WIND: Weighting Instances Differentially for Model-Agnostic Domain Adaptation

Xiang Chen, Yue Cao, Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
Center for Data Science, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{caspar,yuecao,wanxiaojun}@pku.edu.cn

Abstract

Domain Adaptation is a fundamental problem in machine learning and natural language processing. In this paper, we study the domain adaptation problem from the perspective of instance weighting. Conventional instance weighting approaches cannot learn the weights which make the model generalize well in target domain. To tackle this problem, inspired by meta-learning, we formulate the domain adaptation problem as a bi-level optimization problem, and propose a novel differentiable model-agnostic instance weighting algorithm. Our proposed approach can automatically learn the instance weights instead of using manually designed weighting metrics. To reduce the computational complexity, we adopt the second-order approximation technique during training. Experimental results[1] on three different NLP tasks (Sentiment Classification, Neural Machine Translation and Relation Extraction) illustrate the efficacy of our proposed method.

1 Introduction

Domain shift is a challenging problem which is commonly encountered in Natural Language Processing (NLP). Due to the data distribution discrepancy between source and target domain, the model trained on the data from source domain may fail to achieve satisfying performance in target domain. Therefore we face the *domain adaptation* problem. In some real-world situations, we may only focus on the performance of our model on a specific domain. To maintain the performance, we need labeled training data for supervised learning. However, we often cannot collect enough labeled training data relevant to the domain we are interested in (in-domain). Thus, we need to introduce more labeled data from other different domains (out-of-domain). We aim to leverage the general knowledge from out-of-domain dataset to enhance the in-domain performance of our model.

We consider a specific domain adaptation scenario in this work, where we have a few labeled in-domain training data and meanwhile we have sufficient labeled out-of-domain training data from other general domains.

Training on these two datasets jointly is a straightforward solution for this scenario, but not all samples from out-of-domain dataset has equal effect during the training procedure. Several studies (Koehn and Knowles, 2017) on neural machine translation (NMT) task show that, out-of-domain instances relevant to the in-domain data are beneficial while the instances irrelevant to the in-domain data may be even harmful to the translation quality. Apart from that, for sentiment classification task, some general expressions such as "I'm truly impressed by the design." may appear in all domains. Taking them as training samples can help the model to learn general syntactic and semantic knowledge, which improves the cross-domain sentiment classification performance. But using examples like "This chair is solid." (negative sentiment, furniture domain) may reduce the accuracy of classifying "This knife is solid." (positive sentiment, kitchen domain), because "solid" has different meanings in these two domains. Any domain-specific expression like this would probably introduce some noise. So it is essential to find a suitable strategy to measure the importance of each training sample.

There are many instance weighting (or instance selection) methods to tackle this problem. They assign a weight to each instance and transform the loss function to a weighted-sum formula. Most of the conventional methods (Jiang and Zhai, 2007; Gretton et al., 2006, 2009; Axelrod et al., 2011; Wang et al., 2017; Zhang and Xiong, 2018; Wang et al., 2019; Dou et al., 2020) propose different

[1]The code is available at https://github.com/CasparSwift/WIND

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2366–2376
August 1–6, 2021. ©2021 Association for Computational Linguistics

kinds of manually designed metrics to calculate the weights of instances. The core idea of these methods is to weight the instances according to their importance and similarity to the target domain. However, in our domain adaptation setting, the size of out-of-domain corpus is much larger than that of in-domain corpus. The weights learned by the previous methods may be biased to the out-of-domain data, which would unavoidably result in poorer performance on the in-domain data. In this paper, we seek to automatically learn the weights which make the model generalize well on the unbiased in-domain data.

Inspired by Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), we introduce another unbiased subset from in-domain data which serves as a query set. We propose a novel model-agnostic differentiable instance weighting approach named "**WIND**" (means **W**eighting **IN**stances **D**ifferentially) which is a general framework and can be applied to all tasks in our domain adaptation settings. Moreover, we hope to get rid of manually designed metrics and let the weights to be differentiable. To reduce the computational complexity, we adopt a second-order derivation approximation approach for calculating the gradient of weights. We conduct plenty of experiments on datasets from three representative NLP tasks: **sentiment classification**, **machine translation** and **relation extraction**. The results show that our proposed method substantially outperforms several strong baselines.

The contributions of our work can be summarized as follows:

- We propose a novel differentiable instance weighting algorithm for domain adaptation, which learns the weights of instances with gradient descent and does not need manually designed weighting metrics.
- We adopt a second-order approximation technique to speed up the model training.
- We conduct experiments on three typical NLP tasks: Sentiment Classification, Machine Translation and Relation Extraction. Experiment results demonstrate the effectiveness of the proposed method. Code will be released.

