Semi-supervised Domain Adaptation for Dependency Parsing with Dynamic Matching Network

Ying Li, Shuaike Li, Min Zhang
Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, China
yingli_hlt@foxmail.com, skli20@stu.suda.edu.cn
minzhang@suda.edu.cn

Abstract

Supervised parsing models have achieved impressive results on in-domain texts. However, their performances drop drastically on out-of-domain texts due to the data distribution shift. The shared-private model has shown its promising advantages for alleviating this problem via feature separation, whereas prior works pay more attention to enhancing shared features but neglect the in-depth relevance of specific ones. To address this issue, we for the first time apply a dynamic matching network on the shared-private model for semi-supervised cross-domain dependency parsing. Meanwhile, considering the scarcity of target-domain labeled data, we leverage unlabeled data from two aspects, i.e., designing a new training strategy to improve the capability of the dynamic matching network and fine-tuning BERT to obtain domain-related contextualized representations. Experiments on benchmark datasets show that our proposed model consistently outperforms various baselines, leading to new state-of-the-art results on all domains. Detailed analysis on different matching strategies demonstrates that it is essential to learn suitable matching weights to emphasize useful features and ignore useless or even harmful ones. Besides, our proposed model can be directly extended to multi-source domain adaptation and achieves best performances among various baselines, further verifying the effectiveness and robustness.

1 Introduction

Dependency parsing aims to capture syntactic and semantic information over input words via a dependency tree. As depicted in Figure 1, given an input sentence $\mathbf{s} = w_0 w_1 \dots w_n$, a dependency tree is defined as $\mathbf{d} = \{(h, m, l), 0 \leq h \leq n, 1 \leq m \leq n, l \in \mathcal{L}\}$, where (h, m, l) is a dependency from the head word w_h to the modifier word w_m with the relation label $l \in \mathcal{L}$. Recently, supervised neural models have achieved significant improvements in dependency parsing (Chen and Manning, 2014; Andor et al., 2016; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Li et al., 2019a). Particularly, Dozat and Manning (2017) propose a BiAffine parser and achieve good results on various languages.

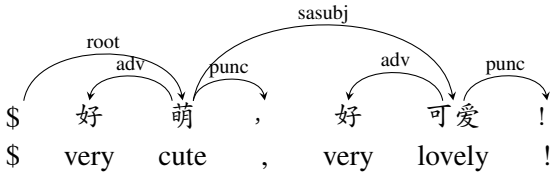
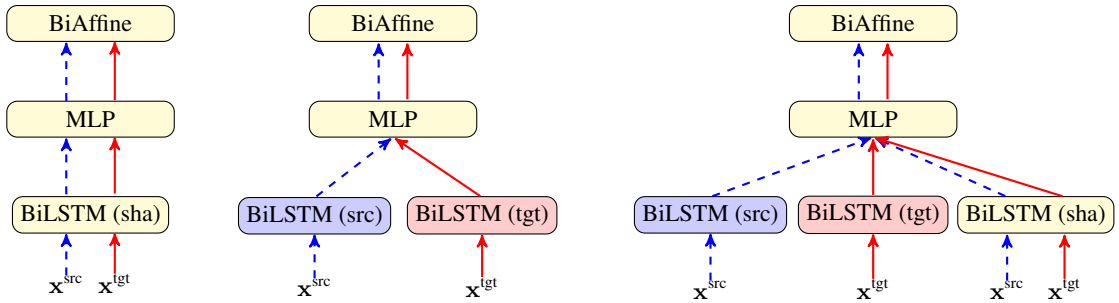


Figure 1: An example of a dependency tree which is from the target-domain product comment (PC) data.

In order to obtain better performance, supervised parsing models rely on sufficient in-domain training data. However, the parsing accuracy degrades significantly when the training data is from out-of-domain that has a large gap between the in-domain data.The main reason can be attributed to different feature distributions between source and target domains. Thus modeling the relevance of these distributions becomes the key challenge for cross-domain dependency parsing.

In the past few years, semi-supervised dependency parsing has attracted more attention with the surge of labeled web data that are user-generated non-canonical texts (Yu et al., 2013; Peng et al., 2019; Li et al., 2019b; Dakota et al., 2021). As shown in Figure 2, these approaches for modeling the similarity and discrepancy among different domains can be classified into three categories. The fully-shared model treats source and target domains equally and shares all model parameters, which may extract domain-invariant features but fail to capture domain-specific ones. In contrast, the fully-private model exploits completely independent encoders for each domain, which can better capture domain-specific features but ignore domain-invariant ones. To combine the advantages

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1035 - 1045
May 22-27, 2022 ©2022 Association for Computational Linguistics



(a) Fully-shared Model (b) Fully-private Model (c) Shared-private Model

Figure 2: Three basic frameworks for semi-supervised dependency parsing where the red solid lines represent the target domain information stream and the blue dashed lines are the source domain information stream.

of fully-shared and fully-private models, the shared-private model naturally separates domain-invariant and domain-specific features via shared and private encoders (Daumé III, 2007; Kim et al., 2016). However, this model still has two issues, i.e., neglecting the in-depth relevance of specific ones and failing to utilize unlabeled data effectively.

For the first issue, we investigate feature transfer approaches that encourage the target feature space to learn useful knowledge from source domain (Zagoruyko and Komodakis, 2017; Jang et al., 2019; Wright and Augenstein, 2020; Li et al., 2020a). Particularly, Jang et al. (2019) successfully use meta-learning to learn transfer weights between heterogeneous architectures and tasks. Motivated by this work, we propose a dynamic matching network based on the shared-private model for semi-supervised dependency parsing. Concretely, our model automatically generates matching weights to emphasize useful information and filter useless or even harmful features, thus further improving the power of target feature space.

