Unsupervised Domain Adaptation for Event Detection via Meta Self-Paced Learning

Anonymous ACL submission

Abstract

A shift in data distribution can have a significant impact on performance of a model to detect important events in text. Recent methods addressing unsupervised domain adaptation for event detection task typically extracted domain-invariant representations through balancing between various objectives to align feature spaces between source and target domains. While effective, these methods are impractical as large-scale language models are drastically growing bigger to achieve optimal performance. To this end, we propose to leverage meta-learning framework to train a neural network-based self-paced learning procedure in an end-to-end manner. Our method, called Meta Self-Paced Domain Adaption (MSP-DA), effectively tune domain-specific hyperparameters including learning schedules, sample weights, and objective balancing coefficients, simultaneously throughout the learning process, by imitating the train-test dataset split based on the difficulties of source domain's samples. Extensive experiments demonstrate our framework substantially improves performance on target domains, surpassing state-of-the-art approaches. Detailed analyses validate our method and provide insight into how each domain affects the learned hyperparameters.

1 Introduction

Event detection (ED) task requires models trained to both locate event triggers in an event mention and classify them into one of the pre-specified event types. In unsupervised domain adaptation (UDA) setting, the problem becomes more complicated while also much more practical, in which the goal is to detect events in a different domain compare to the source domain of the labeled training dataset, given the additional access to easy-to-collect unlabeled data from the target domain. This poses a major challenge for standard systems due to both the intrinsic variation of linguistics (e.g., lexical shift, semantic shift) and the extrinsic factors such as how event-based datasets are collected and annotated. For example, a model trained to predict news events may easily recognize, from medical domain, "*died*" as an event, but would not be able to detect obvious events such as "*mutation*" or "*cancer*". Such a model may even fail to generalize to closer adaptation settings (e.g. news from different times and sources).

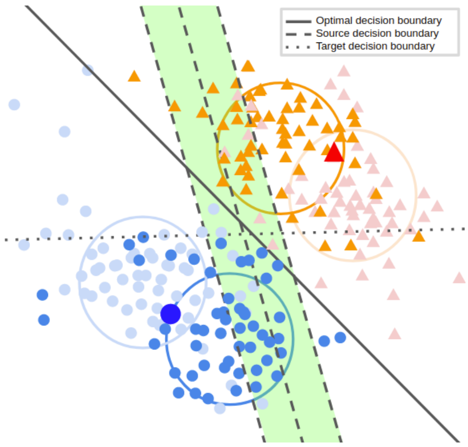

Figure 1: An example where domain shift between source domain (grey colors) and target domain (deep color) results in significant overlaps between high-loss regions of source decision boundary (lime) with high-density target clusters.

The majority of existing UDA approaches combined various training objectives to align different aspects of domain-specific extracted features. In particular, the most prominent approach is domain-adversarial neural network (DANN) (Ganin et al., 2016) that employs a domain-adversarial training procedure 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), discrepancy metrics (Long et al., 2015), 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. For example, state-of-the-art UDA method for ED is DAA (Ngo et al., 2021), which involves manually tuning weights of four auxiliary objectives, several of which even have their own respective hyperparameters.

Meta-learning (ML) framework is an effective solution for the problem of hyperparameter optimization (Franceschi et al., 2018; Behl et al., 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 et al., 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) approach (Kumar et al., 2010). Our framework, called Meta Self-Paced Domain Adaptation (MSP-DA), employs a neural-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 the combined loss' terms, including instance-wise weighting for the main classification task and layer-wise weighting for domain alignment 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, parameters and age threshold of the neural-SPL module are updated based on model's evaluation performance in meta-test step, resulting in tuned weighting coefficients and learning schedules similar to that of a standard hyperparameter tuning process. To our knowledge, this is the first work to devise a neural network-based SPL method, in which both the sample weightings/selections and the age hyperparameter are dynamically optimized, generalizing previous works which require heuristic age schedule and complicated mathematical derivation for the corresponding instance weighting.

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 the 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 SSL methods (Chapelle et al., 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 provide extensively evaluation of the proposed framework for event detection task on ACE-05 dataset, along with additional results for sentiment analysis task on FDU-MTL 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.

