Adaptive Self-Paced Unsupervised Domain Adaptation for Event Detection using Meta Learning

Abstract

Unsupervised Domain Adaptation (UDA) is an important problem that has been studied extensively in the Vision community. The majority of these approaches rely on combining and balancing various objectives, each corresponds to a distinct statistical aspect, to align the feature space between source and target domains. Recently, meta-learning has emerged as an effective method for the joint optimization of neural network weights and hyperparameters in an end-to-end manner. While this technique has been successfully leveraged for domain generalization problems, the lack of labels for a clean meta validation set prevents its application in UDA tasks. To this end, we propose a meta-learning-based Adaptive Self-Paced Domain Adaptation (ASP-DA) framework inspired by the Self-Paced Learning (SPL) technique. Specifically, ASP-DA employs a meta-SPL module to create two sets of training samples from each mini-batch dynamically during the learning process: an easy set used for weighted objectives in meta-train step, and another set consisted of high-loss samples that produces learning signals for hyperparameters in meta-test step. Extensive experiments on event detection task for ACE-05 dataset demonstrate our framework substantially improves performance on target domains against multiple state-of-the-art baselines, without the need for domain-specific hyperparameter tuning. Furthermore, we present detailed ablation studies and empirical analyses to validate our method and provide insight into how each domain affects model hyperparameters differently.

Introduction

Unsupervised Domain Adaptation (UDA) is one of the hardest and most practical learning setting to address the domain shift problem (Kull and Flach 2014). The goal of UDA is to transfer the knowledge from a labeled source domain to a target domain given its unlabeled data. The majority of UDA approaches come from the works of the Vision community focusing on image classification, in which various training objectives are combined to extract domain-invariant features or to align different aspects of domain-specific extracted features. For example, the most prominent approach for UDA is domain-adversarial neural network (DANN) (Ganin et al. 2016) that employs a domain-adversarial training procedure

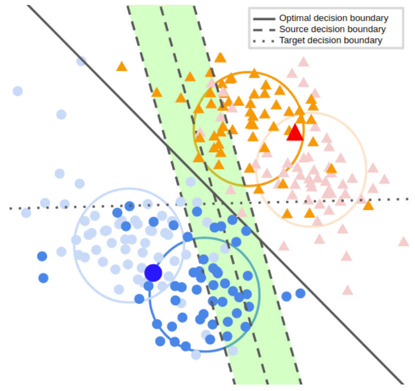

Figure 1: An example where domain shift between source domain (grey colors) and target domain (deep color) results in significant overlaps between high-loss regions of source decision boundary (lime) with high-density target clusters.

between a domain classifier and the network's feature extractor to learn a discriminative and domain-invariant joint feature representation. The simplicity of DANN allows researchers to incorporate it with multiple other objectives such as semi-supervised learning (SSL) regularizers (Shu et al. 2018; Cicek and Soatto 2019), discrepancy metrics (Long et al. 2015; Zellinger et al. 2017), co-training (Kumar et al. 2018), and auxiliary tasks (Bousmalis et al. 2016). Each of them plays an important role in enhancing domain adaptation ability of models in the current state-of-the-art methods. However, it is not trivial to apply these techniques to textual tasks, where large transformer-based language models are essential to achieve top performance, because of the time and resource required to fine-tune and balance the effects of these terms for multiple different adaptation scenarios.

Meta-learning framework is an effective solution for the problem of hyperparameter optimization (Franceschi et al. 2018; Behl, Baydin, and Torr 2019). Furthermore, it has been widely applied by recent works on Domain Generalization (DG) (Li et al. 2018; Dou et al. 2019), in which a learning procedure similar to that of Model-Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017) is leveraged to simulate the domain shift in train-test datasets by a virtual meta train-test set created from data drawn only from source domains. Though DG and UDA share close similarities, the final goal of each learning setting is different. More importantly, the MAML procedure is not applicable for UDA problem because of the lack of a clean validation dataset for meta-test step.

To this end, we propose to dynamically partition the training source data into a low-loss meta-source domain and a high-loss meta-target domain, inspired by Self-paced Learning (SPL) (Kumar, Packer, and Koller 2010). Our framework, called Adaptive Self-Paced Domain Adaptation (ASP-DA), employs a meta-SPL module to control the data selection process for meta train-test set using a learnable age hyperparameter as threshold while also introducing optimized weighting mechanisms for each of DANN's losses. The weighted objectives on meta-source domain are minimized in meta-train step in a direction such that also leading to improvement in model's predictions on meta-target domain. During the learning process, the weight coefficients and the age threshold are also updated based on the model's evaluation performance in meta-test step as in the standard hyperparameter tuning process.

While the meta-target set does not contain samples from the true target domain, we argue that our formulation is beneficial for UDA because of two following reasons. First, the proposed partition can result in two virtual domains with a significant discrepancy, and through learning to address in this hard setting that the model would gain the ability to adapt to other, possibly easier, domains. Another reason is based on the cluster assumption from semi-supervised learning methods (Chapelle, Schölkopf, and Zien 2006), which states that data points of the same class should concentrate around the same cluster, effectively forming a high-density low-loss region. In case of adapting between two highly dissimilar domains, these regions may get shifted significantly, as a consequence low-loss regions of target domain may contain considerable intersection with high-loss regions of source domain, as illustrated in Fig. 1. In other words, by learning to adapt the high-loss meta-target domain, the model would also be able to generalize to a significant portion of the true target domain.

We extensively evaluate the proposed framework over event detection task on ACE-05 dataset. The experimental results when adapting to multiple different domains clearly demonstrate the effectiveness of the model. Ablation studies and detailed analyses are provided to validate each main component of our model and provide insights for future researches.

Related Work

Unsupervised Domain Adaptation The main line of research on UDA focuses on learning domain-invariant, which is either achieved by explicitly reducing the distance between source and target feature space measured by some distribution discrepancy metric (Long et al. 2015; Zellinger et al. 2017), or by adversarial training in which the feature extractor is trained to fool a domain classifier, both are jointly optimized to arrive at an aligned feature space (Ganin et al. 2016). We focus on applying the latter in transformer-based model (BERT) (Devlin et al. 2019) for textual tasks. Previous works have provided empirical results on different domains (Wright and Augenstein 2020; Lin et al. 2020), different tasks (Naik and Rosé 2020; Du et al. 2020), most of which presented little to no improvement following the standard domain adversarial training framework. We further verify this point in our baseline performances. One hypothesis is that the over-parameterization nature of BERT has overwhelmed DANN's effort to align representations between domains, causing the model to over-fit on source dataset in fine-tuning process.

