APGN: Adversarial and Parameter Generation Networks for Multi-Source Cross-Domain Dependency Parsing

Ying Li[1], **Meishan Zhang**[2], **Zhenghua Li**[1]*, **Min Zhang**[1],
Zhefeng Wang[3], **Baoxing Huai**[3], **Nicholas Jing Yuan**[3]
[1] School of Computer Science and Technology, Soochow University
[2] School of New Media and Communication, Tianjin University; [3] Huawei Cloud, China
yingli_hlt@foxmail.com; mason.zms@gmail.com
{zhli13,minzhang}@suda.edu.cn
{wangzhefeng, huaibaoxing, nicholas.yuan}@huawei.com

Abstract

Thanks to the strong representation learning capability of deep learning, especially pre-training techniques with language model loss, dependency parsing has achieved great performance boost in the in-domain scenario with abundant labeled training data for target domains. However, the parsing community has to face the more realistic setting where the parsing performance drops drastically when labeled data only exists for several fixed out-domains. In this work, we propose a novel model for multi-source cross-domain dependency parsing. The model consists of two components, i.e., a parameter generation network for distinguishing domain-specific features, and an adversarial network for learning domain-invariant representations. Experiments on a recently released dataset for multi-domain dependency parsing show that our model can consistently improve cross-domain parsing performance by about 2 points in averaged labeled attachment accuracy (LAS) over strong BERT-enhanced baselines. Detailed analysis is conducted to gain more insights on contributions of the two components.

1 Introduction

Dependency parsing aims to derive syntactic and semantic tree structures over input words (McDonald et al., 2013). Given an input sentence $\mathbf{s} = w_1 w_2 \dots w_n$, a dependency tree, as depicted in Figure 1, 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 child word w_m with the relation label $l \in \mathcal{L}$.

Recently, supervised neural dependency parsing models have achieved great success, leading to impressive performance (Chen and Manning, 2014; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Li et al., 2019a). Remarkably, the BiAffine parsing model can obtain a UAS of 96.67 and a LAS of 95.03 on standard Penn Treebank benchmark for the English language.

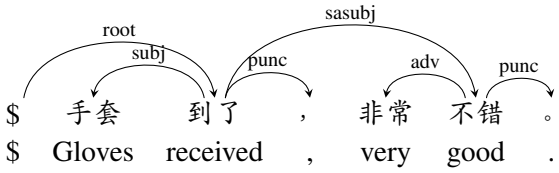

Figure 1: An example of dependency tree from product comments.

In order to obtain competitive performance, supervised dependency parsing models rely on a sufficient amount of training data, which is inevitably dominated to several fixed domains. When the test data is sourced from similar domains, good performance could be achieved. However, the performance could be decreased significantly when the test data is from a different domain which has a large gap between the training domains. Thus domain adaptation for dependency parsing has been concerned by a number of studies (Koo et al., 2008; Yu et al., 2013; Sato et al., 2017; Clark et al., 2018; Li et al., 2020b). These works mostly focus on single-source cross-domain dependency parsing, assuming the training data is from a single source domain (Yu et al., 2013; Sato et al., 2017). In fact, multi-source cross-domain dependency parsing is a more practical setting, considering that several dependency parsing corpora from different domains have been developed (Peng et al., 2019). Intuitively, an effective exploration of all these corpora can give better performance for the target domain compared with the single-source domain adaptation.

Separating domain-invariant and domain-specific features is one popular way for domain adaptation to distinguish the similarity and discrepancy of different domains (Daumé III, 2007; Kim et al., 2016; Sato et al., 2017). Domain-invariant features indicate the shared feature space across domains, which have been widely-adopted as

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1724–1733
November 7–11, 2021.

knowledge transferring. Domain-specific features imply the differences between domains, which could be helpful if the domain gaps could be accurately measured and effectively modeled. The learning of domain invariant and specific features are actually complementary because of mutual exclusivity, especially for single-source domain adaptation. Although single-source and multi-source settings are easily separated domain-invariant and domain-specific features via independent BiLSTMs, the in-depth relevance of domain-specific representations becomes more complicated with the increasing of source domains. Hence, how to construct the relationship between different domain-specific features after a simple feature separation becomes more challenge for multi-source dependency parsing.

