Learning Domain Invariant Word Representations for Parsing Domain Adaptation

Xiuming Qiao[1], Yue Zhang[2,3] and Tiejun Zhao[1]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[2] School of Engineering, Westlake University, China
[3] Institute of Advanced Technology, Westlake Institute for Advanced Study
hitxiaoqiao@gmail.com, zhangyue@westlake.edu.cn,
tjzhao@hit.edu.cn

Abstract. We show that strong domain adaptation results for dependency parsing can be achieved using a conceptually simple method that learns domain-invariant word representations. Lacking labeled resources, dependency parsing for low-resource domains has been a challenging task. Existing work considers adapting a model trained on a resource-rich domain to low-resource domains. A mainstream solution is to find a set of shared features across domains. For neural network models, word embeddings are a fundamental set of initial features. However, little work has been done investigating this simple aspect. We propose to learn domain-invariant word representations by fine-tuning pretrained word representations adversarially. Our parser achieves error reductions of 5.6% UAS, 7.9% LAS on PTB respectively, and 4.2% UAS, 3.2% LAS on Genia respectively, showing the effectiveness of domain invariant word representations for alleviating lexical bias between source and target data.

Keywords: Word representations, Wasserstein distance, Generative Adversarial Network, Domain adaptation, Dependency parsing

1 Introduction

Dependency parsing aims to analyze the syntactic relationships (i.e. $head \rightarrow dependent$) between words within a sentence, which plays an important role on many natural language processing tasks, such as machine translation [1], information extraction [2] and natural language inference [3]. Due to lack of labeled data, dependency parsing for resource-poor domain has been a challenging task. Many work has been focused on transferring models from the resource-rich domain to the resource-poor domain [4] [5].

There are two main factors which make it difficult to do domain adaptation: the difference of the distribution between domains, and different labeling schemes [6]. For instance, for English Penn treebank (PTB) which is a news treebank [7] , and Genia [8] which is a biology treebank, the labeling guidelines are same but the lexical coverage ratio of PTB on Genia is about 25% and even much lower on proper nouns. The heavy lexical bias between these two domains makes it difficult to transfer parsing models directly.

Many efforts have been devoted to alleviating the lexical bias by learning lexical representations. Koo et al. [9] derived word clusters from large unannotated corpus and

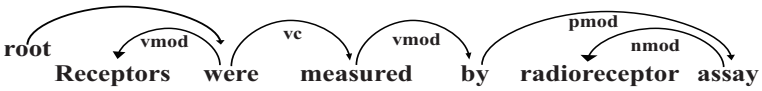


Fig. 1: An example of dependency tree.

added cluster-based features into a discriminator model. Recently, the neural network-based bias correction methods were proposed, through representing one word as a low dimensional real-valued vector [10] For adaptation between two domains, these word representations may not be sufficiently useful because they are trained on a large and diverse corpus but not targeted for only these two domains. We argue that the word representations tuned on specific source and target domains could transfer the source parsing model more effectively.

In this paper, we propose to do domain-adaptive fine-tuning for specific source-target domains pair based on existing pretrained word representations using Wasserstein Generative Adversarial Network (WGAN) [11]. In our method, the generator G of WGAN learns domain invariant word representations, and then represents target sentence representation as close to the source sentence representation as possible. Given the representation of a sentence, the discriminator D then predicts which domain the sentence comes from. Based on the decision of D, both of D and G tune themselves and become better, until D cannot distinguish which domain the sentence is from. As a result, the Wasserstein distance of source data distributions and target data distributions could be increasingly minimized in this way.

During our experiments, the loss of WGAN converges and the accuracy of the discriminator D stabilizes to 0.5. Finally, the data transferability of source domain is improved for the target domain. Experimental results show that our method achieves the performance of 91.85% UAS, and 90.94% LAS on a biology dataset Genia and achieve 95.98% UAS, and 95.01% LAS on PTB respectively.

We make two contributions in this paper:

- We derive domain invariant word representations by Wasserstein Generative Adversarial Networks.
- Using domain invariant word representations, we improve the performance of parsing domain adaptation.

2 Deep Biaffine Parsing Model

Given a sentence x, the task of dependency parsing is to produce its dependency tree y, assigning a head-dependent relation for each word in x, as shown in Fig.1. Our baseline model is the deep biaffine attention model [12].