Sample Weighting The idea of sample weighting was initiated as a preprocessing step to pre-evaluate the sample weights for training loss, using prior knowledge or statistics of given data (Zadrozny 2004). To achieve better weighting for different data situation, the method evolved into designing a weighting function that takes in as input the training loss to adaptively output weight of the corresponding sample during the training process. There are two main research directions of this approach: addressing class imbalance by monotonically increasing function that imposes larger weights to ones with larger loss values (Sun et al. 2007; Lin et al. 2017), and suppressing the effect of noisy labels using monotonically decreasing function which focus on low-loss easy samples (Kumar, Packer, and Koller 2010; Jiang et al. 2014). Although straightforward to apply, the above methods are limited in that they all need a pre-specified closed-form weighting function, while their respective hyperparameters are sensitive to the change of training data such that careful tuning is required.

Meta Learning Originally designed to mimic a human's ability to quickly learn and adapt to new concepts based on prior knowledge (Schmidhuber 1987; Thrun and Pratt 1998), meta-learning is progressively becoming more popular in deep learning researches. There are three main categories of meta-learning algorithms: learning a metric space to measure distance or similarity among data (Vinyals et al. 2016; Sung et al. 2018), learning an optimizer which updates all of model parameters in a latent parameter space (Andrychowicz et al. 2016; Chen et al. 2018), and learning an initialization that is good for all tasks and able to fast adapt to unseen tasks (Finn, Abbeel, and Levine 2017; Jamal and Qi 2019). Our approach falls into the last category, where the learning process follows MAML, more specifically its variant for DG problem in (Li et al. 2018). Furthermore, we employ a meta weighting function similar to that of Meta-Weight-Net (MWN) (Shu et al. 2019) to learn an adaptive weighted objective for each domain adaptation scenario.

Background and Notation

We denote the source dataset $\mathcal{D}^{\mathbf{s}}_{\mathbf{XY}} = \{(x^{\mathbf{s}}_i, y^{\mathbf{s}}_i)\}_{i=1}^{N^{\mathbf{s}}}$ consisted of $N^{\mathbf{s}}$ samples from source domain $\mathbf{s}$ and an unlabeled set of $N^{\mathbf{t}}$ samples $\mathcal{D}^{\mathbf{t}}_{\mathbf{X}} = \{x^{\mathbf{t}}_i\}_{i=1}^{N^{\mathbf{t}}}$ drawn from target domain $\mathbf{t}$. Label space $\mathbf{Y} = \{1, 2, \cdots, K\}$ of K classes is shared across domain.

BERT and Adapter-based Fine-tuning

Pre-trained transformer-based language model has become a stable for NLP task. Its encoder is designed to capture contextual embedding of a given input text through the usage of a layer-block containing a self-attention sub-layer and a feed-forward sub-layer. In particular, a BERT model for classification task is designed as follow:

$$H_c \circ B^{L-1} \circ \cdots \circ B^0(x) \ , \ where \ B^l = b^l_{ff} \circ b^l_{at} \qquad (1)$$

in which L is the number of layer-block, b^l_{ff} and b^l_{at} is the pre-trained feed-forward and attention sub-layer respectively, an additional classification head H_c is added for label prediction.

To avoid having to fine-tuning hundreds of millions of parameters, (Houlsby et al. 2019) introduce Adapter-based Fine-tuning framework, in which new modules called adapters are injected after each layer of BERT: $b^l_i \rightarrow a^l_i \circ b^l_i$, where $i \in [ff, at]$. These adapters take on the task of adapting original BERT's layers, which are now fixed, to the target task. To limit the number of additional parameters, each of them follows a bottleneck architecture: $a^l_i = a^l_{i,up} \cdot a^l_{i,dw}$, where $a^l_{i,dw} : \mathbb{R}^{d_{model}} \rightarrow \mathbb{R}^{d_a}$ and $a^l_{i,up} : \mathbb{R}^{d_a} \rightarrow \mathbb{R}^{d_{model}}$ are feed-forward layers ($d_a \ll d_{model}$).

Domain Adaptation Neural Network

(Ben-David et al. 2010) introduce the notion of $\mathcal{H}\Delta\mathcal{H}$ divergence to bound the loss of a model on target domain with its performance on source domain. (Ganin et al. 2016) adopted this idea for deep learning architecture by learning a domain classifier $H_d(\theta_d)$ concurrently with the main downstream task, using unlabeled samples and their domain labels from both domains. The goal is that the representations obtained from feature extractor $f(\theta)$, while discriminative for the task at hand, are also indistinguishable to H_d. This is achieved by pushing the encoder to both minimize the task loss through H_c and maximally misdirect H_d, whilst also minimizing the domain classification through θ_d, resulting in the following training objective:

$$\min_{\theta,\theta_c,\theta_d} \mathcal{L}_c(f(x^{\mathbf{s}}_i;\theta), y^{\mathbf{s}}_i;\theta_c) + \alpha\mathcal{L}_d(g_{\lambda_d}(f(x_i;\theta)), y^d_i;\theta_d) \qquad (2)$$

The min-max optimization is formed in Eq. 2 using a gradient reversal layer (GRL) g_{λ_d}, which acts as identity function in forward passs while scales the gradient flowing through by λ_d ($\lambda_d < 0$) during backward pass.