2 Methodology

In this section, we first formulate our domain adaptation problem and introduce some notations. Then we present the proposed gradient-based model-agnostic instance weighting framework for our setting and introduce the method to approximate the second-order derivation of query loss. Finally, we discuss some optimization details of our method.

2.1 Problem Formulation

Let $\mathcal{D}_{train}$, $\mathcal{D}_{dev}$ and $\mathcal{D}_{test}$ denote our train, development and test datasets respectively. We use $\mathcal{D}_{train}$ for model training, $\mathcal{D}_{dev}$ for hyperparameter tuning and $\mathcal{D}_{test}$ for model testing. Both $\mathcal{D}_{dev}$ and $\mathcal{D}_{test}$ are in-domain data. Differently, $\mathcal{D}_{train}$ consists of sufficient labeled out-of-domain training samples $\mathcal{D}_{out} = \{(x_i, y_i)\}_{i=1}^{m}$ and a few labeled in-domain training samples $\mathcal{D}_{in} = \{(x_i, y_i)\}_{i=1}^{n}$, where $n << m$.

How to efficiently utilize $\mathcal{D}_{in}$ is the key to better domain transfer. To tackle this problem, in this paper we first sample an *in-domain train* subset $\mathcal{D}_{it} = \{(x_i, y_i)\}_{i=1}^{n_1}$ from $\mathcal{D}_{in}$ and we assign a scalar weight w_i to each instance $(x_i, y_i) \in \mathcal{D}_{it} \cup \mathcal{D}_{out}$. We hope that during training, the model can find the optimal weight $\boldsymbol{w} = (w_1, ..., w_{n_1+m})$ by itself. For this purpose, the weight $\boldsymbol{w}$ should be differentiable and can be optimized by gradient descent. Moreover, we denote the deep neural network (DNN) as a function $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y}$ which is parameterized by $\boldsymbol{\theta}$ and maps x_i from the input space to the label space. In our instance weighting setting, the training loss follows a weighted-sum formula:

$$\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}) = \frac{1}{n_1 + m} \sum_{\substack{(x_i, y_i) \in \\ \mathcal{D}_{it} \cup \mathcal{D}_{out}}} w_i \ell(f_{\boldsymbol{\theta}}(x_i), y_i) \tag{1}$$

where ℓ denotes the loss function, which can be any kind of loss such as cross entropy loss for classification tasks, or label-smoothed cross entropy loss for machine translation.

Jointly optimizing $\boldsymbol{\theta}$ and $\boldsymbol{w}$ using Eq. 1 is a straightforward solution. However, due to the data distribution discrepancy of in-domain and out-of-domain datasets, learning $\boldsymbol{w}$ directly from $\mathcal{D}_{it} \cup \mathcal{D}_{out}$ by Eq. 1 may introduce bias. What we expect is that the model trained on $\boldsymbol{w}$ can be generalized to the in-domain data. In order to achieve this goal, inspired by MAML (Finn et al., 2017), we propose to sample another subset $\mathcal{D}_q = \{(x_i, y_i)\}_{i=1}^{n_2}$ named *query set* from $\mathcal{D}_{in}$. We propose to use this query set to optimize $\boldsymbol{w}$. Specifically, we aim to obtain a weight vector $\boldsymbol{w}$

which minimizes the loss on $\mathcal{D}_q$:

$$\mathcal{L}_q(\boldsymbol{\theta}) = \frac{1}{n_2} \sum_{(x_i,y_i)\in\mathcal{D}_q} \ell(f_{\boldsymbol{\theta}}(x_i), y_i) \quad (2)$$

Note that we only weight $n_1 + m$ instances from $\mathcal{D}_{it} \cup \mathcal{D}_{out}$, so $\mathcal{L}_q(\boldsymbol{\theta})$ has a standard form, not a weighted-sum form.

Given a specific $\boldsymbol{w}$, we can train a model with the loss $\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w})$ and then get the optimized parameter $\boldsymbol{\theta}^*$. We aim to minimize the loss on the query set given $\boldsymbol{\theta}^*$. Therefore, our problem can be formulated as the following bilevel optimization problem (Colson et al., 2007):

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \mathcal{L}_q(\boldsymbol{\theta}^*) \\ \text{s.t.} \quad & \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}) \end{aligned} \quad (3)$$

This bilevel formulation arises in many meta-learning or hyperparameters optimization (HPO) problems (Bergstra et al., 2013; Franceschi et al., 2018), where the optimization of the outer objective $\mathcal{L}_q$ depends on the optimization of inner objective $\mathcal{L}_{train}$. In fact, Eq. 3 is a special case of hyperparamter optimization, because $\boldsymbol{w}$ can be viewed as special hyperparameter of our model. In Section 2.2, we will introduce our proposed algorithm to solve this nested formulation.