Let w_t be the t-th token in the sentence. For each w_t, the word embedding $\mathbf{w}_t \in \mathbb{R}^{d_{word}}$ and a part-of-speech (POS) embedding $\mathbf{p}_t \in \mathbb{R}^{d_{pos}}$ are used as its representations, where d_{word} and d_{pos} are the dimensions of $\mathbf{w}_t$ and $\mathbf{p}_t$. The concatenation of $\mathbf{w}_t$ and $\mathbf{p}_t$ for the word w_t is $\mathbf{x}_t$, which is then passed through a multilayer bidirectional LSTM network and mapped to a hidden vector $\mathbf{r}_t$.

First, the model computes the score $\mathbf{s}_{i,j}^{(arc)}$ for each possible dependency arc for w_i (head) to w_j (dependent), producing four vector representations:

$$\begin{aligned}
\mathbf{r}_t &= BiLSTM(\mathbf{x}_t), \\
\mathbf{h}_i^{(arc-head)} &= MLP^{(arc-head)}(\mathbf{r}_i), \\
\mathbf{h}_j^{(arc-dep)} &= MLP^{(arc-dep)}(\mathbf{r}_j), \\
\mathbf{s}_{i,j}^{(arc)} &= \mathbf{h}_i^{T(arc-head)} U^{(arc)} \mathbf{h}_j^{(arc-dep)} \\
&\quad + \mathbf{h}_i^{T(arc-head)} \mathbf{u}^{(arc)},
\end{aligned} \tag{1}$$

where MLP is a multi layer perceptron, $U^{(arc)}$ is a weight matrix, and $\mathbf{u}^{(arc)}$ is used in the bias term. During decoding, the parsing tree that has the maximum score is found via maximum spanning tree (MST) algorithm.

Second, the model assigns a label for each arc in the dependency tree according to the score $\mathbf{s}_{i,j}^{label}$.

$$\begin{aligned}
\mathbf{h}_i^{(label-head)} &= MLP_{\mathbf{r}_i}^{(label-head)}, \\
\mathbf{h}_j^{(label-dep)} &= MLP_{\mathbf{r}_j}^{(label-dep)}, \\
\mathbf{h}_{i,j}^{(label)} &= \mathbf{h}_i^{(label-head)} \oplus \mathbf{h}_j^{(label-dep)}, \\
\mathbf{s}_{i,j}^{(label)} &= \mathbf{h}_i^{T(label-head)} \mathbf{U}^{(label)} \mathbf{h}_j^{(label-dep)} \\
&\quad + \mathbf{h}_{i,j}^{T(label)} W^{(label)} + \mathbf{u}^{(label)},
\end{aligned} \tag{2}$$

where $\mathbf{U}^{(label)}$ is a third-order tensor, $W^{(label)}$ is a weight matrix, and $\mathbf{u}^{(label)}$ is a bias vector.

Our research focuses on learning an domain invariant embedding matrix through tuning general word embeddings pre-trained on multiple domains. In section 3, we will introduce how we learn word representations for specific source and target domain, aiming to make their data much closer.

3 Learning Word Representations Using WGAN

The problem of supervised domain adaptation for dependency parsing is denoted as follows. We have labeled source data $D_s = \{(x_s, y_s)\}_{s=1}^{N^s}$ and labeled target domain data $D_t = \{x_t, y_t\}_{t=1}^{N^t}$, we assume that they share the same feature space but follow different distribution $\mathbb{P}_S$, $\mathbb{P}_T$ respectively. We additionally assume that they share the same annotation guideline. We adjust the feature distribution through learning domain invariant word representations to make the distance of feature distribution $\mathbb{P}_S$ and $\mathbb{P}_T$ as small as possible.