Self-Paced Learning

Curriculum Learning (Bengio et al. 2009) was introduced to replicate the 'easy to hard' learning method of human. However, the approach requires an ad-hoc implementation of easiness based on some predetermined heuristics, which may not be flexible and expressive enough for the continuously updating model. Thus, (Kumar, Packer, and Koller 2010) devise a method called Self-Paced Learning, that jointly learn the model and its curriculum. Specifically, SPL employs an age hyperparameter λ_a that represents the current learning pace of the model. The objective is reformulated as a weighted loss where each instance's contribution is thresholded by λ_a as follow:

$$\mathcal{L} = \sum_{i=1}^{n} v_i(l_i;\lambda_a)l_i \ ; \ v_i = \begin{cases} 1, & \text{if } l_i < \lambda_a \\ 0, & \text{otherwise.} \end{cases} \qquad (3)$$

where l_i is the corresponding loss of i-th training sample. Intuitively, λ_a is the "age" of the model which is set to gradually grow as training proceed. Thus, only easy samples are considered at the initial learning stage while samples with larger losses will be slowly added to the model's curriculum as it progresses.

Model Agnostic Meta Learning

(Finn, Abbeel, and Levine 2017) proposed a model-agnostic meta-learning framework that seeks to find a globally optimal initialization of parameters for all tasks in case of few-shot learning through an episodic training procedure. Each episode involves a meta-train task that MAML performs a certain number of optimization steps on, the result of which is then evaluated on a meta-test task and provides the final gradient update for the model. Our method focuses on a variant of MAML proposed by (Li et al. 2018) for DG problem called Meta-Learning Domain Generalization (MLDG). Given a set of source domains $\mathbf{S}$ as training data, MLDG partitions it into meta-train domain $\mathbf{S}_{tr}$ and meta-test domain $\mathbf{S}_{ts}$, which are then used in meta-train and meta-test steps as follow:

$$\mathcal{L}_{tr}\left(\mathbf{S}_{tr};\theta\right) = \sum_{x\in\mathbf{S}_{tr}} l(x;\theta); \ \bar{\theta} = \theta - \alpha\nabla_\theta\mathcal{L}_{tr} \qquad (4)$$

$$\mathcal{L}_{ts}\left(\mathbf{S}_{ts};\bar{\theta}\right) = \sum_{x\in\mathbf{S}_{ts}} l(x;\bar{\theta}); \ \theta = \theta - \gamma\frac{\partial(\mathcal{L}_{tr} + \beta\mathcal{L}_{ts})}{\partial\theta} \qquad (5)$$

where α , γ are learning rates and β is balancing term. In the context of DG, $\mathbf{S}_{tr}$ and $\mathbf{S}_{ts}$ are drawn from the a random shuffle of domains so that each parameter's update is alway good for any virtual unseen domain.

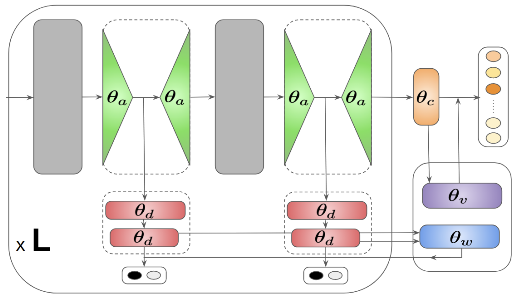

Figure 2: Architecture overview.

Model

Our model's feature encoder is a fixed pre-trained BERT encoder with hidden dimension $\mathbb{R}^{d_h}$, augmented by adapters with bottleneck representation of size $\mathbb{R}^{d_a}$. We refer to the main model learnable parameters as $\theta = (\theta_a, \theta_c, \theta_d)$, which includes the parameters of adapters, the main classification head, and the DANN head. In addition, our meta-SPL module consists of two weighting mechanisms: a layer-wise $f_w(\theta_w) : \mathbb{R}^{d_a} \rightarrow 1$ that takes adapter representation of each layer and outputs the relative "magnitude" of which the corresponding layer should be aligned, and an instance-wise $f_v(\theta_v) : \mathbb{R} \rightarrow \mathbb{R}$ which weights the contribution of each example based on the its loss and a learnable age parameter λ_a. We refer to the set of source samples whose losses are less than λ_a as meta-source domain $\mathbf{S}_{tr}$ while the rest is meta-target domain $\mathbf{S}_{ts}$. The latter acts in meta-test step as a validation set used to evaluate the model after meta-train step and provide learning signals to fine-tune the "hyperparameters" from the meta-SPL module. The overall architecture is presented in Fig. 2.

Meta hyperparameter Optimization

The survey presented by (Rogers, Kovaleva, and Rumshisky 2020) provides a detailed probing and understanding of how the different layer-block of BERT encodes different types of information. In summary, lower-level layers near the original inputs, thus containing mostly general statistical knowledge. This information gradually transformed as middle layers predominantly represent syntactic knowledge, which is the most transferable across tasks. On the other hand, the topmost layers are the closest to the learning of the downstream task, hence encoding extremely specific semantic knowledge regarding the corresponding task. Accordingly, each layer should contain a different amount of discrepancy between source and target domains.

To align these representation spaces between the two domains, we employ multiple domain classifiers at the bottleneck of every adapter:

$$\mathcal{L}_d = \sum_{i \in [ff, at]} \sum_{l=1}^{L} w_i^l \mathcal{L}_{d,i}^l(\mathbf{z}_{d,i}^l, \mathbf{y}_d; \theta_{d,i}^l) \tag{6}$$

where each $\mathcal{L}_d^l$ is an adversarial term of a different DANN as in Eq. 2 , taking adapter representations $\mathbf{z}_{d,i}^l$ of layer lth and domain labels $\mathbf{y}_d$ as inputs. These losses are weighted by a set of coefficients $\{w_i^l\}$, all of which, if following standard learning procedure, would be hyperparameters that required careful tuning for each specific domain. This is impractical, as in case of BERT and our setting, there would be a total of 24 hyperparameters (2 per layer-block). To address the above issue, we employ a small feed-forward network f_w with a softmax final layer to output the relative layer-wise weights:

$$\mathbf{W}_i = [w_i^0, \cdots, w_i^{L-1}] = f_w(\mathbf{Z}_{d,i}; \theta_w), \; i \in [ff, at] \tag{7}$$

where $\mathbf{Z}_{d,i} \in \mathbb{R}^{L \times d_a}$ is a set of layer representations, each element of which is the sum of all adapter representations of the corresponding layer with respect to the current mini-batch.