2.2 Optimization of Instance Weights

It is difficult to directly solve the above-mentioned bilevel optimization problem because of its high complexity of solving the inner objective. There are many gradient-based methods (Maclaurin et al., 2015; Franceschi et al., 2018) to solve this problem. However, unlike typical hyperparameters such as learning rate, the instance weight $\boldsymbol{w}$ is of high dimension. It is even harder to optimize this problem in our setting.

Inspired by the optimization techniques used in model-agnostic meta-learning (MAML) (Finn et al., 2017), we split the training procedure of each iteration into the following three steps.

2.2.1 Pseudo Update

Firstly, we sample two mini-batches of data from $\mathcal{D}_{it} \cup \mathcal{D}_{out}$ and $\mathcal{D}_q$ respectively, Then we compute the model's parameters after one step update by the gradient of $\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w})$ respect to $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}) \quad (4)$$

where β denotes the learning rate of this step.

This step is just "pseudo update". After updating, we do not replace original parameters $\boldsymbol{\theta}$ with the adapted parameters $\widehat{\boldsymbol{\theta}}$. Instead, we store both $\boldsymbol{\theta}$ and $\widehat{\boldsymbol{\theta}}$. We will use $\widehat{\boldsymbol{\theta}}$ to calculate the gradient of $\boldsymbol{w}$ in the second step. So in our proposed algorithm, $\widehat{\boldsymbol{\theta}}$ is just an intermediate variable which will be abandoned in the end of current iteration.

2.2.2 Instance Weight Update

Then we calculate the instance weights $\boldsymbol{w}$ using $\widehat{\boldsymbol{\theta}}$. In this step, our goal is to find an optimal $\boldsymbol{w}^*$. We expect $\boldsymbol{w}^*$ to have the property that: optimizing one step by $\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}^*)$ should result in a decrease of query loss. In other words, we expect $\boldsymbol{w}^*$ to minimize the loss on the query set after one step update:

$$\begin{aligned} \boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}}) \\ &= \arg\min_{\boldsymbol{w}} \mathcal{L}_q(\boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w})) \end{aligned} \quad (5)$$

Note that this is an approximation for the outer objective of Eq. 3. Theorectically, we can perform gradient descent for many steps to find $\boldsymbol{w}^*$. But it is time-consuming. So basically we optimize $\boldsymbol{w}$ with the gradient of $\mathcal{L}_q(\widehat{\boldsymbol{\theta}})$ with respect to $\boldsymbol{w}$ for only one step:

$$\widehat{\boldsymbol{w}} = \boldsymbol{w} - \gamma \cdot \nabla_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}}) \quad (6)$$

where γ denotes the learning rate of $\boldsymbol{w}$.

We take $\widehat{\boldsymbol{w}}$ as an approximation of $\boldsymbol{w}^*$. Using multiple gradient updates for $\boldsymbol{w}$ is a straightforward extension of this step, which will lead to more accurate approximation for $\boldsymbol{w}^*$ while increasing the computational complexity at the same time.

2.2.3 Final Update

In the previous two steps, we have an approximately optimal weights $\widehat{\boldsymbol{w}}$. We use it for actual update for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \widehat{\boldsymbol{w}}) \quad (7)$$

Current iteration ends after this step. As mentioned before, $\widehat{\boldsymbol{\theta}}$ will be abandoned, but we can choose whether $\widehat{\boldsymbol{w}}$ to be abandoned or not. This will be further discussed in Section 2.4.3.

2.3 Second-Order Derivation Approximation

There is a fatal problem when calculating the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}})$ in the instance weight update (Sec-

tion 2.2.2). We apply the chain rule to Eq. 6:

$$\begin{aligned}\widehat{\boldsymbol{w}} &= \boldsymbol{w} - \gamma \cdot \nabla_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}}) \\ &= \boldsymbol{w} - \gamma \cdot \nabla_{\widehat{\boldsymbol{\theta}}} \mathcal{L}_q \cdot \nabla_{\boldsymbol{w}} \widehat{\boldsymbol{\theta}} \\ &= \boldsymbol{w} + \beta\gamma \cdot \nabla_{\widehat{\boldsymbol{\theta}}} \mathcal{L}_q \cdot \nabla^2_{\boldsymbol{\theta},\boldsymbol{w}} \mathcal{L}_{train}\end{aligned} \tag{8}$$