2 Related Work

Event Detection and Unsupervised Domain Adaptation Previous line of research on ED mostly addressed the standard supervised learning setting (Li et al., 2013; Nguyen and Grishman, 2016a; Yang and Mitchell, 2016; Nguyen et al., 2021), with cross-domain evaluation (Nguyen and Grishman, 2016b; Hong et al., 2018). Recently, several works have focused on the UDA problem of the simpler Event Identification task (Naik and Rosé, 2020) using domain-adversarial training. Ngo et al. (2021) further incorporated shared-private architecture efficiently through domain-specific adapters (Houlsby et al., 2019) to solve UDA ED task.

Sample Weighting There are two main research directions to adaptively output weight of a sample during training process: 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 et al., 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 There are three main categories of modern ML 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's 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 et al., 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).

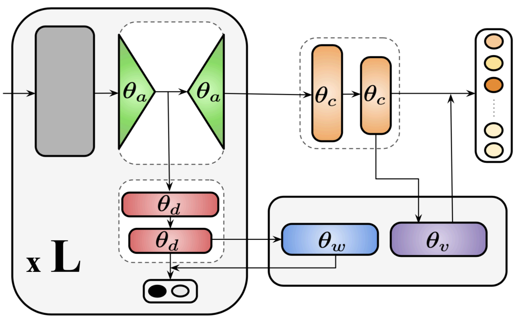

Figure 2: Architecture overview. (**gray**) Fixed BERT layers. (**green**) Adapter layers, bottleneck outputs of which are then fed into domain classifier heads (**red**). The neural-SPL module consists of instance-wise weighting head (**purple**) for main task classification (**orange**) and a layer-wise balancing head (**blue**) for domain adversarial training.

3 Model

We denote the source dataset $\mathbf{S} = \{(x_i^{\mathbf{s}}, y_i^{\mathbf{s}})\}_{i=1}^{N^{\mathbf{s}}}$ consisted of $N^{\mathbf{s}}$ samples and an unlabeled set of $N^{\mathbf{t}}$ samples $\mathbf{T} = \{x_i^{\mathbf{t}}\}_{i=1}^{N^{\mathbf{t}}}$ drawn from target domain. Label space $\mathbf{Y} = \{1, 2, \cdots, K\}$ of K classes is shared across domains.

Our model's feature encoder is a fixed pretrained 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 heads. Following prior work (Ngo et al., 2021), low dimensional output from each layer's adapter is used by a separate DANN head for domain adversarial training. Our neural-SPL module consists of two weighting mechanisms: an instance-wise $f_v(\theta_v) : \mathbb{R} \rightarrow \mathbb{R}$ which weighs the contribution v_i of each example based on the its classification loss and a learnable age parameter λ_a; and 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" w^l of which the corresponding layer l should be aligned. 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 tune the "hyperparameters" from the neural-SPL module. The overall architecture is presented in Fig. 2.

3.1 Meta Self-Paced Learning

Self-Paced Learning Kumar et al. (2010) devised Self-Paced Learning method that extends Curriculum Learning (Bengio et al., 2009) to jointly learn the model and its curriculum, circumventing the need for an ad-hoc implementation of easiness based on some predetermined heuristics. Specifically, SPL employs an age hyperparameter λ_a that represents the current learning pace of the model. The objective is then reformulated as a weighted loss where each instance's contribution is thresholded by λ_a as follow:

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

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.

Adaptive SPL via Meta-Learning The advantage of incorporating SPL into a ML framework is two-fold. First, ML provides a way to adaptively tune the highly sensitive λ_a, alleviating the need for manually devising an age scheduler. At the same time, SPL helps address the lack of clean validation data, by splitting the source domain instances of the current mini-batch into two disjoint sets based on the age value λ_a. The easy samples are used for meta-train step, in which the objective consists of a domain adversarial loss and a SPL-weighted classification loss:

$$\mathcal{L}_{tr}\left(\mathbf{S}_{tr}, \mathbf{T}; \theta\right) = \mathcal{L}_{ce}\left(\mathbf{S}_{tr}; \theta_a, \theta_c\right) + \mathcal{L}_d\left(\mathbf{S}_{tr}, \mathbf{T}; \theta_a, \theta_d\right) \quad (2)$$

$$v_i = f_v \circ \max(0, \frac{-l_i}{\lambda_a} + 1);\ \mathcal{L}_{ce}\left(\mathbf{S}_{tr}\right) = \sum_{x_i, y_i \in \mathbf{S}_{tr}} v_i l_i \quad (3)$$

where $l_i = l(x_i, y_i; \theta)$ is the loss of each sample and $\mathcal{L}_d$ is the weighted domain adversarial objective that is explained in the following section. f_v is a small feed-forward network with sigmoid as final activation function to guarantee the resulting weights located in the interval of [0, 1], and with no bias so that the 0-valued inputs will also correspond to outputs of the same value.