Meta Self-Paced Learning

We leverage meta-learning framework to optimized θ_w in the previous section for target domain. More formally, we simulate the train-test process by splitting the source domain instances of the current mini-batch into two disjoint sets based on the age value λ_a. On one hand, the easy samples are used for meta-train step, in which the objective consists of the domain adversarial loss from Eq. 6 and a SPL-weighted classification loss:

$$v_i = \sigma \circ f_v \circ \max(0, \frac{-l_i}{\lambda_a} + 1) \; ; \; \mathcal{L}_{ce}\left(\mathbf{S}_{tr}\right) = \sum_{x_i \in \mathbf{S}_{tr}} v_i l_i(\theta) \tag{8}$$

$$\mathcal{L}_{tr}\left(\theta\right) = \mathcal{L}_{ce}\left(\mathbf{S}_{tr}; \theta_a, \theta_c\right) + \mathcal{L}_d\left(\mathbf{S}_{tr}, \mathbf{T}; \theta_a, \theta_d\right) \tag{9}$$

where σ is a sigmoid function to guarantee resulting weights located in the interval of [0, 1], f_v is a small feed-forward network with no bias so that the 0-valued inputs will also correspond to outputs of the same value. Usually, k gradient steps are applied on Eq. 9 to approximate the optimal solution that minimize the meta-train objective. Because of the sizeable transformer encoder, high value of k will cost serious computation overhead. Thus, We use $k = 1$, from which we observe no significant performance loss:

$$\bar{\theta} = \theta - \alpha \nabla_\theta (\mathcal{L}_{ce}\left(\theta_a, \theta_c\right) + \mathcal{L}_d\left(\theta_a, \theta_d\right)) \tag{10}$$

Next, the meta-test objective is the standard cross-entropy loss on samples in meta-target domain $\mathbf{S}_{ts}$ with loss values higher than λ_a:

$$\mathcal{L}_{ts}\left(\bar{\theta}; \theta_w, \theta_v, \lambda_a\right) = \sum_{x_i \in \mathbf{S}_{ts}} l_i(\bar{\theta}) \tag{11}$$

Following MLDG, meta-train and meta-test are combined in the final objective as follow:

$$\underset{\theta}{\operatorname{argmin}} \; \beta \mathcal{L}_{ts}\left(\bar{\theta}\right) + \mathcal{L}_{tr}\left(\theta\right) \; ; \; \underset{\theta_w, \theta_v, \lambda_a}{\operatorname{argmin}} \; \mathcal{L}_{ts}\left(\bar{\theta}\right) \tag{12}$$

The second term in Eq. 12 is the result of passing the weights computed by meta-SPL module in Eq. 7 and 8 into Eq. 9 as pre-determined values, not learnable variables.

Incorporating Pseudo Label

Pseudo Labels is an effective method to improve target domain performance by leveraging the predictions of previous step on unlabeled target data as additional learning signals for the main downstream task. We use the pseudo-labeled target data only for $\mathcal{L}_{ce}$ from Eq. 9 in meta-train step, in which they are weighted and thresholded by meta-SPL module using the same λ_a as source data: $\mathcal{L}_{ce}\left(\mathbf{S}_{tr}, \overline{\mathbf{T}}\right) = \sum_{x_i \in \mathbf{S}_{tr} \cup \overline{\mathbf{T}}} v_i l_i(\theta)$ where $\overline{\mathbf{T}}$ is the set of target samples with losses lower than λ_a. To alleviate the confirmation bias in pseudo-labeling, (Xie et al. 2019) provided strong regularizations and data augmentations to prevent model from propagating its own inaccuracy throughout the training process. In our case, meta-SPL module would ensure that only high confident pseudo labels are used, thus suppressing the noises and providing a robust training for the model. In addition, as we will discuss later section, the gradient updates of these samples are also regularized by the meta-learning framework, forcing them to be consistent with meta-target domain.

Experiments

Dataset, Settings, and Baselines

We evaluate our framework on event detection task for ACE-2005 dataset (Walker et al. 2005) whose documents were collected from 6 different domains: Newswire (`nw`) - 20%, Broadcast news (`bn`) - 20%, Broadcast conversation (`bc`) - 15%, Weblog (`wl`) - 15%, Usenet Newsgroups (`un`) - 15%, Conversational Telephone Speech (`cts`) - 15%.

Settings Given a trigger word in the context of an event mention, the model is required to perform a multi-class classification task that assigns a predicted label into one of the

System	In-domain(bn+nw)			Out-of-domain (bc)			Out-of-domain (cts)			Out-of-domain (wl)			Out-of-domain (un)		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
BERT	75.8	72.5	74.1	73.5	68.9	71.1	73.7	69.5	71.5	62.2	51.6	56.4	77.6	59.9	64.8
BERT+DANN	73.4	76.0	74.7	73.9	69.4	71.5	76.4	53.0	62.5	59.9	53.2	56.3	66.9	59.3	62.8
BERT+Adapter	76.1	78.0	77.1	77.9	71.5	74.5	73.5	68.2	71.8	64.6	53.0	58.2	70.0	65.9	67.8
Uniform	76.8	79.4	78.1	75.4	66.3	70.5	80.4	21.0	33.3	61.8	45.7	52.6	72.7	57.0	63.8
Focal Loss	78.2	77.6	77.9	71.7	72.9	72.2	72.9	68.5	70.1	64.8	54.2	59.0	71.4	66.8	69.0
Class-Balanced	79.3	78.3	78.7	77.8	68.0	72.5	78.0	44.0	56.2	59.0	50.3	54.3	68.1	57.8	62.5
SPL	77.1	80.0	78.5	77.9	70.7	74.2	79.2	53.0	63.5	62.1	53.2	57.1	71.7	62.8	66.8
DomCls	79.6	76.4	77.9	73.0	74.5	73.7	78.2	48.7	59.9	62.9	53.1	57.5	68.7	65.0	66.8
ASP-DA - SPL	74.5	79.7	77.0	77.5	72.0	74.6	76.4	67.6	71.6	64.1	51.9	57.4	69.1	64.7	66.8
ASP-DA - DANN	74.3	80.3	77.2	75.7	72.9	74.2	73.9	69.1	71.4	61.6	51.9	56.3	71.6	65.3	68.3
ASP-DA - PL	77.8	75.1	76.4	75.1	73.5	74.3	72.4	70.5	71.4	62.6	52.4	57.0	71.7	66.3	68.6
Random	73.0	76.4	74.7	75.6	73.3	74.4	69.2	53.6	60.2	61.0	50.3	55.0	72.9	61.9	67.0
Reverse	77.7	75.0	76.3	78.2	70.6	74.2	78.9	53.5	63.8	65.0	50.7	57.0	71.5	63.7	67.4
ASP-DA	75.4	80.0	**77.7**	77.5	73.1	**75.2**	76.1	69.5	**72.6**	65.6	54.6	**59.6**	70.2	68.3	**69.2**

Table 1: Unsupervised domain adaptation for event detection. Performance on the **ACE-05** test datasets for different domains.