For the second issue, considering that manually annotating samples for a new domain is time-consuming and expensive, we endeavour to effectively utilize target-domain unlabeled data. We design a new training strategy to use unlabeled data to enhance the power of matching network, thus modeling more effective specific features for the target domain. Meanwhile, we fine-tune BERT model with language model loss to obtain more reliable domain-related contextualized representations.

Experiments on benchmark datasets show that our proposed model outperforms the top submitted system in the NLPCC-2019 shared task (Li et al., 2019c), leading to new state-of-the-art results on all domains. In addition, detailed analysis on different matching settings reveals insights on the effect of intermediate source features. The extension on multi-source domain adaptation further verifies the effectiveness and robustness of our model. The code is released at `https://github.com/suda-yingli/ACL2022-match` to facilitate future research.

2 BiAffine Parser

In this work, we use the simple yet effective BiAffine parser (Dozat and Manning, 2017) as our basic model, which consists of four components, i.e., *Input Layer*, *BiLSTM Encoder*, *MLPs (multi-layer perceptron)*, and *BiAffines*.

Inputs. Each input word w_i is mapped into a dense vector $\mathbf{x}_i$. The vector is the concatenation of pre-trained word embedding $\mathbf{emb}_i^{\texttt{word}}$ and its Chinese character representation $\mathbf{rep}_i^{\texttt{char}}$,

$$\mathbf{x}_i = \mathbf{emb}_i^{\texttt{word}} \oplus \mathbf{rep}_i^{\texttt{char}} \quad (1)$$

where $\mathbf{rep}_i^{\texttt{char}}$ is generated by using one-layer BiLSTM to encode the characters of word w_i (Lample et al., 2016). In addition, we also use BERT representations to enhance our baseline where $\mathbf{emb}_i^{\texttt{word}}$ is substituted by $\mathbf{rep}_i^{\texttt{BERT}}$ simply.

BiLSTM. A three-layer BiLSTM is applied to sequentially encode the input vectors $\mathbf{x}_0\mathbf{x}_1 \ldots \mathbf{x}_n$ in two independent directions (forward and backward), and generates context-aware word representations $\mathbf{h}_0^{'}\mathbf{h}_1^{'} \ldots \mathbf{h}_n^{'}$ via combining the outputs of both directions.

MLPs. Two separate MLPs are used to obtain syntax-related lower-dimensional vectors.

$$\mathbf{r}_i^{\text{H}}, \mathbf{r}_i^{\text{D}} = \text{MLP}^{\text{H}}(\mathbf{h}_i^{'}), \text{MLP}^{\text{D}}(\mathbf{h}_i^{'}) \quad (2)$$

where $\text{MLP}^{\text{H}}(*)$ and $\text{MLP}^{\text{D}}(*)$ have a single hidden layer with the ReLU activation function. $\mathbf{r}_i^{\text{H}}$

and $\mathbf{r}_i^{\mathrm{D}}$ are vector representations of w_i as a head or a dependent word.

BiAffines. The score of a dependency $i \leftarrow j$ is obtained via BiAffine attention,

$$score(i \leftarrow j) = \mathbf{r}_j^{\mathrm{H}} \mathbf{U}^1 \mathbf{r}_i^{\mathrm{D}} + \mathbf{r}_j^{\mathrm{H}} \mathbf{U}^2 \quad (3)$$

where $\mathbf{U}^1$ and $\mathbf{U}^2$ are parameters. After obtaining the scores, the parser finds the highest-scoring tree with the dynamic programming algorithm known as maximum spanning tree (McDonald et al., 2005). Then, the classification of dependency labels is treated as a separate task, and the arc-factorization score is computed as follows:

$$score(i \overset{l}{\leftarrow} j) = \mathbf{r}_j^{\mathrm{H}} \mathbf{U}^3 \mathbf{r}_i^{\mathrm{D}} + (\mathbf{r}_j^{\mathrm{H}} \oplus \mathbf{r}_i^{\mathrm{D}}) \mathbf{U}^4 + b \quad (4)$$

where $\mathbf{U}^3$, $\mathbf{U}^4$, and b are parameters, and l is the relation label.

Parsing loss. During training, the parser computes two independent cross-entropy losses for each position, i.e., maximizing the probability of its correct head and the correct label between them.

$$\begin{aligned} \mathcal{L}_{\mathrm{par}}(i \overset{l}{\leftarrow} j) = &- \log \frac{e^{score(i \leftarrow j)}}{\sum_{0 \le k \le n, k \neq i} e^{score(i \leftarrow k)}} \\ &- \log \frac{e^{score(i \overset{l}{\leftarrow} j)}}{\sum_{l' \in \mathcal{L}} e^{score(i \overset{l'}{\leftarrow} j)}} \end{aligned} \quad (5)$$

where w_j is the gold-standard head of w_i, and l is the corresponding gold relation label.