In this work, we for the first time apply adversarial and parameter generation networks (APGN) to multi-source cross-domain dependency parsing for extracting domain-invariant and domain-specific features. Experiments on a benchmark dataset show that our proposed model can boost the parsing performance significantly, leading to averaged LAS improvements by 2 points over strong BERT-enhanced baselines. First, explorations on different unlabeled data sizes reveal that unlabeled data is an useful resource and proper utilization of unlabeled data further improves our model performance by a large margin. Then, we conduct in-depth analysis to gain crucial insights on the effect of adversarial and parameter generation networks, finding the two components are complementary and both have the capability of modeling short- or long-range dependencies. Finally, detailed comparative experiments on alternative domain representation strategies show that our designed distributed domain representation can accurately measure domain gaps and extract more reliable domain knowledge that benefits the dependency parsing task. We will release our code at https://github.com/suda-yingli/EMNLP2021-apgn for facilitating future researches.

2 Baseline Model

In this work, we adopt the state-of-the-art deep BiAffine parser (Dozat and Manning, 2017) as our baseline model. Figure 2 shows the framework of the parser, which mainly contains four components, i.e., *Input layer*, *Encoder layer*, *MLP layer*, and *BiAffine layer*.

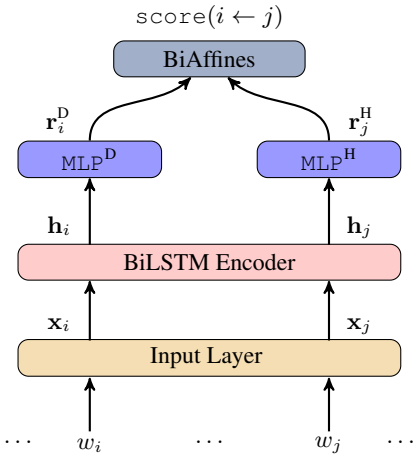

Figure 2: Framework of the BiAffine parser.

Input layer. The input layer maps each word w_i into a dense vector representation $\mathbf{x}_i$. First, we apply a BiLSTM to encode the constituent characters of each word w_i into its character representation $\mathbf{rep}_i^{\texttt{char}}$. Then, we concatenate $\mathbf{rep}_i^{\texttt{char}}$ with $\mathbf{emb}_i^{\texttt{word}}$ as the input vector $\mathbf{x}_i$.

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

where $\mathbf{emb}_i^{\texttt{word}}$ is the pre-trained word embedding, and $\oplus$ indicates vectorial concatenation. In addition, we also use BERT representation to enhance our model, denoted as $\mathbf{rep}_i^{\texttt{BERT}}$, where $\mathbf{emb}_i^{\texttt{word}}$ is substituted by $\mathbf{rep}_i^{\texttt{BERT}}$ simply.

Encoder layer. Following Dozat and Manning (2017), we employ a three-layer BiLSTM to sequentially encode the inputs $\mathbf{x}_0 \ldots \mathbf{x}_n$ and generate context-aware word representations $\mathbf{h}_0 \ldots \mathbf{h}_n$. We omit the detailed computation of the BiLSTM due to space limitation.

MLP layer. The MLP layer uses two independent MLPs to get lower-dimensional vectors of each position $0 \leq i \leq n$.

$$\begin{aligned} \mathbf{r}_i^{\mathrm{H}} &= \mathrm{MLP}^{\mathrm{H}}(\mathbf{h}_i) \\ \mathbf{r}_i^{\mathrm{D}} &= \mathrm{MLP}^{\mathrm{D}}(\mathbf{h}_i) \end{aligned} \tag{2}$$

where $\mathbf{r}_i^{\mathrm{H}}$ is the representation vector of w_i as a head word, and $\mathbf{r}_i^{\mathrm{D}}$ as a dependent.

BiAffine layer. The score of a dependency $i \leftarrow j$ is computed by a BiAffine attention as follows,

$$\texttt{score}(i \leftarrow j) = \begin{bmatrix} \mathbf{r}_i^{\mathrm{D}} \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W}^b \mathbf{r}_j^{\mathrm{H}} \tag{3}$$

where the weight matrix $\mathbf{W}^b$ determines the strength of a link from w_j to w_i.

Parsing loss. The parsing loss for each position i is computed as:

$$\begin{aligned}\mathcal{L}^{\mathrm{par}}(i \overset{l}{\leftarrow} j) = &- \log \frac{e^{score(i \leftarrow j)}}{\sum_{0 \le k \le n, k \neq i} e^{score(i \leftarrow k)}} \\ &- \log \frac{e^{score(i \overset{l}{\leftarrow} j)}}{\sum_{l' \in \mathcal{L}} e^{score(i \overset{l'}{\leftarrow} j)}}\end{aligned} \tag{4}$$

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

3 Proposed APGN Approach

The goal of multi-source cross-domain dependency parsing is to train a parser that generates well to the target domain with labeled training data from multiple source domains and unlabeled data from the target domain. The most straightforward approach is training a parser with the concatenation of all source-domain training data. This can extract common features across different domains but fail to capture domain-specific knowledge. To address this issue, we propose an APGN approach for modeling the discrepancy and commonality between different domains simultaneously. As shown in Figure 3, our APGN model mainly contains two components, i.e., a PGN for distinguishing domain-specific features, and an adversarial network for learning domain-invariant representations.