System	Out-of-domain (bc)			Out-of-domain (wl)		
	P	R	F	P	R	F
Fixed (25)	79.3	68.9	73.7	65.8	50.0	56.8
Fixed (50)	75.0	73.7	74.3	66.3	49.5	56.6
Fixed (75)	76.4	72.0	74.1	65.9	52.7	58.6
Linear Incrs	74.9	71.7	73.3	61.6	54.7	57.9
Meta (Ours)	77.5	73.1	**75.2**	65.6	54.6	**59.6**

Table 2: Performances for Age hyperparameter Analysis

System	Out-of-domain (bc)			Out-of-domain (wl)		
	P	R	F	P	R	F
Constant	75.8	71.5	73.6	63.2	52.6	57.4
Anneal Up	75.4	71.0	73.1	63.5	52.6	57.4
Anneal Down	74.0	74.8	74.4	62.3	51.1	56.1
Meta (Ours)	77.5	73.1	**75.2**	65.6	54.6	**59.6**

Table 3: Peformances for DANN weighting analysis

pre-defined 34 event types (including 1 negative type). For UDA setting, we gather data from two closely related domains, bn and nw, to create a sizable source domain dataset, 80% of which are used training whilst the rest are split equally into training and test target domain for in-domain setting. For out-of-domain settings, each of the other domains is considered the target domain of a single adaptation scenario, where 20% of its documents are unlabeled training target data and the remainders are utilized as the test dataset. In the following sections, all of the considered models' hyperparameters are only tuned based on bc domain.

Baselines We provide a comprehensive comparison of our proposed method with multiple baselines from 3 categories:

- **No Weighting:** models that does not leverage any weighting mechanism. **BERT** and **BERT+Adapter** are only fine-tuned on only labeled source domain, whereas **BERT+DANN** follows the standard adversarial training in Eq. 2
- **Functional:** weight of each sample is given by a pre-determined function. **Uniform** treats each sample's loss equally, **Focal Loss** down-weights well-classified instance exponentially (Lin et al. 2017), and **Class-Balanced** uses a weighting factor that is inversely proportional to the number of samples (Cui et al. 2019).
- **Curriculum:** a curriculum is used to compute the contribution of each training instance. In **DomCls**, the weights are provided in prior by a domain classifier of a trained DANN to output the probabilities of a sample belonging to target domain; whereas **SPL**'s dynamic curriculum computes the weighting coefficients based on the corresponding losses as in Eq. 3.

Noted that models in **Functional** and **Curriculum** categories employ both adapter-based fine-tuning and adversarial training procedure.

Main Results - Unsupervised Domain Adaption

The first two row-blocks of Table 1 present the performances of the above baselines in each domain adaptation scenario. **BERT+DANN** only provides slight improvement for finetune domain bc, while significant degrades model's performances on the other three. On the other hand, adapter-based fine-tuning procedure manages to prevent the model from over-fitting on source domain with the fixed BERT's layers, thus performs better than **BERT** across domains. However, simply applying DANN for the adapter-based model without any weighting mechanism, as in **Uniform**, also has adverse effects on out-of-domain performances. Regarding instance-weighting baselines, their in-domain results are slightly are higher than other models due to their ability to balance the training process, especially for Class-Balanced which addresses the extreme negative-skewed label distribution of the given source data. In contrast, its domain adaptation ability is actually the lowest because of the change in data distribution across domains. **Focal Loss** and **SPL** perform generally better in out-of-domain settings as they generate weighting coefficients adaptively based on the current losses, without involving any domain-specific statistics. On the other hand, **DomCls** requires computing a specific curriculum for each domain, yet perform worse than the dynamic curriculum imposed by SPL. Notably, domain adaptation performances of these baselines vary greatly for each domain, whereas our framework is robust across every adaptation settings. Despite using the same hyperparameters specifically tuned for one domain (bc), **ASP-DA** provides consistent improvements for all four domains, achieving on average 0.5 points higher in F1 score compared to the corresponding best performing baseline.

Ablation Study

In the third row-block of Table 1, we conduct an ablation study to validate the effectiveness of each of our main components by investigating the performance of the following variations of our model:

Self-Paced Learning: **ASP-DA - SPL** follows the normal SPL process in Eq. 3 to produce the weighting coefficients and train-test datasets for meta-learning. Our model outperformed this variant across all domains, even in the in-domain setting, which confirms the superiority and flexibility provided by the jointly optimized pacing and weights from our meta-SPL module. In the following section, we will provide detailed analysis of the impact of the meta-learned age hyperparameter.

Domain Adversarial Training: The effect of domain adversarial training in our framework is measured by removing DANN components as in **ASP-DA - DANN**. This modification markedly degrades out-of-domain performance, most notably the 3.5 F1 points decrease when adapting to `wl` domain which is highly dissimilar to the source domain. The results demonstrate that our model successfully applies domain adversarial for large pre-trained transformer-based models, compared to the ineffective standard implementation of DANN for BERT in the previous section and other prior works.

Pseudo Labels: We train our model without leveraging pseudo-labels for the unlabeled target data in **ASP-DA - PL**. There is a 1 points increase in F1 score on average of both in and out-of-domain settings, suggesting that our framework is able to alleviate the confirmation bias problem and extract useful learning signals from the additional target domain to improve model's capacity for the main downstream task.