Typically, k gradient steps are applied to approximate the optimal solution that minimizes the current meta-train objective. Because of the sizeable transformer encoder, a high value of k will cost serious computation overhead. Thus, we decide to use $k = 1$, from which we observe no significant performance loss:

$$\bar{\theta} = \theta - \alpha \nabla_\theta (\mathcal{L}_{ce}\left(\theta_a, \theta_c\right) + \mathcal{L}_d\left(\theta_a, \theta_d\right)) \quad (4)$$

where α is meta-train learning rate. 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:

$$\mathcal{L}_{ts}\left(\mathbf{S}_{ts}; \bar{\theta}\right) = \sum_{x_i, y_i \in \mathbf{S}_{ts}} (x_i, y_i; \bar{\theta}) \quad (5)$$

This acts as a hard, distinct domain that provides tuning signals for guiding model updates of both model's parameters in θ and hyperparameters v_i and λ_a.

3.2 Balancing domain adversarial objectives

The survey presented by (Rogers et al., 2020) provides a detailed probing and understanding of how the different layer-block of BERT encodes different types of information. 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:

$$\mathcal{L}_d = \sum_{l=1}^{L} w^l \mathcal{L}_d^l(\mathbf{z}_d^l, \mathbf{y}_d; \theta_d^l) \quad (6)$$

where each $\mathcal{L}_d^l$ is an adversarial term of a different DANN , taking adapter representations $\mathbf{z}_d^l$ of layer lth and domain labels $\mathbf{y}_d$ as inputs. These losses are weighted by a set of coefficients $\{w^l\}$ that corresponds to how important it is for the representations at the respective layer to be aligned. Following standard learning procedure, they would be hyperparameters that required careful tuning for each specific domain, which would be impractical (in our setting, there would be a total of 12 hyperparameters). To address the above issue, we employ a small feed-forward network f_w with a final softmax layer to output the relative layer-wise weights:

$$\mathbf{W} = [w^0, \cdots, w^{L-1}] = f_w(\mathbf{Z}_d; \theta_w) \quad (7)$$

where $\mathbf{Z}_d \in \mathbb{R}^{L \times d_a}$ 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. As θ_w is updated throughout the ML process, $\mathbf{W}$ is dynamically tuned to maintain high performance on meta-test set while domain-adversarial training makes representations across layers domain-invariant.

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

$$\operatorname*{argmin}_{\theta} \beta \mathcal{L}_{ts}\left(\bar{\theta}\right) + \mathcal{L}_{tr}\left(\theta\right);\ \operatorname*{argmin}_{\theta_w, \theta_v, \lambda_a} \mathcal{L}_{ts}\left(\bar{\theta}\right) \quad (8)$$

where β is meta-test balancing term. The second term in Eq. 8 is the result of passing the weights computed by neural-SPL module in Eq. 3 and 7 into Eq. 2 as pre-determined values, not learnable variables.

3.3 Incorporating Pseudo Label

Pseudo Label 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. 2 in meta-train step, in which they are weighted and thresholded by neural-SPL module using the same λ_a as source data: $\mathcal{L}_{ce}\left(\mathbf{S}_{tr}, \overline{\mathbf{T}}\right) = \sum_{x_i, y_i \in \mathbf{S}_{tr} \cup \overline{\mathbf{T}}} v_i l_i$, 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, neural-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 ML framework, forcing them to be consistent with meta-target domain.

4 Experiments

4.1 Datasets, Settings, and Baselines

We evaluate the proposed model on ED task in UDA setting. In addition, we also demonstrate the generalization of our framework when applying to multi-domain sentiment analysis (SA) task.

ACE-05 (Walker et al., 2005) A densely annotated corpus collected from 5 different domains. Two of which are used as source data, while each of the rest is a target domain for an adaptation setting. 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 pre-defined 34 event types (including 1 negative type).