We use $|\boldsymbol{\theta}|, |\boldsymbol{w}|$ to denote the dimensions of $\boldsymbol{\theta}, \boldsymbol{w}$ respectively. The second-order derivation $\nabla^2_{\boldsymbol{\theta},\boldsymbol{w}} \mathcal{L}_{train}$ is a $|\boldsymbol{\theta}| \times |\boldsymbol{w}|$ matrix which is too huge to calculate and store. Apart from that, calculating the matrix-vector product is also expensive. Precisely calculating the results is unrealistic. Fortunately, we can adopt the approximation technique used in DARTS (Liu et al., 2018) to solve this problem. This technique uses the finite difference approximation:

$$\begin{aligned}&\nabla_{\widehat{\boldsymbol{\theta}}} \mathcal{L}_q \cdot \nabla^2_{\boldsymbol{\theta},\boldsymbol{w}} \mathcal{L}_{train} \approx \\ &\frac{\nabla_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{\theta}^+, \boldsymbol{w}) - \nabla_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{\theta}^-, \boldsymbol{w})}{2\epsilon}\end{aligned} \tag{9}$$

$$\begin{aligned}\boldsymbol{\theta}^+ &= \boldsymbol{\theta} + \epsilon \nabla_{\widehat{\boldsymbol{\theta}}} \mathcal{L}_q \\ \boldsymbol{\theta}^- &= \boldsymbol{\theta} - \epsilon \nabla_{\widehat{\boldsymbol{\theta}}} \mathcal{L}_q\end{aligned} \tag{10}$$

where ϵ is a small scalar. We follow Liu et al. (2018) to set $\epsilon = 0.01/\|\nabla_{\widehat{\boldsymbol{\theta}}} \mathcal{L}_q\|_2$ which is accurate enough for approximation. Let $\alpha = \beta\gamma$, we can adjust the learning rate of $\boldsymbol{w}$ by tuning α.

Calculating this approximated gradient needs only another two forward passes for $\boldsymbol{\theta}^+$ and $\boldsymbol{\theta}^-$, which greatly accelerates the training procedure. More details about the training process are described in Algorithm 1.

2.4 Optimization Details

2.4.1 Dataset Split Strategy

The data split of query set $\mathcal{D}_q$ is critical. As mentioned in Section 2.1, we randomly sample $\mathcal{D}_{it}$ and $\mathcal{D}_q$ from in-domain training set $\mathcal{D}_{in}$. If we have enough in-domain data, $\mathcal{D}_{it}$ and $\mathcal{D}_q$ should be disjoint. However, our in-domain training set is not so large, and splitting it will make it even smaller. As a result, we use $\mathcal{D}_{in} = \mathcal{D}_q = \mathcal{D}_{it}$ instead of sampling. The ablation studies about this issue are shown in Section 3.5.

2.4.2 Scaling the Weights

In this work, an extreme value of w_i may make the training unstable. It is important to scale it to an appropriate range. In practice, we use sigmoid function to normalize it into $[0, 1]$:

$$\mathcal{L}_{train} = \frac{1}{n_1 + m} \sum_{\substack{(x_i, y_i) \in \\ \mathcal{D}_{it} \cup \mathcal{D}_{out}}} \sigma(w_i) \ell(f_{\boldsymbol{\theta}}(x_i), y_i) \tag{11}$$

Algorithm 1 **WIND** (**W**eighting **IN**stances **D**ifferentially)

Input: In-domain training set $\mathcal{D}_{it}$, out-of-domain set $\mathcal{D}_{out}$, query set $\mathcal{D}_q$, model parameter $\boldsymbol{\theta}$
Initialize the weights $\boldsymbol{w}$ (see Section 2.4.3)
for $i = 1$ **to** $epochs$ **do**
  for $j = 1$ **to** $steps\_per\_epoch$ **do**
    Get a mini-batch $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{D}_{it} \cup \mathcal{D}_{out}$
    Get a mini-batch $(\boldsymbol{x}_v, \boldsymbol{y}_v) \in \mathcal{D}_q$
    for $k = 1$ **to** $inner\_steps$ **do**
      Obtain $\widehat{\boldsymbol{\theta}}$ by Eq. 4
      Obtain $\widehat{\boldsymbol{w}}$ by Eq. 6 and Eq. 9
      Update $\boldsymbol{w}$ in the storage by $\widehat{\boldsymbol{w}}$
    end for
    Update $\boldsymbol{\theta}$ by Eq. 7
  end for
end for

2.4.3 Initialization of Instance Weight

How to initialize $\boldsymbol{w}$ is an important issue. In this paper, we assume that all the training samples from in-domain training set $\mathcal{D}_{it}$ are beneficial and should be highly weighted. For samples in D_{it}, we fix their weights to a very large number at the beginning of training, which is close to 1 after calculating by the sigmoid function. For samples in D_{out}, we initialize their weights all by zeros. During training, we do not optimize the weights of the in-domain training samples and only update the weights of the out-of-domain training samples.

Moreover, when to initialize $\boldsymbol{w}$ is another important issue. We propose two different kinds of initialization strategy. One is to initialize $\boldsymbol{w}$ at the beginning of each iteration. Another alternative is to initialize $\boldsymbol{w}$ at the beginning of the training, and update $\boldsymbol{w}$ in the storage every iteration. In practice, we choose the latter. Although the former is more easy to implement, it cannot make use of $\widehat{\boldsymbol{w}}$ from previous iterations.