Meta-test Selection: To examine the correctness of our assumption, we augment the data selection process for meta domains in **Random** and **Reverse**. The former randomly selects training samples for each meta domain, whereas the latter implements the opposite hypothesis by choosing hard and easy instances for meta-train and meta-test sets, respectively. Both variants result in a considerable decline in domain adaptation results. Notably, the significant performance drop in the in-domain setting of **Random** indicates that simply constructing train-test sets without any appropriate condition can do more harm than good for the meta-learning process. These empirical observations further confirm our initial assumption on how domain shift correlates well with the easy meta-train and hard meta-test sets.

The Values of Age hyperparameter

Age hyperparameter λ_a is usually the hardest to tune in a SPL system due to the fact that aside from the initial value, determining how λ_a changes throughout the training process also has a major impact on the final performance. Several prior works (Li and Gong 2017; Ren et al. 2017) have proposed alternative age schedulers in place of the naive strategy which adds/multiples λ_a with a constant at each epoch. However, the value of λ_a in these methods still follows a predefined sequence, implying the need for a meticulous tuning process. In contrast, our meta-SPL module updates λ_a based on optimization signals from meta-test set, thus always able to create an appropriate dynamic curriculum regardless of different learning tasks and datasets. In Table 2, we examine how different values and schedules of age hyperparameter affect performances on `bc` and `wl` domains, which are the easiest and hardest adaptation scenarios respectively. The **Fixed (p)** settings with $p \in [25, 50, 75]$ are variations of our model with λ_a values always corresponding to the unchanged p-th percentile of the current mini-batch's sample losses; or in other words, the number of samples in meta-train set is always a constant p percent that of the current mini-batch. Additionally, we evaluate the case in which p is linearly increased as training proceeds, similar to the standard SPL process, in **Linear Incrs** setting. The results show that the lower p is, the worse model performs, indicating that with too few meta-train data, the model will not be able to adapt to the hard meta-test domain. Surprisingly, the gradual rising scheduler of **Linear Incrs** is not as effective as the other **Fixed** variants. This means that the easy-to-hard assumption of prior SPL systems is not suitable for our meta-learning framework.

To gain more insight into how age hyperparameter changes throughout the training process of each domain, we plot the values of λ_a in source-losses percentile against the number of update steps for 10 epochs in the right subplot of Fig. 3. While λ_a quickly follows the standard incremental trend in the first 2000 steps, it starts to plateau within the 60-70 percentile range until eventually starting to decrease. Notably, behavior of λ_a diverges across domains in subsequent steps. Whereas λ_a continues the to decline in `bc` and `cts` domains, it experiences a complete trend reversal at the end of the training of `wl` domain. We hypothesis that this drastic change of λ_a is because of the gradients' dot product term that the objective in Eq. 12 implies, which we will delve deeper into in the discuss section below. The $\bigcap$ shape of λ_a correlates with the term's value as the model maximizes it to align the gradient directions between the meta train-test domains, going from negative initially as the training started, to 0 which causing the plateau, then gradually becoming positive as the model was able to adjust the updates of meta-train set to be consistent with that of meta-test set. However, for hard adaptation such as `wl` domain, too few data in meta-train set can cause a major disparity between the two meta domains again, thus the resulting trend reversal at the last few steps.

We also visualize the same plot for target-pseudo-losses percentile, which leads to an interesting observation: Initially, the model followed its own pseudo labels without any constraint and the high value of λ_a percentile represents model's incorrect overconfidence. However, these pseudo-label updates will cause discrepancies with meta-test domain, thus the meta-learning framework will gradually fix the corresponding predictions, allowing only quality pseudo samples to be included in meta-train set. Eventually, the target trend converges with the source ones, suggesting that model's predictions on pseudo labels are then as consistent as on clean training labels.

Balancing Domain Adversarial Losses

Previous works have observed that the weight of DANN in the combined objective has a significant impact in the overall adaptation performance of the model. We further validate this point by investigating how different domain adversarial weighting schemes affect the results on `bc` and `wl` domains.

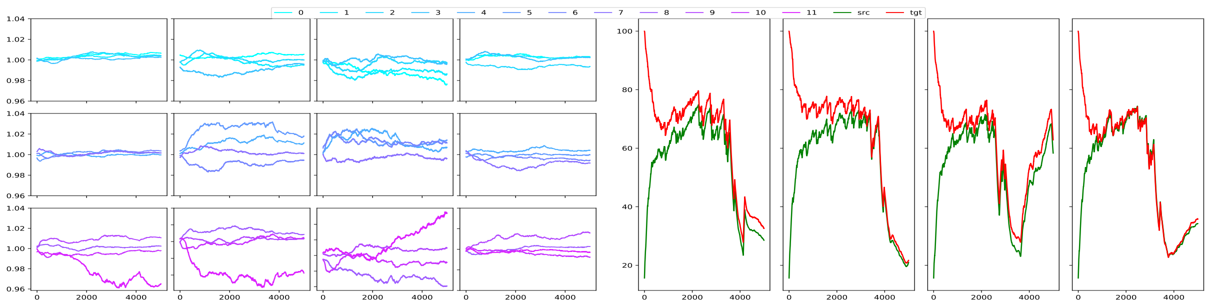
Figure 3: Four columns in each subplot correspond to domain `bc`, `cts`, `wl`, and `un`, respectively .**Left:** Layer-wise DANN weights at each training step. **Right:** source and target age percentiles at each training step.

Specifically, we evaluate 3 types of layer-wise weighting: (i) **Constant** - all layers share the same w^l value, (ii) **Anneal Up** - w^l slowly increases from lower to higher layers, and (iii) **Anneal Down** - w^l is highest for the first layer and gradually declines for subsequent layers (we only consider the coefficients of feed-forward adapters here as we observe weights in the same layer-block behave approximately the same). The results are present in Table 3, in which none of the schemes is better than the others in both domains. In contrast, the meta-learned coefficients of our framework manage to boost model's performances in every adaptation setting, especially for the hard `wl` domain where domain adversarial training matters the most.