3 Our Approach

Semi-supervised dependency parsing aims at learning a parser that generalizes well to the target domain. Although supervised parser has achieved good results on in-domain data, the parsing performance drops dramatically when the training data is mainly from the out-of-domain. The shared-private model has been proven effective for alleviating this problem. However, the model ignores the in-depth relevance of specific ones and fails to directly use unlabeled data for model training. To address these problems, we for the first time apply a dynamic matching network on the shared-private model to learn appropriate matching weights automatically via mimicking well-trained source features. As shown in Figure 3, our model mainly contains two components, i.e., a shared-private schema for feature separation and a dynamic matching network

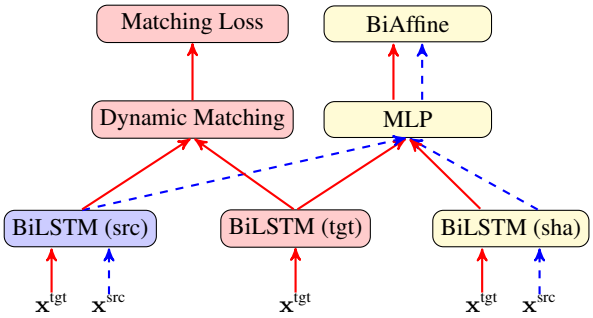


Figure 3: The framework of our proposed model.

for capturing the relevance of domain-specific features. In addition, we propose a new strategy for our model training to make full use of all labeled and unlabeled data.

3.1 Shared-private Schema

The framework of vanilla shared-private model is shown in Figure 2(c). First, each input word is encoded by the shared BiLSTM and its private BiLSTM to obtain domain-invariant and domain-specific representations. Then, the two representations are combined as the final context-aware representation $\mathbf{h}_i'$, which is fed into shared MLPs to obtain syntax-related information. Next, we obtain the scores of dependency arcs and labels via shared BiAffines. Finally, all model parameters are updated via minimizing the parsing loss.

Orthogonality constraints. Although the shared-private model has separated domain-invariant and domain-specific features via the shared and private encoders, the two type features may interfere with each other. To alleviate this problem, we apply orthogonality constraints to encourage the domain-specific features to be mutually exclusive with the shared ones. Following Bousmalis et al. (2016), we define the loss of orthogonality constraints as follows:

$$\mathcal{L}_{\mathrm{ort}} = \begin{cases} \sum_{i=0}^{n} \left\| (\mathbf{h}_i)^{\mathrm{T}} \mathbf{s}_i \right\|, & \text{if } w_i \in \{src\} \\ \sum_{i=0}^{n} \left\| (\mathbf{h}_i)^{\mathrm{T}} \mathbf{t}_i \right\|, & \text{if } w_i \in \{tgt\} \end{cases} \quad (6)$$

where $\mathbf{h}_i$ is the output of shared BiLSTM, $\mathbf{s}_i$ and $\mathbf{t}_i$ are the outputs of source-domain and target-domain private BiLSTMs.

3.2 Dynamic Matching Network

In practical application, some source features are more important than others while some are irrelevant or even harmful depending on the domain

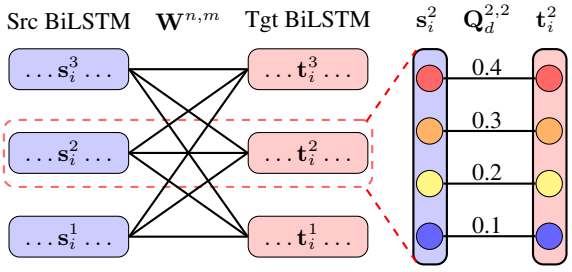


Figure 4: The framework of dynamic matching network.

differences. Hence, directly neglecting source features seems especially profligate. Motivated by Jang et al. (2019), we for the first time apply a dynamic matching network on the shared-private model to learn matching weights automatically. Thus the model is able to pay more attention on useful source features and ignore the useless ones. Considering the source features are well-trained with sufficient labeled data, mimicking these features may be helpful for enhancing the power of the target feature representational space. Hence, we minimize l_2 objection to transfer the knowledge from source features to the target ones:

$$||f_\theta(\mathbf{t}_i^m) - \mathbf{s}_i^n||_2^2 \tag{7}$$

where $f_\theta(*)$ is a linear transformation, $\mathbf{t}_i^m$ is the m^{th}-layer output of target-domain private BiLSTM, and $\mathbf{s}_i^n$ is the n^{th}-layer output of source-domain private BiLSTM. As shown in Figure 4, the key of dynamic matching network is learning layer matching weights $\mathbf{W}$ and element matching weights $\mathbf{Q}$.

Layer matching weights W. Intuitively, each intermediate feature of source domain has a completely different effect on the target domain. When we exploit the matching network to learn useful information from the source domain, a key problem is to decide the layer matching pair (n, m). Previous works select the matching pair based on prior knowledge of architectures or semantic similarities between tasks (Romero et al., 2015; Zagoruyko and Komodakis, 2017). To reduce the complexities of matching pair selection, we use a learnable layer matching weight $\mathbf{W}^{(n,m)} \geq 0$ for each pair (n, m) which can decide the amount of feature matching between the n^{th}-layer outputs of source-domain private BiLSTM and the m^{th}-layer outputs of target-domain private BiLSTM.

$$\mathbf{W}^{n,m} = g_\phi^{n,m}(\mathbf{s}_i^n) \tag{8}$$

where the Relu function is used to ensure non-negativeness of $\mathbf{W}$.