3 Experiments

To evaluate the effectiveness of our proposed method introduced in Section 2 and demonstrate its model-agnostic property, we apply our method to

Dataset	Sentences
B-train ($\mathcal{D}_{in}$)	500
D,E,K ($\mathcal{D}_{out}$)	18,000
B-dev ($\mathcal{D}_{dev}$)	1,500
B-test ($\mathcal{D}_{test}$)	4,000

Table 1: Statistics of sentiment classification setting with *books* domain as the in-domain.

Dataset	Sentences
TED training ($\mathcal{D}_{in}$)	202,356
WMT14 subset ($\mathcal{D}_{out}$)	500,000
TED tst2012 ($\mathcal{D}_{dev}$)	1,700
TED tst2013 ($\mathcal{D}_{test}$)	993
TED tst2014 ($\mathcal{D}_{test}$)	1,305

Table 2: Statistics of machine translation setting.

Dataset	Docs	Relations
bc-train ($\mathcal{D}_{in}$)	10	222
nw & bn ($\mathcal{D}_{out}$)	332	4,695
bc-dev ($\mathcal{D}_{dev}$)	10	347
bc-test ($\mathcal{D}_{test}$)	40	1,036

Table 3: Statistics of relation extraction setting.

three different dataset settings of three tasks: Sentiment Classification, Machine Translation (MT) and Relation Extraction, respectively.

3.1 Datasets

For sentiment classification task, we conduct the experiments on the widely-used Amazon Review Dataset (Blitzer et al., 2007). This dataset contains four domain: *books* (B), *dvd* (D), *electronics* (E) and *kitchen* (K). Each domain contains the reviews of a specific category of products. We use the data processed by He et al. (2018) and collect 6000 labeled samples for each domain. We split the data of each domain into training ($\mathcal{D}_{in}$), development ($\mathcal{D}_{dev}$) and test ($\mathcal{D}_{test}$) set. In each domain adaptation setting, we choose the training data of one domain as the in-domain data ($\mathcal{D}_{in}$) and all data of other three domains as the out-of-domain data ($\mathcal{D}_{out}$). Table 1 shows an example with *books* domain as the in-domain.

For machine translation task, similar to the settings of Luong and Manning (2015); Wang et al. (2017); Zeng et al. (2019), we use the IWSLT 2016 English (EN) to German (DE) corpus (Cettolo et al., 2016) as the in-domain data. This corpus contains about 202K sentences from TED talks. For out-of-domain data, we randomly sample a subset of 500K sentences from the WMT 2014 English-German corpus. Table 2 show the statistics of the datasets.

For relation extraction task, we evaluate our method on the ACE 2005 dataset. This dataset is suitable for evaluating domain adaptation because it contains six different domains. It has been adopted by many previous works (Nguyen and Grishman, 2014; Gormley et al., 2015; Fu et al., 2017) for cross-domain relation extraction. In this work, we take *broadcast news* (bn) and *newswire* (nw) domain as out-of-domain, and split *broadcast conversation* (bc) domain into train/dev/test sets with the ratio of $1:1:4$. Table 3 shows the detailed statistics.

3.2 Implementation Details

For sentiment classification task, we use the pre-trained BERT-base-uncased (Devlin et al., 2018) model provided by HuggingFace (Wolf et al., 2019) as our feature extractor. Our sentiment classifier is a one-hidden-layer MLP with ReLU as the activation function. For the optimization of model parameters $\boldsymbol{\theta}$, we use the AdamW (Loshchilov and Hutter, 2018) as the optimizer with a learning rate of $2e-5$, a warmup of 0.1 (of the total steps) and a linearly decayed learning rate scheduler. The computational cost is about 8-12 GPU hours on Tesla V100.

For machine translation, we choose a vanilla Transformer (Vaswani et al., 2017) as our backbone. We implement some baseline methods and our method via fairseq toolkit (Ott et al., 2019). We use MOSES[2] scripts to tokenize the English and German sentences, and then we apply Byte Pair Encoding (BPE) (Sennrich et al., 2015) algorithm to split the words into subwords. We limit the maximum length of the sentences to 250 subwords. We choose to share the embeddings of English and German with the vocabulary size of 32,000. We use Adam (Kingma and Ba, 2014) as the optimizer and a decayed learning rate of $7e-4$.

For relation extraction, we only focus on relation classification when the entity pairs are given for simplicity. We use the RBERT (Wu and He, 2019) model as our backbone. The configurations of optimizer and learning rate are the same as those in our sentiment classification experiments.