There are many methods to measure the distance between two probability distributions including the KL divergence, the JSD divergence, the Wasserstein distance and so on. We use the Wasserstein distance function [11], also called Earth-Mover (EM) distance, which is calculated as

$$W(\mathbb{P}_S, \mathbb{P}_T) = \inf_{\gamma \in \prod(\mathbb{P}_S, \mathbb{P}_T)} \mathbb{E}_{(x^s, x^t) \sim \gamma}[||x^s - x^t||]. \tag{3}$$

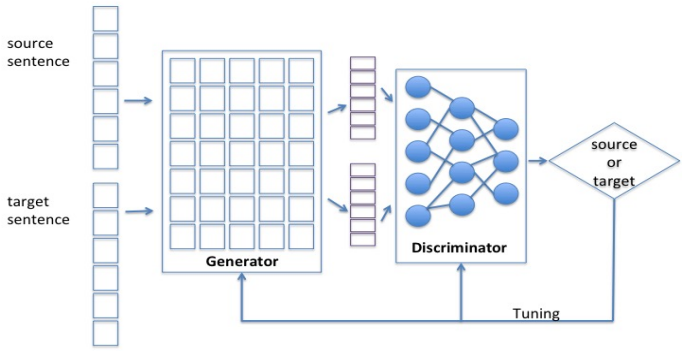


Fig. 2: The structure of our WGAN.

$\prod(\mathbb{P}_S, \mathbb{P}_T)$ is the collection of all possible joint distributions of $\mathbb{P}_S$ and $\mathbb{P}_T$. From each possible joint distribution γ, we can sample $(x^s, x^t) \in \gamma$ and compute $||x^s - x^t||$, which is the distance between the source sample x^s and target sample x^t. Then we can compute the expectation value of this distance under the joint distribution γ. Wasserstein distance can be seen as the minimum cost of the best plan move $\mathbb{P}_S$ to $\mathbb{P}_\mathbb{T}$.

The structure of our WGAN is shown in Fig. 2. It contains two parts: the generator G with parameters w and the discriminator D with parameters θ. G generates domain invariant word representations and changes sentences into vectors. The discriminator D is a sentence classifier. Given sentence vectors, D predicts whether the sentence is from source or target domain.

3.1 Generator

Our generator G consists of a word embedding layer. Each word in the vocabulary V is associated with a k-dimensional vector. Let x_i be the i-th word in the sentence x and $\mathbf{x_{1:n}}$ be the concatenation of word vectors from x_1 to x_n. Given a sentence x of length n (padded when its length is smaller than n), the generator G will output its representations

$$G(x) = \mathbf{x_1} \oplus \mathbf{x_2} \oplus ... \oplus \mathbf{x_n}, \tag{4}$$

where $\oplus$ is the concatenation operator. The parameter in G is the word representation matrix $W \in \mathbb{R}^{k \times |V|}$.

3.2 Discriminator

The discriminator D is a convolutional sentence classifier, similar to the work of [13]. Given $G(x)$, D predicts which domain the sentence x comes from. Our classifier contains one convolutional layer, one max-pooling layer, and one full-connected hidden layer with dropout and softmax.

A feature c_i is produced when a filter $\mathbf{w} \in \mathbb{R}^{\mathbf{h}\times\mathbf{k}}$ is applied to a window of h words,

$$c_i = f(\mathbf{w} \cdot \mathbf{x_{i:i+h-1}} + \mathbf{b}). \tag{5}$$

In Equation 5, $b \in \mathbb{R}$ is a bias item and f is a non-linear activation function. The filter is applied in all possible word windows in the sentence $\mathbf{x_{1:h}}, \mathbf{x_{2:h+1}}, ..., \mathbf{x_{n-h+1:n}}$ to produce a set of feature map,

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}]. \tag{6}$$

Then we use max-over-time pooling method over the feature map and take the max value $\widehat{c} = max(\mathbf{c})$ as the feature corresponding to this filter $\mathbf{w}$. Then features $\mathbf{z} = [\widehat{c_1}, ..., \widehat{c_m}]$ are passed to the fully-connected layer. For regularization, we compute the output with:

$$y = \mathbf{W_1} \cdot \mathbf{z} + \mathbf{b_1}. \tag{7}$$

To mitigate overfitting, we apply dropout in hidden layer with the dropout rate of 0.5. $\mathbf{W_1} \in \mathbb{R}^{m\times m}$ is the weight matrix and $\mathbf{b_1} \in \mathbb{R}^m$ is the bias. Finally, the softmax layer is used to predict the domain the sentence x comes from.

The network parameters in the discriminator are $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{W_1}, \mathbf{b_1}\}$.