Element matching weights Q. After obtaining layer matching weight $\mathbf{W}^{n,m}$, we need to learn element matching weight $\mathbf{Q}_d^{n,m}$ to emphasize the useful intermediate elements according to their utility on the target domain. The matching loss of matching pair (n, m) is

$$\mathcal{L}_{\text{mat}}^{n,m} = \mathbf{W}^{n,m} \frac{1}{D} \sum_{d=1}^{D} \mathbf{Q}_d^{n,m} \left(f_\theta(\mathbf{t}_i^m) - \mathbf{s}_i^n\right)_d^2 \tag{9}$$

where D is the dimension of the BiLSTM output. $\mathbf{Q}_d^{n,m}$ is the non-negative weight of element d with $\sum_{d=1}^{D} \mathbf{Q}_d^{n,m} = 1$. Since the important elements to transfer can vary for each input word w_i, we set element transfer weights as follows:

$$\mathbf{Q}_d^{n,m} = \text{softmax}\left(r_\phi^{n,m}(\mathbf{s}_i^n)\right)_d \tag{10}$$

where $r_\phi^{n,m}(*)$ is the linear transformation.

After obtaining the element and layer matching weights, the combined matching loss is

$$\begin{aligned} \mathcal{L}_{\text{mat}} &= \frac{1}{K} \sum_{n,m} \mathcal{L}_{\text{mat}}^{n,m} \\ &= \frac{1}{KD} \sum_{n,m} \mathbf{W}^{n,m} \sum_{d=1}^{D} \mathbf{Q}_d^{n,m} \left(f_\theta(\mathbf{t}_i^m) - \mathbf{s}_i^n\right)_d^2 \end{aligned} \tag{11}$$

where $n, m \in \{1, 2, 3\}$ are the layer number of BiLSTM, and $K = 3 * 3 = 9$ is the number of matching pairs.

3.3 Joint Training Method

In order to make full use of all available training data, our model adopts a joint training strategy as shown in Algorithm 1. Here we split parameters in the model into two groups: 1) parsing parameters θ include all parameters of shared-private model and the linear function $f_\theta(*)$; 2) matching parameters ϕ include the parameters of all functions that generate matching weights. To balance the parsing and matching tasks, we give them different loss weights and update the dynamic matching parameters with a smaller learning rate.

In the joint training process, minibatches of source domain and target domain take turns to train (lines 3-5 and 6-11, respectively). When the minibatch is from the source domain, we update the parsing parameters θ with parsing and orthogonality losses. When the minibatch comes from target domain, if the data is annotated, the total model (θ & ϕ) is jointly trained with parsing, orthogonality and matching losses; otherwise only matching parameters ϕ are updated with the matching loss.

Algorithm 1 Joint Training Procedure

Input: source-domain labeled data S^l, target-domain labeled data T^l and target-domain unlabeled data T^u.

Hyper-parameters: loss weights α and β.

Repeat
  Take turns to sample a mini-batch x from S^l or T^l/T^u.
  if $x \in S^l$:
    Accumulate loss $\mathcal{L} = \mathcal{L}_{\text{par}} + \alpha\mathcal{L}_{\text{ort}}^{\text{src}}$
    Updating parameters θ via minimizing $\mathcal{L}$.
  if $x \in T^l$:
    Accumulate loss $\mathcal{L} = \mathcal{L}_{\text{par}} + \alpha\mathcal{L}_{\text{ort}}^{\text{tgt}} + \beta\mathcal{L}_{\text{mat}}$
    Updating all parameters via minimizing $\mathcal{L}$.
  if $x \in T^u$:
    Accumulate loss $\mathcal{L} = \alpha\mathcal{L}_{\text{ort}}^{\text{tgt}} + \beta\mathcal{L}_{\text{mat}}$
    Updating parameters ϕ via minimizing $\mathcal{L}$.
until convergence

	BC	PC	PB	ZX
train	16,339	6,885	5,129	1,645
dev	997	1,300	1,300	500
test	1,992	2,600	2,600	1,100
unlabeled	-	349,922	291,481	33,792

Table 1: Data statistics in sentence number

4 Experiments

4.1 Experimental Settings

Datasets. We use the Chinese multi-domain dependency parsing datasets released at the NLPCC-2019 shared task[1], containing four domains: one source domain which is a balanced corpus (BC) from news-wire, three target domains which are the product comments (PC) data from Taobao, the product blog (PB) data from Taobao headline, and a web fiction data named "ZhuXian" (ZX). Table 1 shows the detailed data statistics.

Evaluation. We use unlabeled attachment score (UAS) and labeled attachment score (LAS) to two-evaluate the dependency parsing accuracy (Hajic et al., 2009). Each model is trained for at most $1,000$ iterations, and the performance is evaluated on the dev data after each iteration for model selection. We stop the training if the peak performance does not increase in 100 consecutive iterations.

Hyper-parameters. We set the dimension of char embedding to 100. We train word2vec (Mikolov et al., 2013) on Chinese Gigaword Third Edition to obtain pre-trained word embeddings. To see the effect of contextualized representations, we use the released Chinese BERT-Base model[2] to yield BERT representations for each word. The averaged sum of the top four layer outputs is reduced into a dimension of 100 via an MLP. The learning rate for feature matching network and loss weights α and β are set as 10^{-4}, 0.01, and 0.01. For other hyper-parameters, we keep the default configuration in BiAffine parser (Dozat and Manning, 2017).

Baseline models. To verify the effectiveness of our proposed model, we select the following models as our strong baselines.