In this section, we first give a detailed illustration of the parameter generation network which takes distributed domain embedding as input to alleviate potential domain conflicts caused by the fixed one. Then, we introduce the adversarial network which encourages the BiLSTM to extract more pure shared information by fooling the domain classifier. Finally, we propose a new strategy for our model training to make full use of all labeled and unlabeled data.

3.1 PGN

Jia et al. (2019) first propose PGN to generate BiLSTM parameters based on fixed task and domain embeddings for NER domain adaptation, finding that the PGN can effectively extract domain differences. However, the vanilla PGN requires cross-domain language model task as a bridge to help fixed domain embeddings training. Considering the development of pre-training techniques with language model loss and computational complexity, we first remove the language model from the PGN component and use pre-trained BERT to enhance model performance in the final experiments. Intuitively, each input word has its unique domain distributions, initializing these words with the same fixed domain embedding may lead to potential domain conflicts. We then improve the PGN via replacing the fixed domain embedding with distributed one to more accurately integrate multi-domain information. As shown in the right part of Figure 3, our PGN takes distributed representations as inputs and dynamically generates the domain-related PGN-BiLSTM parameters.

PGN-BiLSTM encoder. To better capture domain-specific features, we exploit the PGN-BiLSTM instead of a standard BiLSTM encoder. For convenience, we directly formalize the vanilla BiLSTM encoder as follows:

$$\mathbf{h}_0 \dots \mathbf{h}_n = \text{BiLSTM}\left(\mathbf{x}_0 \dots \mathbf{x}_n, \mathbf{V}\right) \tag{5}$$

where $\mathbf{V} \in \mathcal{R}^{\mathrm{U}}$ can be regarded as a flattened vector which contains all the BiLSTM parameters. Different from a vanilla BiLSTM which use statically allocated parameters and update them during training, PGN-BiLSTM dynamically generates BiLSTM parameters in order to reflect domain differences as follows.

$$\begin{aligned}\mathbf{h}_0^{\mathrm{spe}} \dots \mathbf{h}_n^{\mathrm{spe}} &= \text{PGN-BiLSTM}\left(\mathbf{x}_0 \dots \mathbf{x}_n, \mathbf{E}\right) \\ &= \text{BiLSTM}\left(\mathbf{x}_0 \dots \mathbf{x}_n, \mathbf{V} = \mathbf{W} \otimes \mathbf{E}\right)\end{aligned} \tag{6}$$

where $\otimes$ denotes matrix multiplication; $\mathbf{W} \in \mathcal{R}^{\mathrm{U} \times \mathrm{D}}$ is a parameter matrix to be trained; $\mathbf{E} \in \mathcal{R}^{\mathrm{D}}$ is distributed domain-aware sentence representation vector and will be explained later.

Distributed domain-aware sentence representation. The distributed domain-aware sentence representation vector can be regarded as a sum of weighted domain embeddings, where higher weights are expected to be assigned to domains that are more similar to the input sentence.

First, we compute domain distribution probabilities of each word via simple domain classification.

$$\mathbf{z}_i = \text{softmax}\left(\text{MLP}\left(\mathbf{h}_i^{\mathrm{dom}}\right)\right) \tag{7}$$

where $\mathbf{h}_i^{\mathrm{dom}}$ is the representation vector of the i-th word generated by a separated standard BiLSTM.

Then, we compute a distributed domain-aware word representation vector for each word via aggregating domain embeddings according to the do-

