 6629–6638.

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2021. Meta self-training for few-shot neural sequence labeling. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pages 1737–1747.

Garrett Wilson and Diane J. Cook. 2020. A survey of unsupervised deep domain adaptation. ACM Transactions on Intelligent Systems and Technology, 11:1–46.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised domain adaptation for neural machine translation. In Proceedings of International Conference on Pattern Recognition, pages 338–343.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In Conference Track Proceedings of International Conference on Learning Representations.

Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. 2019. Confidence regularized self-training. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5982–5991.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016. Learning reporting dynamics during breaking news for rumour detection in social media. CoRR, abs/1610.07363.

A Proof For Theorem 1

Theorem 1. *Let $\mathbf{w}_i$ be the weight of the training instance i, denoted as (x_i, y_i), in $\mathcal{B}$, the gradient of $\mathbf{w}_i$ on $\mathcal{L}_M$ can be represented by the similarity between the gradients on training instance i and the gradients on the meta validation set:*

$$\frac{\partial L_M(\hat{\theta}(\mathbf{w}))}{\partial \mathbf{w}_i} = -\frac{\eta}{|\mathcal{B}|}\cdot[\frac{1}{|D_M|}\sum_{j=1}^{|D_M|}\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j)^T]\cdot\vec{\mathbf{g}}_{\theta}(x_i,y_i)$$

where $\frac{1}{|D_M|}\sum_{j=1}^{|D_M|}\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j)^T$ is the gradients of $\hat{\theta}$ on D_M, $\vec{\mathbf{g}}^i_{\theta}(x_i,y_i)$ is the gradients of θ on the training instance i, η is the learning rate in Eq. (3)

Proof. Based on Eq. (2) and Eq. (3) in § 3.2, we conclude the pseudo updated parameters $\hat{\theta}(\mathbf{w})$ as:

$$\hat{\theta}(\mathbf{w}) = \theta - \eta\cdot\frac{1}{|\mathcal{B}|}\cdot\sum_{x_i,y_i\in\mathcal{B}}\mathbf{w}_i\cdot\frac{\partial\mathcal{E}(\Phi(x_i;\theta),y_i)}{\partial\theta} \quad (12)$$

We then take the gradient of $\mathbf{w}_i$ on $\hat{\theta}(\mathbf{w})$ as:

$$\frac{\partial\hat{\theta}(\mathbf{w})}{\partial\mathbf{w}_i} = -\frac{\eta}{|\mathcal{B}|}\cdot\frac{\partial\mathcal{E}(\Phi(x_i;\theta),y_i)}{\partial\theta} \quad (13)$$

Based on Eq. (13), we derivate the gradient of $\mathbf{w}_i$ on $\mathcal{L}_M$ as:

$$\begin{aligned}\frac{\partial L_M(\hat{\theta}(\mathbf{w}))}{\partial\mathbf{w}_i} &= [\frac{\partial L_M(\hat{\theta}(\mathbf{w}))}{\partial\hat{\theta}(\mathbf{w})}]^T\cdot[\frac{\partial\hat{\theta}(\mathbf{w})}{\partial\mathbf{w}_i}]\\ &= [\frac{1}{|D_M|}\cdot\sum_{j=1}^{|D_M|}\frac{\partial\mathcal{E}(\Phi(x_j;\hat{\theta}(\mathbf{w})),y_j)}{\partial\hat{\theta}(\mathbf{w})}]^T\cdot\\ &\quad [-\frac{\eta}{|\mathcal{B}|}\cdot\frac{\partial\mathcal{E}(\Phi(x_i;\theta),y_i)}{\partial\theta}]\\ &= -\frac{\eta}{|\mathcal{B}|}\cdot[\frac{1}{|D_M|}\sum_{j=1}^{|D_M|}\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j)^T]\cdot\vec{\mathbf{g}}_{\theta}(x_i,y_i)\end{aligned} \quad (14)$$

where the second line is obtained by substituting $\mathcal{L}_M$ and $\hat{\theta}$ with Eq. (4) and Eq. (12). Substitute $\vec{\mathbf{g}}_{\hat{\theta}}(x_j,y_j) = \frac{\partial\mathcal{E}(\Phi(x_j;\hat{\theta}(\mathbf{w})),y_j)}{\partial\hat{\theta}(\mathbf{w})}$ and $\vec{\mathbf{g}}_{\theta}(x_i,y_i) = \frac{\partial\mathcal{E}(\Phi(x_i;\theta),y_i)}{\partial\theta}$ and rearrange the terms, we obtain the third line. The proof of Theorem 1 is completed. □