- **FulSha (Fully-shared).** The FulSha model, shown as Figure 2(a), directly trains the BiAffine parser with all labeled data from source and target domains.
- **FulPri (Fully-private).** The FulPri model, shown as Figure 2(b), exploits two independent BiLSTMs to separate source and target features absolutely.
- **ShaPri (Shared-private).** As shown in Figure 2(c), the ShaPri model can combine the advantages of fully-shared and fully-private models. It captures domain-invariant and domain-specific features simultaneously via utilizing two private and one shared BiLSTMs.
- **DoEmb (Domain Embedding).** The DoEmb model, proposed by Li et al. (2019b), has been proven effective for semi-supervised dependency parsing. The key idea is to use an extra domain embedding to indicate which domain the input sentence comes from.
- **ADE (Adversarial Domain Embedding).** Li et al. (2020b) successfully apply adversarial learning on the Doemb model and achieve good performances on semi-supervised dependency parsing. They leverage an extra domain embedding to capture domain-related information and adversarial network to extract more shared knowledge across different domains.

4.2 Analysis on Different Datasets

To gain more insights on the data distribution of different domains, we give a detailed analysis on our benchmark datasets from both lexical and syntactic aspects. On the one hand, we calculate word distributions for each domain. Figure 5 clearly shows that the same word appearing in different domains has completely different distributional probabilities. For example, the distributional probabilities

[1] http://hlt.suda.edu.cn/index.php/Nlpcc-2019-shared-task

[2] https://github.com/google-research/bert

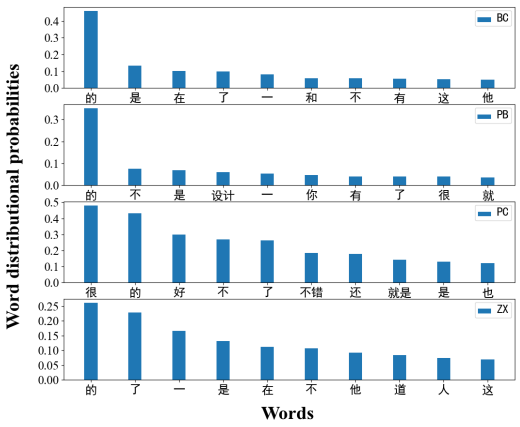


Figure 5: Word distributional probabilities of different domains.

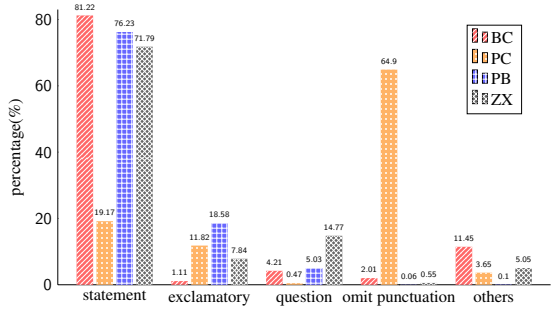


Figure 6: Sentence distributional probabilities of different domains.

of word "的" are 0.46 in BC domain, 0.43 in PC domain, 0.35 in PB domain, and 0.26 in ZX domain. Thus, it may inevitably lead to the shift of data distributions between different domains. On the other hand, we count sentence distributions for each domain based on the punctuation of input sentences. As shown in Figure 6, we find that the sentence distribution of source domain (BC) is similar to target domains (PB and ZX), but it is much different from the PC domain. The main reason is that the data of PC domain is non-canonical and contains a lot of ellipsis phenomena. Hence, not all source domain knowledge is equally important for the target domain, and it is necessary to automatically select the useful information from the source domain to enhance the performance on the target domain.

4.3 Utilization of Unlabeled Data

In this work, we leverage unlabeled data from two aspects: 1) learning more appropriate matching weights via enhancing the power of the dynamic matching network; 2) obtaining more reliable domain-related word representations by fine-tuning BERT.

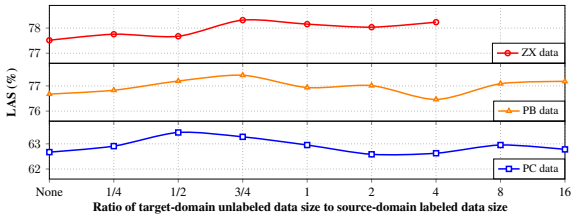


Figure 7: Influence of utilizing different amount of unlabeled data on our proposed model.

	PC		PB		ZX		AVG	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Models with BERT								
FulPri	71.58	64.15	83.53	79.26	84.25	80.00	79.79	74.47
FulSha	72.71	64.75	84.44	80.40	85.73	81.73	80.96	75.63
ShaPri	73.34	65.68	84.59	80.57	85.77	82.08	81.23	76.11
Our	**74.29**	**67.09**	**85.49**	**81.24**	**85.69**	**82.76**	**81.82**	**77.03**
Models with Fine-tuned BERT								
FulPri	73.55	65.88	83.89	79.62	83.17	79.04	80.20	74.85
FulSha	74.12	66.44	84.66	80.60	86.65	82.73	81.81	76.59
ShaPri	73.97	66.72	84.53	80.32	86.49	82.85	81.66	76.63
Our	**75.66**	**68.59**	**86.33**	**82.45**	**86.85**	**82.89**	**82.95**	**77.98**

Table 2: Results of different models on dev data regarding the utilization of BERT.

Since the amount of labeled data on target domain is much smaller than the source domain, we attempt to utilize target-domain unlabeled data to help the model to learn matching weights. Figure 7 illustrates the influence of unlabeled data sizes on dev data. In each curve, we fix the size of source-domain labeled data and incrementally add a random subset of target-domain unlabeled data. On the one hand, enlarging the size of unlabeled data leads to consistent improvements when the ratio is less than 3/4. This shows that the unlabeled data plays an important role in the matching weights learning. On the other hand, we can see that the parsing performance slightly degrades when the ratio increases larger than 1, indicating that the usefulness of the unlabeled data becomes limited when the size is too large.