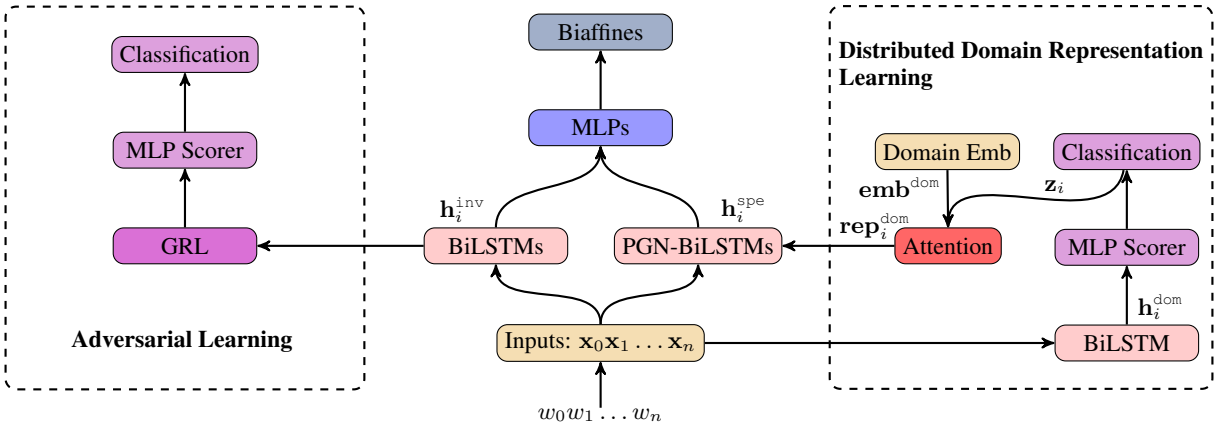

Figure 3: Framework of our proposed APGN model.

main distribution of the word.

$$\mathbf{rep}_i^{\text{dom}} = \sum_{j=1}^{m+1} \mathbf{z}_{i,j} \mathbf{emb}_j^{\text{dom}} \tag{8}$$

where $\mathbf{emb}_j^{\text{dom}}$ is the embedding vector of the j-th domain; $\mathbf{z}_{i,j}$ is the probability of the i-th word belonging to the j-th domain.

Finally, we utilize an average pooling to yield distributed domain-aware sentence representation vector, i.e., $\mathbf{E}$, which is used to generate BiLSTM parameters.

$$\mathbf{E} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{rep}_i^{\text{dom}} \tag{9}$$

Domain classification loss. The domain classification module, as shown in the right part of Figure 3, is trained via minimizing a standard cross-entropy loss.

$$\mathcal{L}^{\text{dom}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m+1} (\hat{\mathbf{z}}_{i,j}) \log ((\mathbf{z}_{i,j})) \tag{10}$$

where m is the number of source domains (plus a target domain); n is the word number of the input sentence; $\hat{\mathbf{z}}_i$ is the gold-standard domain distribution vector, where only one element is 1 corresponding to the domain index where the sentence comes from.

3.2 The Adversarial Network

The goal of adversarial learning is to encourage the shared BiLSTM to extract domain-invariant features that are not specific to a particular domain as much as possible (Ganin et al., 2017). During training, we expect the BiLSTM to make it difficult for the domain classifier to correctly distinguish domain categories. The architecture of adversarial network is shown in the left part of Figure 3. First, input words from different domains are encoded by the same standard BiLSTM. Before feeding the BiLSTM output $\mathbf{h}_i^{\text{inv}}$ to the domain classifier, $\mathbf{h}_i^{\text{inv}}$ goes through the gradient reversal layer (GRL). Following Ganin and Lempitsky (2015), the forward and backward propagations for the GRL are defined as follows:

$$\begin{aligned} G_\lambda(\mathbf{h}_i^{\text{inv}}) &= \mathbf{h}_i^{\text{inv}} \\ \frac{\mathrm{d}G_\lambda(\mathbf{h}_i^{\text{inv}})}{\mathrm{d}(\mathbf{h}_i^{\text{inv}})} &= -\lambda I \end{aligned} \tag{11}$$

where λ is a hyper-parameter. Over the GRL, the domain classifier is applied to identify the domain of input word. Finally, the adversarial network is trained via minimizing the cross-entropy loss $\mathcal{L}^{\text{adv}}$.

3.3 Joint Training

In this work, we design a joint training strategy to make full use of all available training datasets, shown as Algorithm 1. In the first k iterations, mini-batches of source-domain and target-domain take turns to train. If the mini-batch comes from the source-domain labeled data, we jointly train the model with the parsing, adversarial, and domain classification losses. Otherwise, the model is trained with the adversarial and domain classification losses. In the first stage, all data is used to select domain-invariant and domain-specific features via the adversarial and parameter generation networks. In the second stage, only source domain labeled data is available and the model is updated with the parsing loss until convergence after k iterations, which is helpful to deal with the overfitting problem of domain classifications.

Algorithm 1 Joint Training Procedure

Input: source-domain labeled data $\mathrm{S} = \{\mathrm{S}_i\}_{i=1}^{m}$, target-domain unlabeled data T.
Hyper-parameters: loss weights α and β, joint training iteration k.
Output: Target model.