3.3 Loss Function

For the input of the discriminator $G(x)$, we denote the logits output by hidden layer as $D(G(x))$. When computing the Wasserstein distance of the source and target distributions, we use the logits output by the hidden layer without softmax.

$$wd_{loss} = D(G(S)) - D(G(T)). \tag{8}$$

Given the source data, the output of the discriminator is $D(G(S))$, and $D(G(T))$ for target data. The loss function for the generator G is the wd_{loss}

$$G_{loss} = wd_{loss}. \tag{9}$$

We set the loss of classifier C as the cross-entropy of output from softmax layer of the discriminator. The loss of the discriminator is a combination of the loss of classifier C_{loss} and wd_{loss} with a hyper parameter $wd_{param} \in [0, 1]$

$$D_{loss} = C_{loss}(S \cup T) - wd_{param} * wd_{loss} \tag{10}$$

3.4 Training

The training process of our WGAN is shown in Algorithm 1. First, parameters of the generator and discriminator are initialized. Second, we randomly split training data into mini-batches with batch size m^s for the source data and m^t for the target data. For each batch, we first train the discriminator for d_s steps and then train the generator for g_s steps. Parameter optimization is performed by RMSprop (root mean square) optimizers with learning rate α_1 for G and α_2 for D. If the parameters do not converge, we repeat the second step for more epochs until the maximum epoch is reached.

Algorithm 1 Learning word representations via WGAN

Require: Training data: source data $\boldsymbol{S} = \{(x_i^s)\}_{i=1}^{N^s}$; target data $\boldsymbol{T} = \{x_i^t\}_{i=1}^{N^t}$; minibatch size m^s and m^t; discriminator training steps d_s; generator training steps g_s; learning rate for generator α_1; learning rate for discriminator α_2; the maximum epoch $epoch_{max}$;

$ep = 0$
while $ep \leqslant epoch_{max}$ **do**
  $ep = ep + 1$
  while θ has not converged **do**
    $batch_{num} = math.ceil(N_s/m_s, N_t/m_t)$
    for b =1 to $batch_{num}$ **do**
      Sample $\{x_i^s\}_{i=1}^{m^s}$, $\{x_i^t\}_{i=1}^{m^t}$ from $\boldsymbol{S}$, $\boldsymbol{T}$
      for $j = 1$ to d_s **do**
        $g_\theta \leftarrow \nabla_\theta[C_{loss} - wd_{param} * wd_{loss}]$
        $\theta \leftarrow \theta + \alpha_2 \cdot RMSprop(\theta, g_\theta)$
      end for
      for $k = 1$ to g_s **do**
        $g_w \leftarrow \nabla_w G_{loss}$
        $w \leftarrow w - \alpha_1 \cdot RMSprop(\theta, g_w)$
      end for
    end for
  end while
end while

4 Experiments

In this section, we describe our experiments on learning domain invariant word representations and applications on dependency parsing, and discuss experimental results from different views.

4.1 Experiments on Learning Domain Invariant Word Representation

The source data consists of training sentences of the PTB, which are from the news domain. The target sentence is the training set of Genia corpus [8], a semantically annotated corpus for biology text.

The dimension k of word embeddings is 100. The size of filters in the convolutional layer are 3, 4 and 5 and the number of filters per size is 128. wd_{param}, the weight for wd_{loss} in D_{loss}, is 0.5. Both the learning rates α_1 and α_2 are 0.001. Steps of optimizing the discriminator and generator d_s and g_s are 5 and 1, respectively. The maximum number of epochs $epoch_{max}$ is 100. Both of the source and target batch size are 512.

Stability and Convergence of WGAN We show the change of wd_{loss} along with training steps on Fig. 3. It is clear that wd_{loss} gets smaller and converges gradually. The smaller the wd_{loss} is, the source and target domain data are much more closer under domain invariant word representations.

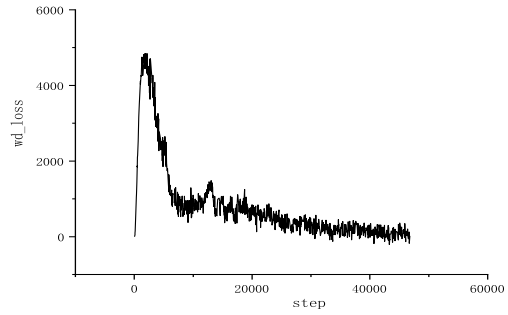


Fig. 3: Change of wd_{loss} against the number of training steps.