FDU-MTL (Liu et al., 2017) A dataset included reviews from 16 domains for binary sentiment classification task. In each adaptation setting, a single domain is assigned as the target with unlabeled data while the other 15 are labeled source. Given the contextual sequence computed by models from a review, we use the first token [CLS] as the feature to predict its positive or negative sentiment.

Information about each dataset for UDA setting is described in Appendix B.

ED baselines We provide a comprehensive comparison of our proposed method with multiple baselines from 3 categories: (`No Weighting`) models that do not leverage any weighting mechanism. **BERT** is only fine-tuned on only labeled source domain, whereas **BERT+DANN** follows the standard adversarial training; (`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 **Dom-Cls**, 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. 1. Finally, we include results from recent approach **DAA** (Ngo et al., 2021), in which three adapters were employed to create shared-private representations through layer-wise domain adversarial training, Wasserstein-based data selection, similarity constraint, and a self-supervised auxiliary task.

We include details regarding implementations, trainings and evaluations of our experiments in Appendix A.

SA baselines **ASP-MTL** (Liu et al., 2017) and **DAEA** (Cai and Wan, 2019) are LSTM-based approaches, while **BERT** and **BERT+DANN** are the same as in ED baselines. Finally, **BertMasker** (Yuan et al., 2021) is the state-of-the-art approach that learns to explicitly mask domain-related words from text, resulting in domain-agnostic sentences.

4.2 Main Results

Event Dectection The first three row-blocks of Table 1 present the performances of the above baselines in each domain adaptation scenario. **BERT+DANN** only provides slight improvement for domain `bc` compare to **BERT**, while significantly degrades model's performances on the other two. Similarly, 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, the change in data distribution across domains results in **Class-Balanced**'s low domain adaptation ability. **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, **Dom-Cls** requires computing a specific curriculum for each domain, yet performs worse than the dynamic curriculum imposed by **SPL**. Finally, compared to the state-of-the-art **DAA**, **MSP-DA** provides a

System	In-domain(`bn+nw`)			Out-of-domain (`bc`)			Out-of-domain (`cts`)			Out-of-domain (`wl`)			
	P	R	F	P	R	F	P	R	F	P	R	F	**aF1**
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	66.3
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	63.4
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	52.1
Focal	78.2	77.6	77.9	71.7	72.9	72.2	72.9	68.5	70.1	64.8	54.2	59.0	67.1
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	61.0
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	64.9
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	63.7
DAA	79.7	75.7	77.7	78.5	75.6	**76.9**	78.4	73.2	75.6	66.2	60.3	63.1	71.9
MSP-DA	75.4	80.0	77.7	76.2	75.5	75.8	75.3	76.8	**76.1**	70.8	59.9	**64.8**	**72.2**

Table 1: UDA performances for ED task on **ACE-05** test datasets. **aF1** is the average out-of-domain F1 score.

System	MR	Appr.	Baby	Books	Cam.	DVD	Elec.	Hlth.	IMDB	Kitc.	Magz.	Musics	Softw.	Sport	Toys	Video	**aAcc**
ASP-MTL	76.7	87.0	88.2	84.0	89.2	85.5	86.8	88.2	85.5	86.2	92.2	82.5	87.2	85.7	88.0	84.5	86.1
DAEA	77.0	89.0	92.3	89.0	92.0	88.3	91.8	89.8	90.8	90.3	**96.5**	88.0	92.8	90.8	91.8	92.3	90.2
BERT	90.5	90.8	90.3	91.3	91.5	89.0	91.3	91.3	91.3	90.0	88.5	90.3	90.5	92.0	90.8	92.0	90.7
BERT+DANN	90.5	91.8	92.5	90.8	90.0	91.3	90.5	90.8	91.0	91.8	91.0	90.5	91.0	90.5	90.3	90.3	90.9
BertMasker	83.8	92.3	**92.8**	93.0	92.8	89.3	**93.3**	**95.3**	86.0	90.8	94.5	89.5	93.0	92.5	**93.8**	91.3	91.5
MSP-DA	**93.3**	**93.1**	92.5	**93.2**	**93.3**	**92.4**	93.1	93.2	**93.4**	**93.0**	93.1	**92.7**	**93.1**	**93.3**	93.5	**92.8**	**93.0**