B Proof For Theorem 2

Definition 1. *disagreement is a measure to quantify the different performances of two different hypotheses on a specific dataset. Denote the two hypotheses as h_1 and h_2, and denote the specific dataset as D, then the disagreement of h_1 and h_2 on D is formulated as:*

$$\epsilon_D(h_1,h_2) = \frac{1}{|D|}\sum_{i=1}^{|D|}[\frac{1}{C}*||h_1(x)-h_2(x)||_1] \quad (15)$$

where C is the number of classes, $h_1(x)$ and $h_2(x)$ are one-hot vectors representing the models' predictions.

Definition 2. *$H\Delta H$-distance is a metric for evaluating the divergence of the data distribution between two datasets. Formally, $H\Delta H$-distance is computed as:*

$$d_{\mathcal{H}\Delta\mathcal{H}}(D_1,D_2) = 2\sup_{h_1,h_2\in\mathcal{H}}|\epsilon_{D_1}(h1,h2)-\epsilon_{D_2}(h1,h2)| \quad (16)$$

where H is the hypothesis space and $\sup$ denotes the supremum.

The concepts *disagreement* and $H\Delta H$-distance are introduced in Definition 1 and Definition 2, respectively. Based on the *disagreement* and $H\Delta H$-distance, the proof for Theorem 2 is presented as below.

Theorem 2. *Assume there exists an ideal hypothesis, denoted as h^*, which correctly map all instances in the target domain to their groud-truth labels. In the self-training iteration t, let $\epsilon_{D_T^l}(h^t)$ and $\epsilon_{D_E}(h^t)$ be the error rate of the hypothesis h^t on D_T^l and D_E, respectively. Then, the error rate of the hypothesis h^t on the target domain is upper bounded by:*

$$\epsilon_{\mathbb{D}_T}(h^t) \le \epsilon_{D_T^l\cup D_E}(h^t)+\frac{1}{2}d_{H\Delta H}(\mathbb{D}_T,D_T^l\cup D_E)+\rho\cdot\epsilon_{D_E}(h^*,h^{t-1}) \quad (17)$$

where $\rho = \frac{|D_E|}{|D_T^l|+|D_E|}$ is a coefficient related to the size of D_T^l and D_E, $\epsilon_{D_T^l\cup D_E}(h^t)$ is the error rate of the hypothesis h^t on the union of D_T^l and D_E.

Proof. In the meta-learning module, the final objective is to minimize the risk loss on the meta validation set $D_T^l\cup D_E$. Thus, according to the learning theory (Ben-David et al., 2010), the upper bound of the error rate on the test set (i.e., the target domain) is:

$$\begin{aligned}\epsilon_{\mathbb{D}_T}(h^t) &\le \epsilon_{D_T^l\cup D_E}(h^t)+\frac{1}{2}d_{H\Delta H}(\mathbb{D}_T,D_T^l\cup D_E)\\ &\quad +\epsilon_{\mathbb{D}_T}(h^*)+\epsilon_{D_T^l\cup D_E}(h^*)\end{aligned} \quad (18)$$

Because h^* is an ideal hypothesis on the target domain, $\epsilon_{\mathbb{D}_T}(h^*)=0$ holds true.