Repeat
  if $iter < k$ **do**
    Take turns to sample a mini-batch x from S and T.
    if $x \in S$ do
      Accumulate loss $\mathcal{L} = \mathcal{L}^{\mathrm{par}} + \alpha\mathcal{L}^{\mathrm{adv}} + \beta\mathcal{L}^{\mathrm{dom}}$
    else
      Accumulate loss $\mathcal{L} = \alpha\mathcal{L}^{\mathrm{adv}} + \beta\mathcal{L}^{\mathrm{dom}}$
  else
    Sample a mini-batch $x \in$ S
    Accumulate loss $\mathcal{L} = \mathcal{L}^{\mathrm{par}}$
  Updating parameters via minimizing $\mathcal{L}$.
  $iter+ = 1$
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 Settings

Data. 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 illustration of the data statistics. In this work, we pick one target dataset as the target domain, and the rest are the source domains. For example, if the target domain is PC, source domains are BC, PB, and ZX.

Evaluation. We use unlabeled attachment score (UAS) and labeled attachment score (LAS) to 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.

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

- **Concatenation (CON).** We directly train BiAffine parser (Dozat and Manning, 2017) with all source-domain labeled data. The main drawback is that the parser shares all parameters across different domains and ignores domain differences, thus making it difficult to build the relationship between different domains.

- **Domain embedding (DE).** The vanilla DE method has been proven more effective than CON on semi-supervised dependency parsing (Li et al., 2019b). The key idea is to train BiAffine parser with an extra fixed domain embedding to indicate which domain the input sentence comes from. However, when the DE is directly applied to our task, fixed embeddings are trained inadequately due to the lack of target-domain labeled data.

- **Adversarial domain embedding (ADE).** Li et al. (2020b) propose to apply adversarial network on DE method, which separates domain-specific and domain-invariant features via domain-aware embeddings and adversarial learning. The ADE model can be regarded as the APGN removing the PGN component.

- **Parameter generation network (PGN).** Motivated by Jia et al. (2019), we exploit the PGN based on distributed domain representations to generate domain-related BiLSTM parameters as our strong baseline. The PGN can be regarded as our APGN model removing the adversarial network.

4.2 Hyper-parameter Choices

We mostly follow the hyper-parameter settings of Dozat and Manning (2017), such as learning rate, dropout ratios, and so on. The loss weights both α and β are set as 0.01. The domain embedding size is set as 8. The Chinese character embeddings are randomly initialized, and the dimension is 100. For pre-trained word embeddings, we train word2vec (Mikolov et al., 2013) embeddings on Chinese Gigaword Third Edition, consisting of about 1.2 million sentences. For BERT, we use the released Chinese BERT-Based model to obtain BERT representations for each word.[2] Following Li et al. (2019a), we utilize the averaged sum of the top-4 layer outputs as the final BERT representation $\mathbf{rep}_i^{\mathrm{BERT}}$.

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

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

Iter k	PC		PB		ZX		Avg.	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
10	49.62	37.89	73.96	68.26	74.19	66.90	65.92	57.68
20	**52.22**	**40.58**	**74.60**	**68.90**	**75.19**	**68.26**	**67.34**	**59.25**
30	50.57	38.46	73.99	68.24	74.51	67.50	66.36	58.07
40	49.77	37.61	74.25	68.09	74.67	67.21	66.23	57.64
50	50.10	38.09	74.01	67.97	74.39	67.20	66.17	57.75

Table 2: Results on the dev data regarding the joint training iteration k.

	PC		PB		ZX		Avg.	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Comparison with Baseline Models								
CON	47.30	35.63	72.81	67.24	71.00	62.91	63.70	55.26
DE	47.49	35.56	72.61	67.08	70.98	62.68	63.69	55.11
ADE	48.61	36.90	72.80	67.25	71.46	63.59	64.29	55.91
PGN	49.53	36.87	72.71	66.93	70.65	63.16	64.30	55.66
APGN	**51.48**	**39.12**	**73.86**	**68.10**	**72.43**	**64.80**	**65.92**	**57.34**
Comparison with BERT-Enhanced Baseline Models								
CON	60.62	49.52	81.59	77.07	80.60	74.53	74.27	67.04
DE	60.45	49.49	82.08	77.15	79.85	73.65	74.13	66.76
ADE	60.76	50.22	82.54	78.04	81.43	75.70	74.91	67.99
PGN	62.87	50.94	82.50	77.93	81.59	76.24	75.65	68.37
APGN	**63.17**	**52.11**	**82.92**	**78.21**	**82.71**	**77.03**	**76.27**	**69.12**

Table 3: Final results on test data.

Preliminary experiments show that our model is insensitive to most of the above parameters, while the setting of joint training iteration has a larger impact on the performance as shown in the following results.

Joint training iteration k. Table 2 shows the results with different joint training iteration k on the dev data. Increasing the iterations from 10 to 20 consistently improves the performance on all domains. The performance drops significantly when using iteration k above 20. These results indicate that more joint training iterations not only increase the complexity of the model, but also make the model prone to overfit the training data.