We further visualize how each layer's weight changes during the learning process across domains in the left subplot of Fig. 3. In particular, we partition 12 layers of BERT-base model into 3 groups of 4 sequential layers, each of which is known to contain a different type of information that are important for a different type of task as describe in the previous section. We can observe from the graphs a certain pattern: the higher level the group is, the more volatile its layers' coefficients are. However, there is no specific rule shared among all domains regarding the value of each layer's weight. Considering the last layer whose representation standard DANN systems usually use for domain adversarial training, w^{11} in our case, when adapting to `bc` and `cts` domains, moves in a complete opposite direction compared to that of `wl` domain. This affirms the sensitivity of domain adversarial balancing term to each individual domain and further justifies the effectiveness of the jointly optimized weighting in our framework.

Discussion

Following the analysis of MLDG framework presented in (Li et al. 2018), we decompose the meta-test loss, given that $\bar{\theta} = \theta - \alpha\mathcal{L}'_{tr}(\theta)$, using the first order Taylor expansion:

$$\mathcal{L}_{ts}\left(\theta - \alpha\mathcal{L}'_{tr}(\theta)\right) = \mathcal{L}_{ts}\left(\theta\right) + \frac{\partial\mathcal{L}_{ts}\left(\theta\right)}{\partial\theta}\left(-\alpha\frac{\partial\mathcal{L}_{tr}\left(\theta\right)}{\partial\theta}\right) \quad (13)$$

Denoting $\mathbf{G} = \frac{\partial\mathcal{L}_{ts}(\theta)}{\partial\theta} \cdot \frac{\partial\mathcal{L}_{tr}(\theta)}{\partial\theta}$ and plugging Eq. 13 into the final objective to update main model parameters from Eq. 12 results in the following optimization problem:

$$\underset{\theta}{\operatorname{argmin}}\ \mathcal{L}_{tr}\left(\theta\right) + \mathcal{L}_{ts}\left(\theta\right) - \beta\alpha\mathbf{G} \quad (14)$$

The third term in Eq. 14 is a gradient-based regularization that penalizes inconsistency between parameter updates of meta-train and meta-test domains. By enforcing loss gradients of the two domains to follow a similar direction, Eq. 14 prevents the model from over-fitting to a single domain, effectively improves model's adaptation capacity provided that meta-test set is 'close' to target domain.

We further examine how the meta-learning framework affects the values of meta-SPL module's parameters $(\theta_w, \theta_v, \lambda_a)$ in our model. Plugging Eq. 13 into the gradient of λ_a, we have:

$$\frac{\partial\mathcal{L}_{ts}\left(\bar{\theta}\right)}{\partial\lambda_a} = -\alpha\frac{\partial\mathcal{L}_{ts}\left(\theta\right)}{\partial\theta} \cdot \frac{\partial^2\mathcal{L}_{tr}\left(\theta\right)}{\partial\theta\partial\lambda_a} = -\alpha\mathbf{G} \cdot \frac{\partial f_v(\lambda_a)}{\partial\lambda_a} \quad (15)$$

From Eq. 15, we see that the multiplicative factor $\mathbf{G}$ also controls how the value of λ_a changes throughout the meta-learning process. When there is a significant discrepancy between meta-train and meta-test domain, $\mathbf{G}$ would have a negative value, which would in effect pushing λ_a higher and allowing more samples into meta-train set for easier adaptation to meta-test set. Conversely, a positive $\mathbf{G}$ would imply that the model is good enough to align the current meta domains, thus gradually pulling λ_a down to make the task harder. This behavior is clearly illustrated in Fig. 3. Similar arguments can be made for the meta-learned weighting coefficients, where $\mathbf{G}$ would encourage samples whose gradients are similar across domains while decreasing the contribution of those whose gradients are not. These understanding are also presented in (Shu et al. 2019) and closely related to how MAML works (Nichol, Achiam, and Schulman 2018; Raghu et al. 2019)

Conclusion

We present a novel meta-learning framework for UDA setting that leverages the cluster assumption to simulate the domain shift problem. A meta-SPL module is employed to adaptively partition source domain into meta-train and meta-test set, while simultaneously learns the instance-wise and layer-wise weights for the loss terms of downstream task and domain adversarial task respectively. The proposed model significantly improves domain adaptation performances against various baselines on every domain without domain-specific hyper-parameter tuning. In the future, we are going to extend our approach to other domains and tasks while also incorporating different novel domain adaptation regularization methods.

References

Andrychowicz, M.; Denil, M.; Colmenarejo, S. G.; Hoffman, M. W.; Pfau, D.; Schaul, T.; and de Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., *NIPS*, 3981–3989.

Behl, H. S.; Baydin, A. G.; and Torr, P. H. S. 2019. Alpha MAML: Adaptive Model-Agnostic Meta-Learning. *ArXiv*, abs/1905.07435.

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2): 151–175.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 41–48. New York, NY, USA: Association for Computing Machinery. ISBN 9781605585161.

Bousmalis, K.; Trigeorgis, G.; Silberman, N.; Krishnan, D.; and Erhan, D. 2016. Domain Separation Networks. In *NIPS*.

Chapelle, O.; Schölkopf, B.; and Zien, A., eds. 2006. *Semi-Supervised Learning*. The MIT Press. ISBN 9780262033589.

Chen, J.; Qiu, X.; Liu, P.; and Huang, X. 2018. Meta Multi-Task Learning for Sequence Modeling. In McIlraith, S. A.; and Weinberger, K. Q., eds., *AAAI*, 5070–5077. AAAI Press.

Cicek, S.; and Soatto, S. 2019. Unsupervised Domain Adaptation via Regularized Conditional Alignment. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. J. 2019. Class-Balanced Loss Based on Effective Number of Samples. In *CVPR*, 9268–9277. Computer Vision Foundation / IEEE.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Dou, Q.; de Castro, D. C.; Kamnitsas, K.; and Glocker, B. 2019. Domain Generalization via Model-Agnostic Learning of Semantic Features. In *NeurIPS*.

Du, C.; Sun, H.; Wang, J.; Qi, Q.; and Liao, J. 2020. Adversarial and Domain-Aware BERT for Cross-Domain Sentiment Analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4019–4028. Online: Association for Computational Linguistics.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1126–1135. PMLR.