Table 2: UDA performances for SA task on **FDU-MTL** test datasets. **aAcc** is the average accuracy score across all domains.

significant 1.7 points increase when adapting to hardest domain `wl`, and achieving on average 0.3 points higher in F1 score. While it is a marginal boost, we would like to note that DAA leverages multiple encoders through different constraints and auxiliary tasks to address domain shift problem. In contrast, our work focuses on simple domain adaptation approaches (sample weighting and domain adversarial training), and is effective because of good tuned hyperparameters and training schedule. Moreover, the two methods are orthogonal and can be complementary to each other for further improvements.

Sentiment Analysis SA results are presented in Table 2. While simple model using contextual embedding **BERT** outperforms all previous LSTM-based methods, we again observe little to no improvement applying domain adversarial training naively with it. In contrast, our framework achieves the best performance for 13 out of 16 review domains, surpassing the current state-of-the-art method **BertMasker** by 1.5 points on average.

4.3 Ablation Study

In the first row-block of Table 3, 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: **MSP-DA–mSPL** follows the normal SPL process to produce the weighting coefficients and train-test datasets for ML; **MSP-DA–DANN** trains only on source domain without utilizing unlabeled target data for domain adversarial objective; and **MSP-DA–PL** in which no pseudo-labels are leveraged for training. In general, our full model outperforms all variants across domains, even in the in-domain setting, which confirms the superiority and flexibility provided by the jointly optimized pacing and weights from our neural-SPL module. Especially for `wl` domain, domain adversarial training in MSP-DA manages to improve more than 8 F1 points.

Meta-test Selection To examine the correctness of our assumption, we augment the data selection process for meta domains in **Random** and **Reverse** variants. 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 as shown in 3. 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 ML 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.

System	In-domain(`bn+nw`)			Out-of-domain (`bc`)			Out-of-domain (`wl`)		
	P	R	F	P	R	F	P	R	F
MSP-DA – mSPL	74.5	79.7	77.0	77.5	72.0	74.6	64.1	51.9	57.4
MSP-DA – DANN	74.3	80.3	77.2	75.7	72.9	74.2	61.6	51.9	56.3
MSP-DA – PL	77.8	75.1	76.4	75.1	73.5	74.3	62.6	52.4	57.0
MSP-DA (Random)	73.0	76.4	74.7	75.6	73.3	74.4	61.0	50.3	55.0
MSP-DA (Reverse)	77.7	75.0	76.3	78.2	70.6	74.2	65.0	50.7	57.0
MSP-DA (Ours)	75.4	80.0	**77.7**	76.2	75.5	**75.8**	70.8	59.9	**64.8**

Table 3: Performances for Ablation Study

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)	76.2	75.5	**75.8**	70.8	59.9	**64.8**

Table 4: Performances for Age Hyperparameter Analysis

4.4 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

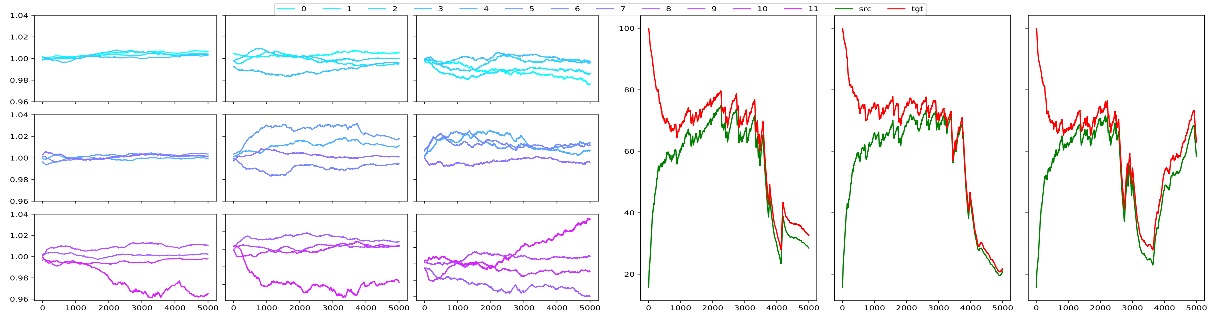

Figure 3: Three columns in each subplot correspond to domain `bc`, `cts`, `wl`, respectively. (**Left**) Layer-wise DANN weights at each training step. (**Right**) source and target age percentiles at each training step.