Expanding $\epsilon_{D_T^l\cup D_E}(h^*)$ with the definition in

Eq. (15),

$$\begin{aligned}\epsilon_{D_T^l \cup D_E}(h^*) &= \frac{1}{|D_T^l|+|D_E|} \sum_{(x,y)\in D_T^l \cup D_E} [\frac{1}{C} * ||h^*(x)-y||_1] \\ &= \frac{1}{|D_T^l|+|D_E|}\{ \sum_{(x,y)\in D_T^l} [\frac{1}{C} * ||h^*(x)-y||_1] \\ &\quad + \sum_{(x,y)\in D_E} [\frac{1}{C} * ||h^*(x)-y||_1]\} \\ &= \frac{1}{|D_T^l|+|D_E|}\{|D_T^l| \cdot \epsilon_{D_T^l}(h^*) + |D_E| \cdot \epsilon_{D_E}(h^*)\}\end{aligned} \tag{19}$$

Substituting Eq. (19) into Eq. (18), we have:

$$\begin{aligned}\epsilon_{\mathbb{D}_T}(h^t) \leq\ & \epsilon_{D_T^l \cup D_E}(h^t) + \frac{1}{2} d_{H\Delta H}(\mathbb{D}_T, D_T^l \cup D_E) + \epsilon_{\mathbb{D}_T}(h^*) \\ & + \frac{1}{|D_T^l|+|D_E|}\{|D_T^l| \cdot \epsilon_{D_T^l}(h^*) + |D_E| \cdot \epsilon_{D_E}(h^*)\}\end{aligned} \tag{20}$$

For any instance $(x, y) \in D_E$, y is the pseudo label, i.e., the prediction of hypothesis h^{t-1}. Thus, we have:

$$\begin{aligned}\epsilon_{D_E}(h^*) &= \frac{1}{|D_E|} \sum_{(x,y)\in D_E} [\frac{1}{C} * ||h^*(x)-y||_1] \\ &= \frac{1}{|D_E|} \sum_{(x,y)\in D_E} [\frac{1}{C} * ||h^*(x)-h^{t-1}(x)||_1] \\ &= \epsilon_{D_E}(h^*, h^{t-1})\end{aligned} \tag{21}$$

Since D_T^l is a subset of $\mathbb{D}_T$, $\epsilon_{D_T^l}(h^*) = 0$ holds true. By eliminating $\epsilon_{\mathbb{D}_T}(h^*)$ and $\epsilon_{D_T^l}(h^*)$ in Eq.(20), and substituting $\epsilon_{D_E}(h^*)$ with $\epsilon_{D_E}(h^*, h^{t-1})$, we have:

$$\begin{aligned}\epsilon_{\mathbb{D}_T}(h^t) \leq\ & \epsilon_{D_T^l \cup D_E}(h^t) + \frac{1}{2} d_{H\Delta H}(\mathbb{D}_T, D_T^l \cup D_E) \\ & + \frac{|D_E|}{|D_T^l|+|D_E|} \cdot \epsilon_{D_E}(h^*, h^{t-1})\}\end{aligned}$$

The proof of Theorem 2 is completed. □

C Implementation Details

The implementation of BiGCN to realize the rumor detection task is provided in (Bian et al., 2020), and we follow the description in (Bian et al., 2020) to train the BiGCN model with the TWITTER dataset. The implementation of BERT to realize the sentiment analysis task can be found in (Devlin et al., 2019). We download the pre-trained BERT from `https://huggingface.co/bert-base-uncased`[2] and fit the BERT on the Amazon dataset with the instruction in (Devlin et al., 2019). Since DANN, CRST, MME, WIND, and BiAT are model agnostic, we implement them according to the cited references (Ganin et al., 2016; Zou et al., 2019; Saito et al., 2019; Chen et al., 2021; Wang and Zhang, 2019). For the symbols in Algorithm 1, we set $\mathcal{T}_M$ as 5, $\mathcal{T}_D$ as 5, $\mathcal{T}_G$ as 1. We set η_1 and η_2 in Algorithm 1 as $5e-4$ and $5e-3$ for the BiGCN model, and as $5e-6$ and $2e-5$ for the BERT model. We set both η in Eq. (3) and ϵ in Eq. (6) as $5e-5$ for the BERT model, and $5e-3$ for the BiGCN model. We set γ in Eq. (6) as 0.1 for both the BERT and the BiGCN model. We conduct all experiments the GeForce RTX 3090 GPU with 24GB memory.