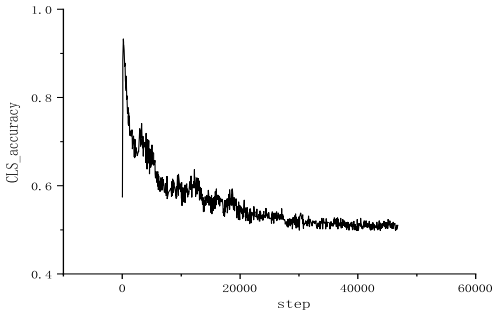


Fig. 4: Accuracy of the classifier in D against the number of training steps.

The accuracy of the classifier in D converges to roughly 0.5, as shown in Fig. 4, which shows that the discriminator can not distinguish the domain of one sentence any more [14].

Our learning of domain invariant word representations performs well in stability and model collapse. In addition, it takes two hours to finish our adversarial learning using one GPU of GTX 1080.

Evaluation by Word Similarity We compare word representations output by our WGAN (denoted as emb_wg) with GloVe word embeddings with 100 dimensions trained on Wikipedia 2014 and Gigaword 5 [15] (denoted as GloVe100) in obtaining word semantic similarities.

Given a target word, we can get obtain top-n similar words in semantics according to the Euclidean distance of embeddings. As shown in Table 1, when the target word is $cause$, the similar words given in emb_wg contain $illness$ from biology domain. The results of GloVe100 are all from general domains. For the case of $expression$ which is frequently used in biology domain, the output of GloVe100 only contains similar words

Table 1: Comparison of similar words induced by word representations.

Target Word	Our Embeddings	GloVe100
cause	causes, causing, caused	causes, causing, caused
	illness, risk	failure, prevent
inflammatory	**responses**, viral, drugs	autoimmune, insensitive, incendiary
	autoimmune, hiv, disease	bowel,inflammation
necrosis	tumor, apoptosis,cells	apoptosis, factor-alpha, avascular
	differentiation, inflammation	tumor, inflammation
expression	sense, means,different	sense, emotion, meaning
	function, genes	manner, discourse

from the general domain, mostly to show feelings or ideas, but our results also contains words that means structure of an molecule or genes.

Our embeddings may connect words from the source and target domain with domain invariant distributions. Then similar syntactic structures from source domain could be better transferred to target domain.

4.2 Dependency Parsing Experiments

We use the deep biaffine attention parser [12] to evaluate our domain invariant word representations in the domain adaptation task for dependency parsing. The parsing performance is measured using unlabeled attachment score (UAS) and labeled attachment score (LAS). Following previous work [12,16], we compute UAS and LAS excluding punctuation.

Adapting PTB to Genia The dependency parsing results on Genia are shown in Table 2. The training data is the combination of the PTB training set and the Genia training set. The development and test set are from Genia. We adopt the division of [17]. Our baseline is the biaffine attention dependency model, and the word embedding is the GloVe100. Different with the baseline, we use the concatenation of word representations output by our WGAN and GloVe100, denoted as emb_wg. In Table 2, Genia-train stands for the experiment whose training data are Genia training set and the word representation is GloVe100. And Fine-tune stands for the model is trained on PTB and then fine-tuned on Genia.

The performance of baseline parser is better than PTB-train, showing the effectiveness of adapting the source model to the target domain. The results of Fine-tune are much worse than baseline, suggesting that news training does not give a better starting model for bio data. We improve the UAS by 0.36 and LAS by 0.3 as shown in Table 2, compared to the baseline model, showing the effectiveness of our domain invariant word representations in parsing domain adaptation from the news domain to the biology domain.

Adapting Genia to PTB We apply our domain invariant word representation on the domain adaptation dependency parsing from the biology to the news domain. The training data is the combination of PTB training set and Genia training set. We also train the parser only using PTB training set and GloVe100, denoted as PTB-train in Table 3.

Table 2: Parsing results of adapting PTB to Genia.

System	word embedding	UAS	LAS
baseline	GloVe100	91.49	90.64
Genia-train	GloVe100	91.39	90.46
Fine-tune	GloVe100	88.46	86.76
Ours	emb_wg	**91.85**	**90.94**

Table 3: Parsing results of adapting Genia to PTB.