4.3 Final Results

Table 3 shows the final results and makes a comparison with multiple baselines on test data. First, we can see that our proposed APGN model achieves the best results on all domains, demonstrating that the APGN is extremely useful for multi-source cross-domain dependency parsing. Second, compared the results of ADE and PGN, we find that both adversarial and parameter generation networks have the capability of capturing useful information to improve the parsing accuracy. Finally, although the performance of different models is obviously improved by utilizing BERT representations, our model still achieves consistently higher accuracy than other baselines, further demonstrating the effectiveness of our proposed method.

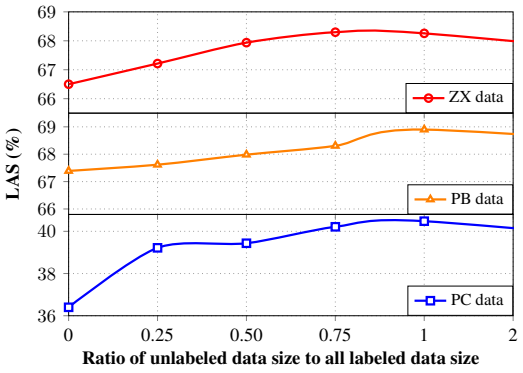

Figure 4: Influence of utilizing different amount of unlabeled data on APGN model. The x-axis is the ratio of target-domain unlabeled data size to source-domain labeled data.

4.4 Utilization of Unlabeled Data

Considering the lack of target-domain labeled data, we directly use unlabeled data for the model training. For unlabeled sentences, the model discards the parsing loss and updates the parameters with only adversarial and domain classifier losses. Figure 4 illustrates the influence of target-domain unlabeled data sizes on dev data. In each curve, we fix the size of labeled data and incrementally add a random subset of unlabeled data. Considering a large-scale unlabeled data may lead to the sample unbalance problem, we randomly sample unlabeled data with the ratios less than 1. On the one hand, we can see that using unlabeled data leads to consistently higher performance for all three domains, indicating that the unlabeled data is an important resource that contributes the target-domain dependency parsing. On the other hand, we find that the improvement of parsing accuracy is obviously steady when the ratio is set as 0.75, showing that the APGN model can achieve best performances with a suitable amount of unlabeled data.

4.5 Analysis

Ablation study. The results of ablation study on dev data are shown in Table 4. We can see that removing any component from the APGN causes obvious performance degradation. First, compared with the accuracy of "w/o two", "w/o PGN" can further improve parsing performance, showing the usefulness and importance of domain-invariant features generated by adversarial network. Second, it is clear that "w/o Adv" achieves better performance than "w/o PGN", indicating that the parameter generation network is crucial. The reason may be that

	PC		PB		ZX		Avg.	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
APGN	**52.22**	**40.58**	**74.60**	**68.90**	**75.19**	**68.26**	**67.34**	**59.25**
w/o Adv	51.16	38.72	73.96	67.88	74.00	67.17	66.37	57.92
w/o PGN	49.34	37.50	73.23	67.54	73.87	66.73	65.48	57.26
w/o Two	48.97	37.32	73.36	67.61	73.30	65.53	65.21	56.82

Table 4: Ablation study on reducing the component of the APGN model on dev data. "w/o Adv" and "w/o PGN" mean removing adversarial network or parameter generation network.

the parameter generation network enable correctly construct domain relations and extract practical domain-specific features, which is significant for dependency parsing. Finally and most importantly, we find that our proposed APGN model achieves consistently higher accuracy than "w/o PGN" and "w/o Adv", demonstrating that the two components are complementary.

Error analysis. Since ablation study only gives an overall performance trend, we conduct in-depth error analysis in order to gain more insights on the contributions of adversarial and parameter generation networks. We divide the gold-standard dependencies into seven subsets according to the absolute distance between the head word and the modifier word, and calculate the accuracy for each subset. The group whose dependency distance is 0 means the words which take the pseudo node "root" as their head words. Figure 5 compares the accuracy curves of ADE ("w/o PGN"), PGN ("w/o Adv"), and APGN models with regard to the dependency distance on the test data. First, we can see that the parsing accuracy becomes better on all models when the dependency distance is smaller. The reason may be that the contextualized information decays when the distance between two words is too far. Second, there seems slight difference between ADE and PGN performances on the same dependency distance, indicating that adversarial and parameter generation networks, as two typical feature extraction methods, both have the competitive capability of capturing short- and long-range dependencies. Finally, we find that the APGN model achieves better performances than ADE and PGN models, demonstrating that adversarial and parameter generation networks are complementary and can certainly benefit from each other.