D Datasets

TWITTER dataset is provided in the site[3] under a CC-BY license. Amazon dataset is accessed from https://github.com/ruidan/DAS. The statistics of the TWITTER dataset and the Amazon dataset is listed in Tab. 5 and Tab. 6.

Table 5: Statistics of the TWITTER dataset.

Domain	Rumours	Non-Rumours	Total
Charlie Hebdo#	458 (22%)	1,621 (78%)	2,079
Ferguson#	284 (24.8%)	859 (75.2%)	1,143
Germanwings Crash	238 (50.7%)	231 (49.3%)	469
Ottawa Shooting	470 (52.8%)	420 (47.2%)	890
Sydney Siege	522 (42.8%)	699 (57.2%)	1,221
Total	1,921 (34.0%)	3,830 (66.0%)	5,802

Table 6: Statistics of the Amazon dataset

Domains	positive	negative	unlabeled
books	1000 (50%)	1000(50%)	6001
dvd	1000 (50%)	1000 (50%)	34,742
electronics	1000 (50%)	1000 (50%)	13,154
kitchen	1000 (50%)	1000 (50%)	16,786

E Limitation

Although our approach produces promising results on two datasets, there are certain limitations. In the future, we will continue to dig into these concerns. Firstly, we evaluate the DaMSTF on two classification tasks. We do not conduct experiments on other complex NLP tasks, such as machine translation (Yang et al., 2018) or named entity recognition (Jia et al., 2019), although domain adaptation is also important in these tasks. Actually, classification is a fundamental task, and other complex NLP applications can be specified as a case of classification. For example, named entity recognition can be formulated as a word-word relation classification task (Li et al., 2022). Furthermore, DaMSTF is an

[2] under the license apache-2.0

[3] https://figshare.com/ndownloader/articles/6392078/versions/1

improved version of self-training, so it would not be hard to apply DaMSTF in the tasks that can be solved with self-training approaches. In the future, we will investigate our method on other NLP tasks.

Secondly, this paper lacks a theoretical analysis for the convergence of the meta-learning module in the DaMSTF. Although similar algorithms (L2R (Ren et al., 2018) or meta-weight-net (Shu et al., 2019)) provide thorough theoretical analyses for convergence, it is hard to apply their theoretical results to the DaMSTF due to the different designs. Nonetheless, our implementation of DaMSTF converges in all experiments, providing empirical results for the convergence of DaMSTF.

Thirdly, the design in DaMSTF carries out extra computation overhead, which is reflected in two aspects: (i) In every self-training iteration, we use the domain adversarial learning module to re-initialize the model's parameters for model retraining. Thus, the training process in the domain adversarial learning module carries out extra computation overhead. (ii) In the model retraining phase, the updating of the model's parameters are accompanied by the updating of the instances weights, i.e., $\mathcal{T}_M$ step in Algorithm 1. Thus, the training cost in DaMSTF is higher than in previous self-training approaches. In our implementation, we find that a larger batchsize can accelerate the convergence of domain adversarial learning, thereby alleviating the first kind of computation overhead. Also, the approximation techniques in Eq. (6) can effectively reduce the computation cost in the meta-learning module (as illustrated in § 3.2), thereby reducing the second kind of computation overhead. In the future, we will investigate other techniques to accelerate the DaMSTF.

F Ethical considerations and social impacts

This paper involves the use of existing artifact(s), including two benchmark datasets and the pre-trained BERT model. Their intention for providing the artifacts is to inspire the following research, our use is consistent with their intended use.

Rumor, as well as rumor detection, is very sensitive for the social order. In this paper, we conduct experiments on a rumor detection task and prepare to release the code in the future. Since the model's prediction is not that reliable, it may lead to social harm when the model's error prediction is used with malicious intentions. For example, people may use the model's error prediction as support evidence, so as to deny a correct claim or to approve a rumor claim. Here, we seriously declare that the model's prediction cannot be taken as the support evidence. In the released code, we will constrain the input format of the model, making the unprofessional individuals unable to directly use the model.