3.3 Baselines

We implemented the following baseline methods for comparison with our methods. It's worth noting that we don't choose some baselines (Jiang and Zhai, 2007; Wang et al., 2017) of instance weight-

[2] https://github.com/moses-smt/mosesdecoder

Method	B	D	E	K	Avg.
In	87.50	86.75	89.38	89.50	88.28
Out	90.17	90.17	92.58	92.95	91.47
In+Out	91.20	90.78	92.58	93.12	91.92
Ensemble (Wang et al., 2017)	90.55	90.00	92.83	92.83	91.55
IW-Fit (Wang et al., 2019)	90.40	90.00	92.85	92.62	91.47
DANN (Ganin et al., 2016)	91.15	90.88	**93.35**	93.35	92.18
WIND (ours)	**91.65**	**91.08**	93.10	**94.00**	**92.46**

Table 4: Experiment results (Accuracy) of domain adaptation for sentiment classification.

Method	tst2012	tst2013	tst2014
In	-	29.47	25.18
Out	21.45	22.50	19.63
In+Out	29.60	32.50	28.72
DM (Britz et al., 2017)	-	31.57	27.60
IDDA (Zeng et al., 2019)	-	32.93	28.88
WIND (ours)	**30.77**	**33.58**	**29.26**

Table 5: Experiment results (BLEU) of domain adaptation for machine translation.

Method	test set
In	68.05
Out	88.22
In+Out	89.58
DANN (Fu et al., 2017)	89.38
WIND (ours)	**90.54**

Table 6: Experiment results (Accuracy) of domain adaptation for relation extraction.

ing because they are quite early work and it's unfair to compare with them.

For sentiment classification:

- **In** A pre-trained BERT only fine-tuned on the in-domain training set.
- **Out** A pre-trained BERT only fine-tuned on the out-of-domain training set.
- **In+Out** A pre-trained BERT fine-tuned on both in-domain and out-of-domain data.
- **Ensemble** It ensembles the **in** model and the **out** model by adding their predictions. Note that this method is used as a baseline in Wang et al. (2017). Although Wang et al. (2017) conducted the experiments on machine translation, we can still adopt this method on sentiment classification task.
- **IW-Fit** It uses the weighting strategy proposed by Wang et al. (2019) for domain transfer.
- **DANN** It introduces the domain classifier and adversarial training as proposed by Ganin et al. (2016).

For machine translation, the meanings of **In**, **Out** and **In+Out** is the same as those in the sentiment classification setting. There are some other baselines for machine translation setting:

- **DM** This indicates the Discriminative Mixing method proposed by Britz et al. (2017), which adds a domain classifier to the encodings of source sentences similar to DANN (Ganin et al., 2016).
- **IDDA** This indicates Iterative Dual Domain Adaptation methods proposed by Zeng et al. (2019), which iteratively performs bidirectional translation knowledge transfer using knowledge distillation between in-domain and out-of-domain. Note that this method focuses on the performance of both domains but in this paper we only focus on in-domain performance.

For relation extraction, besides the **In**, **Out** and **In+Out** approaches, we also choose Fu et al. (2017) as our baseline. This method simply introduces DANN (Ganin et al., 2016) to cross-domain relation extraction. Note that it is implemented by convolutional neural network, we reimplement a RBERT (Wu and He, 2019) version of it.

3.4 Experiment Results

Table 4 shows the overall performance of our methods in the domain adaptation setting on the sentiment classification task. Our method achieves an absolute improvement of 0.45, 0.40, 0.52 and 0.88 points on four settings respectively in comparison to the **In+Out** baseline. Moreover, our method outperforms all the domain adaptation methods on the settings with B, D, K as the in-domain data except the E domain. Although our method does not beat

all baselines on all settings, it achieves the best average performance across the four settings. On average, we achieve an improvement of 0.28 point over DANN and 0.54 point over **In+Out**.

Table 5 shows the performance for the machine translation task. We use BLEU (Papineni et al., 2002) scores to measure the performance. Our method beats all baselines on all test sets (tst2013,tst2014) and the development set (tst2012). On these three datasets, we observe an improvement of 1.17, 1.08 and 0.54 BLEU points compared to **In+Out**. On tst2013 and tst2014 test sets, we also achieve an improvement of 0.65 and 0.38 BLEU points compared to IDDA (Zeng et al., 2019) method. Table 6 further shows our method's effectiveness on the relation extraction task.

Furthermore, from the results in Tables 4, 5 and 6, we can make the following observations:

(1) In comparison to the method in (Wang et al., 2019) which uses manually designed weighting metrics, our differentiable weighting approach outperforms it in the sentiment classification task. This result demonstrates that designing the metrics manually may not be the best solution for all the tasks. Designing them requires many prior human expert knowledge which is hard to generalize well across tasks. By contrast, our method can learn instance weights with the help of meta-learning based algorithm to improve the models' in-domain generalization capability.