System	word embedding	UAS	LAS
Baseline	GloVe100	95.74	94.69
PTB-train	GloVe100	95.77	94.78
Ours	emb_wg	**95.98**	**95.01**
D&M [2017]	glove	95.74	94.08
Ma et al. [2018]	M-word2vec	95.87	94.19

Dozat and Manning [2017] (denoted as D&M [2017]) use the GloVe embedding. Ma et al. [2018] is based on stack-pointer networks and uses word embeddings output by a modified word2vec (denoted as M-word2vec in Table 3) for syntax [18].

The dependency parsing results of adapting Genia to PTB are shown in Table 3. Our model improves the performance by 0.24 UAS and 0.32 LAS respectively, compared to the baseline, showing the effectiveness of our domain invariant word representation for parsing domain adaptation from the biology domain to the news domain. Our results are also better than both D&M [2017] and Ma et al. [2018]. In paticular, our parser has strong performance on predicting the labeled dependency relations by 95.01 LAS. Compared to D&M[2017], we improve the performance by 0.93 LAS. As the analysis in their paper (page 7) shows one reason for their lower LAS may be the inefficiencies of the GloVe embeddings [12].

The results on PTB and Genia show that: 1) given domain invariant word representations, model of resource-rich domain can better adapt to the resource-poor domain; 2) data of resource-poor domain can also improve the parsing performance of resource-rich domain.

Results by Different Dependency Relations We analyze the error reduction in different dependency relations and select the top-10 relations according to their error counts. The results are shown in Table 4 and Table 5. For the result on Genia, we improved the UAS on 9 out of 10 dependency relations and 7 out of 10 for PTB data. Our domain invariant word representations well work on dependency types of nmod, dep, advmod, nsubj, det, and case.

Case Study As observed in the PTB training dataset, the word $genome$ is an OOV(out of the vocabulary) word. From our word representations emb_{wg} we find that the most similar word is $genes$, which occurs frequently in PTB. When the parser tries to judge if the is a dependent of $genome$, the dependent pair $genes \rightarrow the$ in PTB can be transferred to this scenario. The most similar word given by GloVe100 is $sequencing$, which dose not have dependent of the in PTB.

5 Related Work

Dependency parsing has been investigated as a fundamental syntactic task. Typical methods include graph-based [12] and transition-based [19,20]. While most work focuses on news domain training using PTB, we here consider the issue of domain transfer. Bias correction techniques aim to make up the data bias between the samples of

Table 4: Error reductions of two systems on Genia in different dependency relations.

Relation	Explanation	Count	Baseline	Ours	Error Reduction (%)
nmod	nominal modifier	4135	523	503	3.82
dep	unspecified dependency	859	424	400	5.66
compound	compound	3864	312	298	4.49
conj	conjunct	1402	237	225	3.37
amod	adjectival modifier	3449	203	191	5.91
case	case marking	4279	184	171	7.07
cc	coordinating conjunction	1247	141	136	3.55
advmod	adverbial modifier	860	101	104	-2.97
det	determiner	2446	77	70	9.09
nsubj	nominal subject	1428	54	47	12.96

Table 5: Error reductions of two systems on PTB in different dependency relations.

Relation	Explanation	Count	Baseline	Ours	Error Reduction (%)
nmod	nominal modifier	5089	546	527	3.48
dep	unspecified dependency	762	225	196	12.89
advmod	adverbial modifier	1986	186	166	10.75
conj	conjunct	1345	117	118	-0.85
case	case marking	5869	103	97	5.83
cc	coordinating conjunction	1381	98	102	-4.08
nsubj	nominal subject	3991	91	86	5.49
advcl	adverbial clause modifier	647	71	70	1.41
acl	clausal modifier of noun	457	69	67	2.90
compound	compound	4798	66	68	-3.03

source and target domains. Much attention has been paid on bias correction which is aimed to make up the data bias between the samples of source and target domains. Importance weighting is one of bias correction methods, and it can be in different granularities such as corpus [21], document or sentence [22,23]. Feature embedding can enhance the transferability of features but it relies much on handcrafted features [24,25,26].