Franceschi, L.; Frasconi, P.; Salzo, S.; Grazzi, R.; and Pontil, M. 2018. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1568–1577. PMLR.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; March, M.; and Lempitsky, V. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17(59): 1–35.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Jamal, M. A.; and Qi, G.-J. 2019. Task Agnostic Meta-Learning for Few-Shot Learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11711–11719.

Jiang, L.; Meng, D.; Mitamura, T.; and Hauptmann, A. 2014. Easy Samples First: Self-paced Reranking for Zero-Example Multimedia Search. *Proceedings of the 22nd ACM international conference on Multimedia*.

Kull, M.; and Flach, P. A. 2014. Patterns of dataset shift.

Kumar, A.; Sattigeri, P.; Wadhawan, K.; Karlinsky, L.; Feris, R.; Freeman, B.; and Wornell, G. 2018. Co-regularized Alignment for Unsupervised Domain Adaptation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Kumar, M. P.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 1189–1197.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. 2018. Learning to Generalize: Meta-Learning for Domain Generalization.

Li, H.; and Gong, M. 2017. Self-paced Convolutional Neural Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2110–2116.

Lin, C.; Bethard, S.; Dligach, D.; Sadeque, F.; Savova, G. K.; and Miller, T. A. 2020. Does BERT need domain adaptation for clinical negation detection? *Journal of the American Medical Informatics Association (JAMIA)*.

Lin, T.-Y.; Goyal, P.; Girshick, R. B.; He, K.; and Dollár, P. 2017. Focal Loss for Dense Object Detection. In *ICCV*, 2999–3007. IEEE Computer Society. ISBN 978-1-5386-1032-9.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning Transferable Features with Deep Adaptation Networks. arXiv:1502.02791.

Naik, A.; and Rosé, C. 2020. Towards Open Domain Event Trigger Identification using Adversarial Domain Adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nguyen, T. H.; and Grishman, R. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Nguyen, T. H.; and Grishman, R. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 886–891.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On First-Order Meta-Learning Algorithms. *ArXiv*, abs/1803.02999.

Raghu, A.; Raghu, M.; Bengio, S.; and Vinyals, O. 2019. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*.

Ren, Y.; Zhao, P.; Sheng, Y.; Yao, D.; and Xu, Z. 2017. Robust Softmax Regression for Multi-class Classification with Self-Paced Learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2641–2647.

Rogers, A.; Kovaleva, O.; and Rumshisky, A. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8: 842–866.

Schmidhuber, J. 1987. *Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook*. Diploma thesis, Technische Universitat Munchen, Germany.

Shu, J.; Xie, Q.; Yi, L.; Zhao, Q.; Zhou, S.; Xu, Z.; and Meng, D. 2019. Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. In *NeurIPS*.

Shu, R.; Bui, H. H.; Narui, H.; and Ermon, S. 2018. A DIRT-T Approach to Unsupervised Domain Adaptation. *CoRR*, abs/1802.08735.

Sun, Y.; Kamel, M. S.; Wong, A. K. C.; and Wang, Y. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognit.*, 40(12): 3358–3378.

Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H. S.; and Hospedales, T. M. 2018. Learning to Compare: Relation Network for Few-Shot Learning. In *CVPR*, 1199–1208. IEEE Computer Society.

Thrun, S.; and Pratt, L. Y., eds. 1998. *Learning to Learn.* Springer. ISBN 978-1-4615-5529-2.

Vinyals, O.; Blundell, C.; Lillicrap, T.; kavukcuoglu, k.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2005. ACE 2005 multilingual training corpus. In *Technical report, Linguistic Data Consortium*.

Wright, D.; and Augenstein, I. 2020. Transformer Based Multi-Source Domain Adaptation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xie, Q.; Hovy, E. H.; Luong, M.-T.; and Le, Q. V. 2019. Self-training with Noisy Student improves ImageNet classification. *CoRR*, abs/1911.04252.

Zadrozny, B. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, 114. Banff, Alberta, Canada: ACM. ISBN 1-58113-828-5.

Zellinger, W.; Grubinger, T.; Lughofer, E.; Natschläger, T.; and Saminger-Platz, S. 2017. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. arXiv:1702.08811.

Reproducibility Checklist

- **Source code**: We will publicly release the code and the necessary dependencies and libraries for this paper upon the acceptance of the paper.
- **Description of computing infrastructure used**: We use a single NVIDIA A100 GPU with 40GB memory in this work. PyTorch 1.7 is used to implement the models.
- **Average runtime for each approach**: Each epoch of our full model requires 45 minutes on average to finish. We train the models for 10 epochs.
- **Number of parameters in the model**: Our full model has 115 million parameters in total, fewer than 5% of which are optimized during training. We use the same model architecture across all adaptation scenarios.
- **Explanation of evaluation metrics**: We follow the standard performance metrics in prior work for model evaluation. In particular, precision, recall, and F1 scores for examples are used for performance metrics as in previous work (Nguyen and Grishman 2015, 2016). Our reported results are averages of five runs with different random seeds.
- **Bounds for each hyperparameter**: The bottleneck dimensionality of adapters is chosen among [48, 96, 128, 256]. Each objective head is a feed-forward network with 2 or 3 layers with hidden vector of size [100, 50] or [200, 100, 50], respectively. To train the proposed model, we use Adam optimizer with meta-train and meta-test learning rates α and γ both chosen from [5e-5, 1e-4, 5e-4, 1e-3, 5e-3], the mini-batch size from [50, 100, 150] of which 20% or 40% are unlabeled target data, and the meta-test balancing term β from [5, 2, 1, 0.5, 0.1].
- **Hyperparameter configurations for best-performing models**: In the best model, fixed pre-train BERT-base layers augmented by adapters with bottleneck size 96 are used as our feature encoder. All objective heads have 2 hidden layers. We use Adam optimizer with learning rate of 1e-4 for both meta-train and meta-test step, 100 for mini-batch size with 20% target data, and the meta-test balancing term is 2.
- **The method of choosing hyperparameter values and the criterion used to select among them**: We tune the hyperparameters for the proposed model using a random search. All the hyperparameters are selected based on the F1 scores on the development set of `bc` domain. The same hyperparameters from this fine-tuning are then applied for other domains.