(2) Domain adversarial based method is a strong baseline which is surpassed only by our method in the sentiment classification task and the relation extraction task. However, it performs not so well for the machine translation task. The potential reason may be that Britz et al. (2017) introduces the domain classifier after the encoder to learn domain-invariant features of sentences from source language, but both domains share the same decoder which cannot discriminate the features encoded by the encoder. In other words, this type of method may only pay attention to the encoder and ignore the domain transfer of decoder. In contrast, our method overcomes this problem by considering weighting the loss of the whole model and thus achieves better performance.

(3) For all three tasks, adding out-of-domain corpus to the training set will improve the overall performance. We believe that adding the data of some general domains can help the model better learn domain-invariant syntactic and semantic knowledge, so it can improve the performance on the in-domain data. This is consistent with the conclusions reached by transfer learning. Interestingly, this result is contradictory to the observation of Wang et al. (2017), whose experiment results show that adding out-of-domain to in-domain data degraded machine translation performance. We suspect that there is a problem with their training strategy. The hyperparameters required under each setting may be different. Some hyperparameters that are set incorrectly (e.g. the same as **In**) may make the result of **In+Out** even worse. Another reason may be that the RNN-based sequence-to-sequence NMT system they used tends to be more sensitive to the noise while the Transformer (Vaswani et al., 2017) model we used is more robust.

All in all, as we expected, our proposed method WIND achieves the best performance under the three task settings. This illustrates the advantages of using differentiable method for data weighting.

Method	**B**	**D**	**E**	**K**
WIND+split-init	91.10	90.55	92.68	93.60
WIND+split	91.50	91.12	93.42	94.00
WIND+rand	91.17	91.03	93.33	93.95
WIND	**92.12**	**91.28**	**93.65**	**94.15**

Table 7: The comparison between different variants of our method on the sentiment classification task. Note that the results in this table are evaluated under $|\mathcal{D}_{in}| = 1,000$ setting. "+rand" means randomly initializing $\boldsymbol{w}$. "+split" means splitting $\mathcal{D}_{in}$ into disjoint $\mathcal{D}_{it}$ and $\mathcal{D}_q$. "-init" means not assigning a large number to in-domain data weights.

3.5 Ablation Study

In this part, we study the effect of the strategies mentioned in Section 2.4.1 and Section 2.4.3. The experiment results shown in Table 7 demonstrate that:

(1) Assigning the weights of in-domain instances to a large number ($1e8$) and fixing them during training can improve the accuracy. The weights of this part do not actually need to be learned. Fixing them may reduce the interference to the learning process from the out-of-domain data.

(2) Zero initialization for weights of out-of-domain instances is better than the random initialization. The underlying reason for this may be that random initialization may easily make the model stuck into a local minima.

(3) No splitting for $\mathcal{D}_{in}$ can improve the performance as well. Intuitively, this improvement comes

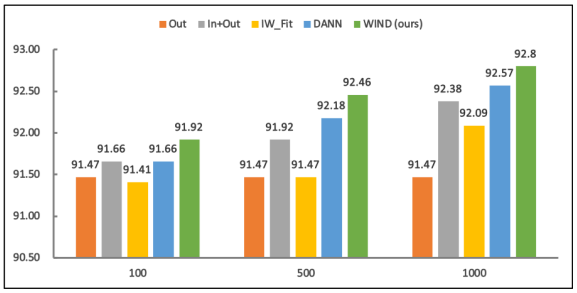

Figure 1: The effect of in-domain dataset size on the sentiment classification task.

from the increased size of in-domain training set, which enables us to make more use of the scarce in-domain training samples.

3.6 Effect of In-Domain Dataset Size

In this part, we aim to study the impact of in-domain dataset size n. Besides $n = 500$ setting, we sample another two different $\mathcal{D}_{in}$ with $n = 100$ and $n = 1000$. The rest of in-domain data is used as development set. We evaluate all three settings on the same $\mathcal{D}_{test}$ mentioned in Table 1.

Figure 1 shows the average accuracy over four domain settings when using different domain adaptation methods. We found that DANN (Ganin et al., 2016) may not perform so well when in-domain data are scarce. But our method can still achieve consistent improvements in this three dataset size settings.

4 Related Work

4.1 Domain Adaptation

Domain Adaptation is a fundamental problem in machine learning and NLP. We aim to train a well-performing model on a source domain which can be generalized to a target domain.