Additionally, we leverage large-scale target-domain unlabeled data to fine-tune BERT model parameters, and detailed comparative experimental results are shown in Table 2. First, we observe that the model with fine-tuned BERT consistently outperforms the one with primary BERT representations, demonstrating that fine-tuning BERT is able to learn domain-related knowledge. Second, even the accuracy gap between different models reduces, our proposed model still achieves better per-

	PC		PB		ZX		AVG	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Results of previous works								
Yu (19)*	72.18	64.12	82.57	77.83	80.53	75.84	78.43	72.60
Peng (19)FE	73.16	64.33	83.05	78.57	82.09	77.08	79.43	73.33
Li (19)FB*	75.25	67.77	**85.53**	**81.51**	**86.14**	**81.65**	82.30	**76.98**
Li (20)FB	**75.93**	**68.34**	85.07	80.99	85.94	81.45	**82.31**	76.93
Compare with baseline models								
FulPri	70.02	61.43	79.60	74.74	76.56	71.05	75.39	69.07
FulSha	69.66	61.21	80.03	75.26	79.42	74.55	76.37	70.34
ShaPri	70.47	62.06	80.14	75.10	79.27	74.21	76.63	70.46
DoEmb	70.31	61.45	79.71	74.67	79.65	74.61	76.56	70.24
ADE	71.41	63.16	80.35	75.55	80.26	75.30	77.34	71.33
Our	**71.91**	**63.88**	**81.24**	**76.61**	**80.44**	**75.58**	**77.86**	**72.03**
Enhance models with BERT representations								
FulPri	72.75	65.08	83.96	79.64	83.08	78.48	79.93	74.40
FulSha	73.87	66.12	84.21	79.98	84.75	80.23	80.94	75.44
ShaPri	73.88	66.35	84.50	80.15	84.73	80.29	81.03	75.59
DoEmb	74.10	66.39	84.10	79.79	84.93	80.46	81.04	75.55
ADE	74.61	66.81	84.77	80.62	85.06	80.60	81.48	76.01
Our	75.24	67.36	85.38	81.21	85.87	81.54	82.16	76.71
OurFB	**76.73**	**69.38**	**86.06**	**81.63**	**86.56**	**82.49**	**83.12**	**77.83**

Table 3: Final results on test data where "FE" denotes "model with fine-tuned ELMo", "FB" denotes "model with fine-tuned BERT", and "*" denotes "model ensamble".

formance than the ShaPri model, which further verifies the effectiveness of feature matching network. Overall, unlabeled data is extremely helpful to enhance the feature representations that contribute for semi-supervised dependency parsing via fine-tuning BERT or enhancing the power of feature matching network.

4.4 Final Results

Table 3 shows the final results on test data and makes a comparison with previous works. First, we can see that the ShaPri model achieves better performance than FulPri and FulSha models, demonstrating that both domain-invariant and domain-specific features are helpful for semi-supervised dependency parsing. More specially, the FulSha model outperforms the FulPri one on PB and ZX domains but slightly declines on PC domain, possibly because the huge divergence between source and target domains leads to the interference for shared features learning. Although the ShaPri model already achieves better parsing accuracy, our model still outperforms it by 1.5% improvement in averaged LAS, indicating that the dynamic matching network is useful for enhancing the capability of target feature representational space via learning information from source domain. Second, the utilization of BERT boosts all model performances by a large margin. Fine-tuning BERT with unlabeled data can further enhance the model performance. Even the baseline models with BERT become much stronger, our proposed model still achieves the best

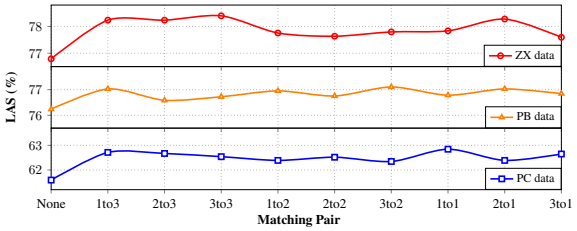


Figure 8: Accuracy curves regarding matching pairs.

	PC		PB		ZX		AVG	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
None	70.06	61.59	81.10	76.24	81.52	76.79	77.56	71.54
One	**71.07**	62.64	81.40	77.10	82.53	78.24	78.33	72.66
All	70.16	62.13	81.51	76.80	82.29	77.68	77.99	72.20
Learned	70.86	**62.93**	**81.96**	**77.76**	**82.81**	**78.32**	**78.54**	**73.00**

Table 4: Results of our proposed model on dev data with different matching settings.

performance, which further demonstrates the effectiveness of our proposed model. Finally, we present the remarkable results of previous works in the top block. Yu et al. (2019) combine self-training and model ensemble approaches to improve the model performance. Peng et al. (2019) re-implement the DoEmb model with fine-tuned ELMo using the codes released by Li et al. (2019b). The top system submitted by Li et al. (2019c) joints the advantages of tri-training, model ensemble, and BERT for the model training. (Li et al., 2020b) propose the ADE model and utilize fine-tuned BERT for semi-supervised dependency parsing, achieving competitive performances with the top system. Our proposed single model outperforms all these baseline models, leading to new state-of-the-art results on all domains.

4.5 Effect on Different Matching Settings

Because there still lacks related studies of feature transfer on semi-supervised dependency parsing, we for the first time design detailed comparative experiments to gain more insight on the impact of migrating the intermediate features from source to target domain. Here, "One-to-one" means only selecting a matching pair with the matching weight 1; "All-to-all" means that all matching pairs are used with matching weights 1; "Learned matching" means that all matching pairs are used with generated matching weights by our matching network.