4.6 Comparisons on Alternative Domain Representation Strategies

Most previous works use a fixed domain embedding to indicate the domain of each input word (Jia et al., 2019; Li et al., 2019b). However, the fixed representation may lead to potential domain conflicts when a word belongs to multiple domains. As shown in Figure 6, we can see that each word has its unique domain distribution and it is difficult to define all word with an explicit fixed representation. Hence, it is necessary to design a more accurate representation, named as distributed domain embedding, which can be regarded as weighted sum of the fixed domain embeddings and its distributional probabilities.

Detailed comparative experiments are conducted to verify the effectiveness of two domain representation strategies on various models, and results are shown in Table 5. First, we find that the APGN with fixed domain representations like Jia et al. (2019) achieves lower performance than other models. The main reason may be that without cross-domain language model as a bridge, it is difficult for the

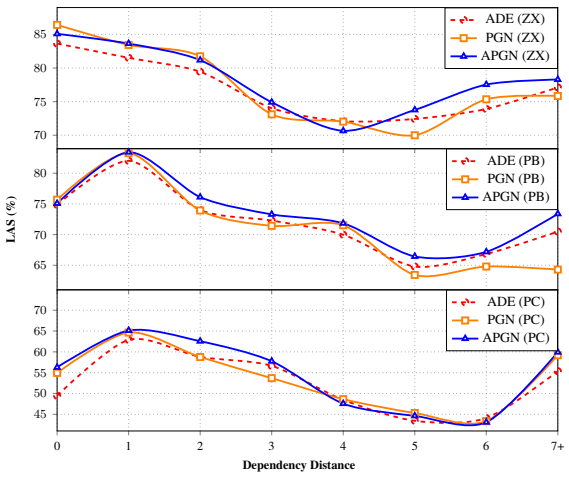

Figure 5: Accuracy curves regarding dependency distances.

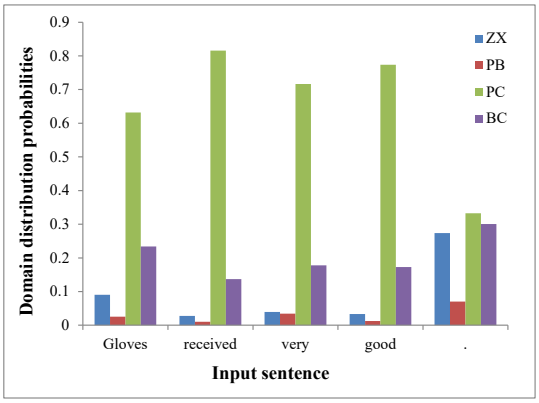

Figure 6: Domain distributional probabilities of different words.

	PC		PB		ZX		Avg.	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Models with the fixed domain representations								
DE	48.23	36.40	73.25	67.39	73.27	66.49	64.92	56.76
ADE	**49.16**	**36.68**	**73.49**	**67.89**	**73.91**	**67.01**	**65.52**	**57.19**
APGN	44.20	30.89	71.28	65.35	71.50	63.85	62.33	53.36
Models with the distributed domain representations								
DE	50.37	38.13	73.96	67.88	73.71	66.61	66.01	57.54
ADE	50.63	38.50	73.90	68.08	73.72	67.79	66.08	58.12
APGN	**52.22**	**40.58**	**74.60**	**68.90**	**75.19**	**68.26**	**67.34**	**59.25**

Table 5: Results of different models with fixed or distributed domain representations.

PGN to model the relationships of different domains. Second, the APGN with distributed domain representations achieves best performance among all models, revealing that the PGN with distributed domain embeddings can accurately measure the domain similarity and significantly improve our model performance. Finally, we can see that all models with distributed domain representation outperform them with the fixed one by a large margin, demonstrating that distributed domain representation is helpful to reduce potential domain conflicts and extracts more reliable domain knowledge that benefits the parsing task.

5 Related Work

Domain adaptation has been extensively studied in many research areas, including machine learning (Wang et al., 2017; Kim et al., 2017), computer vision (Ganin and Lempitsky, 2015; Rozantsev et al., 2019) and natural language processing (Kim et al., 2016; Sun et al., 2020). Here, we first simply review single-source domain adaptation researches, and then give more detailed illustration about the studies of multi-source domain adaptation.

Single-source domain adaptation. Single-source domain adaptation assumes training data comes from a source domain. Due to lacking target-domain labeled data, previous researches mainly investigate *unsupervised* domain adaptation, which attempt to create pseudo training samples by self-training (Charniak, 1997; Steedman et al., 2003; Reichart and Rappoport, 2007; Yu et al., 2015), co-training (Sarkar, 2001), or tri-training (Li et al., 2019c). However, selecting high confidence samples is a challenge.