The basic idea for domain adaptation is to learn domain-invariant representations which generalize across the domains. To achieve this, the most prevailing method Domain Adversarial Neural Network (DANN) (Ganin et al., 2016; Qu et al., 2019; Xue et al., 2020) introduces a domain classifier and uses adversarial training to make the features unable to discriminate between source and target domains. This method has been applied to many NLP tasks. However, out-of-domain data is far more than in-domain data in our setting. DANN may cause some bias in this unbalanced dataset. Another type of methods (Fang and Xie, 2020; Li et al., 2020) propose to learn domain-general representations by contrastive learning (Chen et al., 2020a; He et al., 2019; Chen et al., 2020b). But they mainly focus on classification task and the methods are not model-agnostic frameworks.

4.2 Cross-Domain Sentiment Classification

Sentiment classification task aims to automatically classify the sentiment polarity of the given texts. Cross-domain sentiment classification aims to generalize the sentiment classifier from source domain to target domain.

Besides the domain adaptation methods introduced in Section 4.1, there are some methods which are specific for cross-domain sentiment classification. An important line of works follow the Structural Correspondence Learning (SCL) (Blitzer et al., 2006), and they design an auxiliary task called pivot prediction to transfer domain-invariant knowledge (Pan et al., 2010; Yu and Jiang, 2016; Ziser and Reichart, 2016, 2018, 2019). But the pivot words need human knowledge to select, which may be not so accurate. Recently, the pre-trained language models such as BERT (Devlin et al., 2018) have achieved state-of-the-art on many NLP tasks. DAAT (Du et al., 2020) performs a novel post-training procedure on BERT and uses adversarial training to transfer domain knowledge. But this method only works for classification task while our method is model-agnostic and does not need two-stage post-training and fine-tuning.

4.3 Meta-Learning

The goal of meta-learning is to train a model that can adapt to a new task quickly given a few new samples. In other words, meta-learning can learn the initial value of the model that is close to the optimums of many different tasks. MAML (Finn et al., 2017) is a classical method for meta-learning. Each entry of the meta-training set of MAML is a subset contains training data (support set) and test data (query set). MAML calculates the loss on the query set based on the parameters after one-step optimization on support set, and uses the gradient of this loss to update the model parameters. MAML has also been adopt for natural language understanding task before (Dou et al., 2019). Despite our domain adaptation setting is quite different from that in MAML, we can still utilize the idea of their work to help domain generation.

5 Conclusion

In this paper, we propose WIND, a differentiable instance weighting method for model-agnostic domain adaptation, which is inspired by the ideas of meta-learning to learn the weights on the in-domain query set. Experiment results on three typical NLP tasks show the efficacy of our framework.

It remains an open question how to efficiently transfer the domain knowledge. In the future, we plan to evaluate our method on more different tasks.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Xiaojun Wan is the corresponding author. Thank the reviewers for their constructive suggestions.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362.

James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128.

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126.

Mauro Cettolo, Niehues Jan, Stüker Sebastian, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2016. The iwslt 2016 evaluation campaign. In *International Workshop on Spoken Language Translation*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020a. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020b. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zi-Yi Dou, Antonios Anastasopoulos, and Graham Neubig. 2020. Dynamic data selection and weighting for iterative back-translation. *arXiv preprint arXiv:2004.03672*.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *arXiv preprint arXiv:1908.10423*.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, and Jianxin Liao. 2020. Adversarial and domain-aware bert for cross-domain sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4019–4028.

Hongchao Fang and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–429.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. 2006. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19:513–520.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. 2009. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. *arXiv preprint arXiv:1809.00530*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Tian Li, Xiang Chen, Shanghang Zhang, Zhen Dong, and Kurt Keutzer. 2020. Cross-domain sentiment classification with contrastive learning and mutual information maximization. *arXiv preprint arXiv:2010.16088*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 68–74.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Xiaoye Qu, Zhikang Zou, Yu Cheng, Yang Yang, and Pan Zhou. 2019. Adversarial category alignment network for cross-domain sentiment classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2496–2508.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488.

Zhi Wang, Wei Bi, Yan Wang, and Xiaojiang Liu. 2019. Better fine-tuning via instance weighting for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7241–7248.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364.

Qianming Xue, Wei Zhang, and Hongyuan Zha. 2020. Improving domain-adapted sentiment classification by deep adversarial mutual learning. In *Proceedings*

of the AAAI Conference on Artificial Intelligence, volume 34, pages 9362–9369.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jiali Zeng, Yang Liu, Jinsong Su, Yubin Ge, Yaojie Lu, Yongjing Yin, and Jiebo Luo. 2019. Iterative dual domain adaptation for neural machine translation. *arXiv preprint arXiv:1912.07239*.

Shiqi Zhang and Deyi Xiong. 2018. Sentence weighting for neural machine translation domain adaptation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3181–3190.

Yftah Ziser and Roi Reichart. 2016. Neural structural correspondence learning for domain adaptation. *arXiv preprint arXiv:1610.01588*.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Yftah Ziser and Roi Reichart. 2019. Task refinement learning for improved accuracy and stability of unsupervised domain adaptation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5895–5906.