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)	76.2	75.5	**75.8**	70.8	59.9	**64.8**

Table 5: Performances for DANN Weighting Analysis

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 neural-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 4, we examine how different values and schedules of age hyperparameter affect performances on `bc` and `wl` domains. The **Fixed (p)** settings with $\mathbf{p} \in [25, 50, 75]$ 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 ML framework.

λ_a Visualization 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 initially, 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. 8 implies, which we will delve deeper into in the discussion 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 ML 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.

4.5 Balancing Domain Adversarial Losses

Previous works have observed that the weight of DANN in the combined objective has a significant impact on 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. 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. 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 is important for a different type of task as described 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. 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.

5 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:

$$\mathcal{L}_{ts}\left(\theta - \alpha\mathcal{L}'_{tr}(\theta)\right) = \mathcal{L}_{ts}\left(\theta\right) + \frac{\partial\mathcal{L}_{ts}\left(\theta\right)}{\partial\theta}\left(-\alpha\frac{\partial\mathcal{L}_{tr}\left(\theta\right)}{\partial\theta}\right) \quad (9)$$

Denoting $\mathbf{G} = \frac{\partial\mathcal{L}_{ts}(\theta)}{\partial\theta} \cdot \frac{\partial\mathcal{L}_{tr}(\theta)}{\partial\theta}$ and plugging Eq. 9 into the final objective to update main model's parameters from Eq. 8 results in the following optimization problem:

$$\underset{\theta}{\operatorname{argmin}}\ \mathcal{L}_{tr}\left(\theta\right) + \mathcal{L}_{ts}\left(\theta\right) - \beta\alpha\mathbf{G} \quad (10)$$

The third term in Eq. 10 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. 10 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 ML framework affects the values of neural-SPL module's parameters $(\theta_w, \theta_v, \lambda_a)$ in our model. Plugging Eq. 9 into the gradient of λ_a, we have:

$$\frac{\partial\mathcal{L}_{ts}\left(\bar{\theta}\right)}{\partial\lambda_a} = -\alpha\frac{\partial\mathcal{L}_{ts}\left(\theta\right)}{\partial\theta}\cdot\frac{\partial^2\mathcal{L}_{tr}\left(\theta\right)}{\partial\theta\partial\lambda_a} = -\alpha\mathbf{G}\cdot\frac{\partial f_v(\lambda_a)}{\partial\lambda_a} \quad (11)$$

From Eq. 11, we see that the multiplicative factor $\mathbf{G}$ also controls how the value of λ_a changes throughout the ML 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 push λ_a higher and allow 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 et al., 2018; Raghu et al., 2019)

6 Conclusion

We present a novel ML framework for UDA setting that achieves state-of-the-art performance on ED task. In particular, a neural-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 hyperparameter tuning. In the future, we intend to apply our approach to other domains and tasks while incorporating different novel domain adaptation regularization methods.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

Sébastien M R Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. 2020. learn2learn: A library for Meta-Learning research.

Harkirat Singh Behl, Atilim Gunecs Baydin, and Philip H. S. Torr. 2019. Alpha maml: Adaptive model-agnostic meta-learning. *ArXiv*, abs/1905.07435.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Konstantinos Bousmalis, George Trigeorgis, N. Silberman, Dilip Krishnan, and D. Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*.

Yitao Cai and Xiaojun Wan. 2019. Multi-domain sentiment classification based on domain-aware embedding and attention. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4904–4910. International Joint Conferences on Artificial Intelligence Organization.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. 2006. *Semi-Supervised Learning*. The MIT Press.

Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *AAAI Conference on Artificial Intelligence*, pages 5070–5077. AAAI Press.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. 2019. Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277. Computer Vision Foundation / IEEE.

Q. Dou, Daniel Coelho de Castro, K. Kamnitsas, and B. Glocker. 2019. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR.

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

Yu Hong, Wenxuan Zhou, jingli zhang jingli, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Muhammad Abdullah Jamal and Guo-Jun Qi. 2019. Task agnostic meta-learning for few-shot learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11711–11719.