Thanks to large-scale labeled web data released by parsing communities, recent existing works pay more attention to *semi-supervised* scenario. Yu et al. (2013) give detailed error analysis on cross-domain dependency parsing and solve the ambiguous features problem. Sato et al. (2017) propose to separate domain-specific and domain-invariant features via applying adversarial learning on shared-private model, but find that there is little gains and even damage the performance, specially when the scale of target-domain training data is small. Most recently, Li et al. (2019b) propose to leverage an extra domain embedding to indicate domain source and achieve better performance on semi-supervised domain adaptation. In this work, we adjust the domain embedding method as our strong baseline.

Multi-source domain adaptation. Multi-source domain adaptation assumes the training data comes from multiple source domains. Many approaches of multi-source domain adaptation focus on leveraging domain knowledge to extract domain-related features, thus boosting the performance of target domain (Daumé III, 2007; Guo et al., 2018; Li et al., 2020a; Wright and Augenstein, 2020). Zeng et al. (2018) design a domain classifier and an adversarial network to capture domain-specific and domain-invariant features, achieving good performances on machine translation. Guo et al. (2018) apply meta-training and adversarial learning to compute the point-to-set distance as the weights of multi-task learning network, leading to improvement on classification tasks.

As another interesting direction, Platanios et al. (2018) propose a parameter generation network to generate the parameters of the encoder and decoder by accepting the source and target language embeddings as input. Recently, a number of works attempt to use the parameter generation network to improve the cross-domain or cross-language performance (Cai et al., 2019; Stoica et al., 2020; Jin et al., 2020; Nekvinda and Dusek, 2020). Particularly, Jia et al. (2019) propose to generate BiLSTM parameters based on task and domain representation vectors, leading to very promising performances on cross-domain NER task.

Due to the limitation of annotation corpus and the essential difficulty of multi-source domain adaptation, there still lacks such studies on dependency parsing. Inspired by these prior works, we propose a novel approach to separate domain-invariant and domain-specific features by the utilization of adversarial and parameter generation networks.

6 Conclusion

This work for the first time apply the APGN approach to multi-source cross-domain dependency parsing, obtaining better performance than multiple

baselines, even when all models are enhanced with BERT representations. The ablation study reveals that both adversarial and parameter generation networks are equally important and complementary in capturing domain-related features, which motivates us to make a deep analysis to gain crucial insights on the effectiveness of the two components. Based on the in-depth error analysis, we find that in spite of local divergences, domain-invariant and domain-specific features generated by adversarial and parameter generation networks actually both have the power of modeling short- or long-range dependencies and can certainly benefit from each other. Furthermore, detailed comparative experiments demonstrate that the distributed domain representation is extremely useful to reduce domain conflicts and accurately measure the domain similarity, thus extracting more reliable domain-specific features to boost the parsing performance.

6.1 Acknowledgments

We thank our anonymous reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (Grant No. 61876116 and 62176173), and a Project Funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions.

References

Hengyi Cai, Hongshen Chen, Cheng Zhang, Yonghao Song, Xiaofang Zhao, and Dawei Yin. 2019. Adaptive parameterization for neural dialogue generation. In *Proceedings of EMNLP-IJCNLP*, pages 1793–1802.

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.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of EMNLP*, pages 1914–1925.

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.

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

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2017. Domain-adversarial training of neural networks. In *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition, pages 189–209.

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 the Thirteenth Conference on Computational Natural Language Learning: Shared Task, CoNLL 2009*, pages 1–18.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain NER using cross-domain language modeling. In *Proceedings of ACL*, pages 2464–2474.

Tian Jin, Zhun Liu, Shengjia Yan, Alexandre E. Eichenberger, and Louis-Philippe Morency. 2020. Language to network: Conditional parameter adaptation with natural language descriptions. In *Proceedings of ACL*, pages 6994–7007.

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.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603.

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.

Ryan T. McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B. Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL*, pages 92–97.

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.

Tomás Nekvinda and Ondrej Dusek. 2020. One model, many languages: Meta-learning for multilingual text-to-speech. In *Proceedings of the Annual Conference of the International Speech Communication Association*, pages 2972–2976.

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.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom M. Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of EMNLP*, pages 425–435.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of ACL*.

Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. 2019. Beyond sharing weights for deep domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(4):801–814.

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*, pages 71–79.

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.

George Stoica, Otilia Stretcu, Emmanouil Antonios Platanios, Tom M. Mitchell, and Barnabás Póczos. 2020. Contextual parameter generation for knowledge graph link prediction. In *Proceedings of AAAI*, pages 3000–3008.

Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Learning sparse sharing architectures for multiple tasks. In *Proceedings of AAAI*, pages 8936–8943.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of EMNLP*, pages 1482–1488.

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

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.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of EMNLP*, pages 447–457.