Koo et al. [9] derive word clusters from large unannotated corpus and adds cluster-based features into the discriminator learner. Word representations by a low dimensional value have been explored in [10,27] . Their word embeddings are not effective enough if directly used in domain adaptation, because they are general for any domain. Our domain invariant word representations tuned adversarially among the source and target domain are more useful for a specified source-target pair. We do not rely on the large amount of unlabeled data and just use sentences of training set for source and target domain. So it is not too much time-consuming in our adversarial learning.

Recently, much research has focused on learning word representations based on Wasserstein distance. Sun et al. [28] introduce non-contextual word embeddings based on Wasserstein distance and uses external information in their model. Xu et al. [29]

propose a method to learn word embeddings and topics in a unified framework. Their word embeddings are based on Euclidean distance and topics are based on Wasserstein distance. The above two works learn non-contextual word embeddings and use Wasserstein distance as their distance measures, but they do not use adversarial training. Shen et al. [26] use adversarial training to learn feature representations for domain adaptation. Same to us, their goal is make the distributions of two domains more closer. But we learn domain invariant word representations and do not rely on the feature extractor. To our knowledge, we are the first to learn domain invariant word representations by Wasserstein generative adversarial networks.

Sato et al. [30] add a domain classifier in biaffine parsing model to do adversarial training and their models include shared and domain-specific model. We do adversarial training in the stage of learning word representations and do not have any relationship with dependency parsing. So our method can be applied to a wider set of tasks.

6 Conclusions

We investigated the problem of domain adaptation for dependency parsing from a simple point of view, namely the word embedding biases between a source domain and a target domain. We showed that bridging this difference brings a large improvement to parsing accuracies. In particular, domain invariant word representations are gained using by using generative adversarial learning given a set of general word representations, so that a resource-rich domain can be leveraged for better training a model on low-resource domains. Results of news and bio-informatics data show the effectiveness of our method.

We additionally find that labeled data from low-resource domains can also improve parsing on resource-rich domains thanks to domain invariant word representations. Future work includes the investigation of domain invariant word representations to more tasks such as named entity recognition.

Acknowledgments

This work was done when the first author was visiting Westlake University. We gratefully acknowledge the funding from the project of National Key Research and Development Program of China (No.2018YFC0830700). We also thank the anonymous reviewers for their helpful comments and suggestions.

References

1. Libin Shen, Jinxi Xu, and Ralph Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *ACL 2008*.
2. David McClosky, Mihai Surdeanu, and Christopher Manning. Event extraction as dependency parsing. In *ACL 2011*.
3. Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *ACL 2017*.

4. Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. A representation learning framework for multi-source transfer parsing. In *AAAI 2016*.
5. Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL 2015*.
6. Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NIPS 2006*.
7. Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 1993*.
8. J-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus–semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 Suppl 1:i180–i182, 2003.
9. Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *ACL 2008*.
10. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
11. Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *CoRR*, abs/1701.07875, 2017.
12. Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. In *ICLR 2017*.
13. Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP 2014*.
14. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS 2014*.
15. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP 2014*.
16. Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. Stack-pointer networks for dependency parsing. In *ACL 2018*.
17. David McClosky and Eugene Charniak. Self-training for biomedical parsing. In *ACL 2008*.
18. Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. Two/too simple adaptations of Word2Vec for syntax problems. In *NAACL 2015*.
19. Yue Zhang and Stephen Clark. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP 2008*.
20. Jiangming Liu and Yue Zhang. Encoder-decoder shift-reduce syntactic parsing. In *IWPT 2017*.
21. David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *NAACL 2010*.
22. Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL 2007*.
23. Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. Instance weighting for neural machine translation domain adaptation. In *EMNLP 2017*.
24. Wenliang Chen, Min Zhang, and Yue Zhang. Distributed feature representations for dependency parsing. *IEEE/ACM TASLP 2015*.
25. Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML 2015*.
26. Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *AAAI 2018*.
27. Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL 2014*.
28. Chi Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. Gaussian word embedding with a wasserstein distance loss. *CoRR*, abs/1808.07016, 2018.
29. Hongteng Xu, Wenlin Wang, Wei Liu, and Lawrence Carin. Distilled wasserstein learning for word embedding and topic modeling. In *NIPS 2018*.
30. Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. Adversarial training for cross-domain universal dependency parsing. In *CoNLL 2017*.