Figure 8 shows results of different "One-to-one" models where "1to3" means learning information from the 1^{th}-layer outputs of source-domain private BiLSTM to the 3^{th}-layer outputs of target-domain private BiLSTM. First, we can see that almost all

	PC		PB		ZX		AVG	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
FulPri	68.44	59.96	79.35	74.60	74.43	70.22	74.07	68.26
FulSha	70.19	62.11	81.42	76.62	81.84	76.87	77.82	71.81
ShaPri	69.87	61.87	81.33	76.71	82.20	77.15	77.80	71.91
DoEmb	70.26	62.00	81.19	76.75	82.37	77.52	77.91	72.09
ADE	70.80	62.69	81.57	76.92	82.57	77.84	78.31	72.48
Our	**71.36**	**63.67**	**82.00**	**77.53**	**82.45**	**78.16**	**78.61**	**73.12**

Table 5: Results of multi-source domain adaptation on dev data.

"One-to-one" models outperform the "None" one, demonstrating that the model can learn some useful information from source domain to target domain via a simple feature matching process. Second, the model achieves a slight improvement when we transform features from the source domain to the higher layer outputs of target private BiLSTM. The reason may be that the higher layer outputs of BiLSTM contain much syntax-related information which has a higher domain relevance. Finally, we find that different domains have different trends in the curves, so it is difficult to select an explicit matching setting that adapts all domains.

Table 4 presents that the best "One-to-one" model achieves better performances than "All-to-all". We suspect the reason may be that "All-to-all" model treats all matching pairs equally, thus may lead to potential conflicts between different matching pairs. Additionally, "Learned matching" boosts the "All-to-all" performance by a large margin, indicating that our model is extremely useful for learning matching weights and alleviating the conflicts of feature transfer. Overall, the results can clearly demonstrate that modeling appropriate matching weights to emphasize useful information and filter out harmful knowledge is crucial to improve the capability of domain adaptation.

4.6 Applications on Multi-source Domain Adaptation

Table 5 presents the parsing accuracy on dev data where each model is trained with multi-source domain training data. For example, if the target domain is PC, its training data comes from BC, PB, and ZX domains. On the one hand, we observe that the same model trained with multi-source domains slightly outperforms it trained with only one source domain. The reason may be that although multi-source domains can provide more knowledge for the target domain, the data distribution shift leads to the negative transfer. Therefore, using all source domains simultaneously always requires more sophisticated hand-crafted configurations of the feature transfer. On the other hand, we can see that our proposed model achieves the best performance over various baselines. It demonstrates that the learned matching weights are helpful for constructing the relationships between target and multiply source domains, thus further boosting the parsing accuracy of the target domain.

5 Related Work

Domain adaptation generally falls into two categories: *semi-supervised* where large-scale labeled data for the source and small-scale labeled data for the target are available and *unsupervised* where only the labeled data for the source domain is given.

5.1 Unsupervised Domain Adaptation

Due to the lack of target-domain labeled data, previous works focus on unsupervised domain adaptation. One stream of work attempts to create pseudo training samples for the target domain via self-training, co-training, or tri-training processes (Yarowsky, 1995; Blum and Mitchell, 1998; Clark et al., 2003; Søgaard and Rishøj, 2010; Yu et al., 2015; Li et al., 2019c; Saito et al., 2020). As a coin has two sides, *self-training* has been proven effective on cross-domain constituency parsing (McClosky et al., 2006) and dependency parsing (Yu et al., 2015), but Charniak (1997) reports either minor improvements or significant damage for parsing by self-training. Clark et al. (2003) show the same findings on POS-tagging. Both Sarkar (2001) and Steedman et al. (2003) demonstrate that *co-training* is helpful for cross-domain dependency parsing. Li et al. (2019c) successfully use *tri-training* and fine-tuned BERT to improve the parsing accuracy. However, these approaches often require both caution and experience for selecting the appropriate pseudo samples. Another stream of work focuses on learning the feature representations from multiple source domain via mixture of experts. Kim et al. (2017) combine the predictions of domain experts via attention. Guo et al. (2018) propose a mixture of experts which uses a point to set metric. (Wright and Augenstein, 2020) extend the mixture of experts method on large pre-trained transformer models, leading to significant improvements. Motivated by these works, our work attempts to learn the relationship between different domain-specific representations.

5.2 Semi-supervised domain adaptation

In the past few years, semi-supervised domain adaptation for dependency parsing has achieved great improvements with the development of parsing communities (Chen et al., 2013; Yu et al., 2013; Li et al., 2019b; Peng et al., 2019). *Feature separation*, as a strand work of semi-supervised domain adaptation, is first proposed by Daumé III (2007) and achieves good results on sequence labeling tasks. Finkel and Manning (2009) extend this method by using a hierarchical Bayesian prior. Kim et al. (2016) apply it on neural-based model which uses a shared and multiple private BiLSTMs to separate domain-invariant and domain-specific features. *Adversarial learning* is a common method to encourage the shared encoder to extract more pure domain-invariant features via cheating the domain classifier (Ganin and Lempitsky, 2015; Bousmalis et al., 2016; Cao et al., 2018). Most relatively, Sato et al. (2017) apply adversarial learning on shared-private model but find slight improvements for semi-supervised dependency parsing. Li et al. (2020b) also exploit adversarial learning on the shared-private and domain embedding models with two strategies and achieves better performances than no-adversarial ones. Another strand work is *feature transformation*. Ando and Zhang (2005) design a variety of auxiliary problems to learn various aspects of the target problem from unlabeled data. Chen et al. (2013) propose the traditional feature transformation for dependency parsing which is similar as a way of doing feature smoothing. Jang et al. (2019) utilize the meta-learning to learn transfer weights of heterogeneous networks and tasks, leading to great improvements. Hu et al. (2021) propose a multi-view framework which combines multiple source models into an aggregated source view at language, sentence, or sub-structure levels. However, there still lacks related researches on the neural-based model for cross-domain dependency parsing.