Lu Jiang, Deyu Meng, T. Mitamura, and A. Hauptmann. 2014. Easy samples first: Self-paced reranking for zero-example multimedia search. *Proceedings of the 22nd ACM international conference on Multimedia*.

Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, Bill Freeman, and Gregory Wornell. 2018. Co-regularized alignment for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. 2018. Learning to generalize: Meta-learning for domain generalization.

Hao Li and Maoguo Gong. 2017. Self-paced convolutional neural networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2110–2116.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV*, pages 2999–3007. IEEE Computer Society.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning transferable features with deep adaptation networks.

Aakanksha Naik and Carolyn Rosé. 2020. Towards open domain event trigger identification using adversarial domain adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nghia Trung Ngo, Duy Phung, and Thien Huu Nguyen. 2021. Unsupervised domain adaptation for event detection using domain-specific adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, page 4015–4025. Association for Computational Linguistics.

Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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

Thien Huu Nguyen and Ralph Grishman. 2016b. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alex Nichol, Joshua Achiam, and J. Schulman. 2018. On first-order meta-learning algorithms. *ArXiv*, abs/1803.02999.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. 2019. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*.

Yazhou Ren, Peng Zhao, Yongpan Sheng, Dezhong Yao, and Zenglin Xu. 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*, pages 2641–2647.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

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 *Advances in Neural Information Processing Systems*.

Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. 2018. A DIRT-T approach to unsupervised domain adaptation. *CoRR*, abs/1802.08735.

Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognit.*, 40(12):3358–3378.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208. IEEE Computer Society.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2005. Ace 2005 multilingual training corpus. In *Technical report, Linguistic Data Consortium*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qizhe Xie, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Self-training with noisy student improves imagenet classification. *CoRR*, abs/1911.04252.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jianhua Yuan, Yanyan Zhao, Bing Qin, and Ting Liu. 2021. Learning to share by masking the non-shared for multi-domain sentiment classification.

A Implementation Details

All models are implemented in Pytorch. We leverage pre-trained BERT-base models and checkpoints from Huggingface repository. (Wolf et al., 2020). Meta-learning process is implemented following ANIL algorithm in (Arnold et al., 2020).

Bounds for each hyperparameter Adapter layers injected after every feed-forward sub-blocks have bottleneck feed-forward architecture with down-sampled dimension chosen among [48, 96, 128]. All of the downstream heads are implemented as feed-forward networks with activation functions between layers. Each weighting net of neural-SPL module is a feed-forward network with 2 or 3 layers with hidden vectors 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].

Method of choosing hyperparameter values We tune the hyperparameters for the proposed model using a random search. All 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.

Best hyperparameter configuration 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 a 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. Our reported results are averages of five runs using the best hyperparameter configuration with different random seeds.

B Data Settings

We provide statistics of each domain in UDA setting for **ACE-05** and **FDU-MTL** in Table 6 and Table 7, respectively.

For **ACE-05** dataset, we gather data from two closely related domains, `bn` and `nw`, to create a sizable source domain dataset, 80% of which are used for training whilst the rest are used as 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. All of the considered models' hyperparameters are only tuned based on `bc` domain.

Domains	Train	Unlabeled	Test
bn+nw	38644	N/A	9661
bc	N/A	3130	12520
cts	N/A	2885	10972
wl	N/A	3424	12767

Table 6: Statistics of **ACE-05**'s domains in UDA setting.

For **FDU-MTL** dataset, each of the 16 domains has a test set of 400 samples. The amount of training labeled and unlabeled data vary across domains, ranging from 1400 to 2000 samples. In each adaptation setting, a single domain is designated as the target domain while its unlabeled data are used in training set together with labeled data from the other 15 domains.

Domains	Train	Unlabeled	Test
Books	1400	2000	400
Elec.	1398	2000	400
DVD	1400	2000	400
Kitchen	1400	2000	400
Apparel	1400	2000	400
Camera	1397	2000	400
Health	1400	2000	400
Music	1400	2000	400
Toys	1400	2000	400
Video	1400	2000	400
Baby	1300	2000	400
Magaz.	1370	2000	400
Soft.	1315	475	400
Sport	1400	2000	400
IMDb	1400	2000	400
MR	1400	2000	400

Table 7: Statistics of the 16 domains in **FDU-MTL**