6 Conclusion

This work proposes a feature matching shared-private model for semi-supervised dependency parsing. Meanwhile, we utilize unlabeled data to enhance the power of feature matching network and the BERT representations. Our proposed approach achieves consistent improvements among various baseline models, leading to new state-of-the-art results on all domains. The detailed analysis shows that compared with manual matching setting, the automatically learned matching weights by our designed dynamic matching network can improve the parsing accuracy. Furthermore, our proposed model can be directly extended to multi-source domain adaptation and achieves the best performance among various baselines, further demonstrating the effectiveness and robustness of our proposed method.

Acknowledgments

We thank our anonymous reviewers for their helpful comments. We are very grateful to Zhenghua Li for his careful guidance of our work. We also thank Chen Gong, Qingrong Xia, Yu Zhang, Houquan Zhou, Yahui Liu, and Tong Zhu for their help in paper writing and polishing. This work was supported by National Natural Science Foundation of China (Grant No. 62036004) and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL*, pages 1–9.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*, pages 2442–2452.

Avrim Blum and Tom M. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*, pages 92–100. ACM.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Proceedings of NIPS*, pages 343–351.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of EMNLP*, pages 182–192.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI*, pages 598–603.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Wenliang Chen, Min Zhang, and Yue Zhang. 2013. Semi-supervised feature transformation for dependency parsing. In *Proceedings of EMNLP*, pages 1303–1313. ACL.

Stephen Clark, James R. Curran, and Miles Osborne. 2003. Bootstrapping pos-taggers using unlabelled data. In *Proceedings of CoNLL*, pages 49–55.

Daniel Dakota, Zeeshan Ali Sayyed, and Sandra Kübler. 2021. Bidirectional domain adaptation using weighted multi-task learning. In *Proceedings of IWPT*, pages 93–105.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*, pages 256–263.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. abs/1611.01734.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of NAACL*, pages 602–610.

Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of ICML*, pages 1180–1189.

Jiang Guo, Darsh J. Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of EMNLP*, pages 4694–4703.

Jan Hajic, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Stepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL: Shared Task*, pages 1–18.

Zechuan Hu, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Multi-view cross-lingual structured prediction with minimum supervision. In *Proceedings of ACL/IJCNLP*, pages 2661–2674.

Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. 2019. Learning what and where to transfer. In *Proceedings of ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3030–3039.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Proceedings of ACL*, pages 643–653.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING*, pages 387–396.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*, pages 260–270.

Rumeng Li, Xun Wang, and Hong Yu. 2020a. Metamt, a meta learning method leveraging multiple domain data for low resource machine translation. In *Proceedings of AAAI*, pages 8245–8252.

Ying Li, Zhenghua Li, and Min Zhang. 2020b. Semi-supervised domain adaptation for dependency parsing via improved contextualized word representations. In *Proceedings of COLING*, pages 3806–3817.

Ying Li, Zhenghua Li, Min Zhang, Rui Wang, Sheng Li, and Luo Si. 2019a. Self-attentive biaffine dependency parsing. In *Proceedings of IJCAI*, pages 5067–5073.

Zhenghua Li, Xue Peng, Min Zhang, Rui Wang, and Luo Si. 2019b. Semi-supervised domain adaptation for dependency parsing. In *Proceedings of ACL*, pages 2386–2395.

Zuchao Li, Junru Zhou, Hai Zhao, and Rui Wang. 2019c. Cross-domain transfer learning for dependency parsing. In *Proceedings of NLPCC*, pages 835–844.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of NAACL*, pages 152–159.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*, pages 523–530.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Xue Peng, Zhenghua Li, Min Zhang, Rui Wang, Yue Zhang, and Luo Si. 2019. Overview of the nlpcc 2019 shared task:cross-domain dependency parsing. In *Proceedings of NLPCC*, pages 760–771.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *Proceedings of ICLR*.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. 2020. Universal domain adaptation through self supervision. In *Advances in NeurIPS*.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver*, pages 71–79.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of COLING*, pages 1065–1073.

Mark Steedman, Anoop Sarkar, Miles Osborne, Rebecca Hwa, Stephen Clark, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL*, pages 331–338.

Dustin Wright and Isabelle Augenstein. 2020. Transformer based multi-source domain adaptation. In *Proceedings of EMNLP*, pages 7963–7974.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*, pages 189–196.

Juntao Yu, Mohab Elkaref, and Bernd Bohnet. 2015. Domain adaptation for dependency parsing via self-training. In *Proceedings of IWPT*, pages 1–10.

Mo Yu, Tiejun Zhao, and Yalong Bai. 2013. Learning domain differences automatically for dependency parsing adaptation. In *Proceedings of IJCAI*, pages 1876–1882.

Nan Yu, Zonglin Liu, Ranran Zhen, Tao Liu, Meishan Zhang, and Guohong Fu. 2019. Domain information enhanced dependency parser. In *Proceedings of NLPCC*, pages 801–810.

Sergey Zagoruyko and Nikos Komodakis. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of ICLR*.