Adversarial Training for Cross-Domain Universal Dependency Parsing

Motoki Sato[1] **Hitoshi Manabe**[1] **Hiroshi Noji**[1] **Yuji Matsumoto**[1,2]
[1] Nara Institute of Science and Technology
[2] RIKEN Center for Advanced Intelligence Project (AIP)
{ sato.motoki.sa7, manabe.hitoshi.me0, noji, matsu }@is.naist.jp

Abstract

We describe our submission to the CoNLL 2017 shared task, which exploits the shared common knowledge of a language across different domains via a domain adaptation technique. Our approach is an extension to the recently proposed adversarial training technique for domain adaptation, which we apply on top of a graph-based neural dependency parsing model on bidirectional LSTMs. In our experiments, we find our baseline graph-based parser already outperforms the official baseline model (UDPipe) by a large margin. Further, by applying our technique to the treebanks of the same language with different domains, we observe an additional gain in the performance, in particular for the domains with less training data.

1 Introduction

In the CoNLL 2017 shared task (Zeman et al., 2017), some language data is available in more than one treebanks typically from different annotation projects. While the treebanks differ in many respects such as the genre and the source of the text (i.e., original or translated text), the most notable difference is that the size of the treebanks often varies significantly. For example, there are three variants of English treebanks: *en*, *en_lines*, and *en_parunt*, in which the largest dataset *en* contains 12,543 training sentences while *en_lines* and *en_parunt* contain only 2,738 and 1,090 sentences, respectively.

In this paper, we describe our approach to improve the parser performance for the treebanks with lesser training data (e.g., *en_lines* and *en_parunt*), by jointly learning with the dominant treebank of the same language (e.g, *en*). We formulate our approach as a kind of domain adaptation, in which we treat the dominant treebank as the source domain while the others as the target domains.

Our approach to domain adaptation, which we call *SharedGateAdvNet*, is an extension to the recently proposed neural architecture for domain adaptation (Ganin and Lempitsky, 2015) with adversarial training (Goodfellow et al., 2014), which learns domain-invariant feature representations through an adversarial domain classifier. We extend this architecture with an additional neural layer for each domain, which captures domain-specific feature representations. To our knowledge this is the first study to apply the adversarial training-based domain adaptation to parsing.

We utilize this architecture to obtain the representation of each token of a sentence, and feed it into a graph-based dependency parsing model where each dependency arc score is calculated using bilinear attention (Dozat and Manning, 2017). Specifically, we obtain the domain-specific and domain-invariant feature representations for each token via separate bidirectional LSTMs (Bi-LSTMs), and then combine them via a gated mechanism.

Our baseline method is our reimplementation of the graph-based dependency parser with LSTMs (Dozat and Manning, 2017) trained with a single treebank. First, we observe that this model is already much stronger than the official baseline model of UDPipe (Straka et al., 2016) in most treebanks. We then apply our domain adaptation technique to the set of treebanks of the same language, and in most cases we observe a clear improvement of the scores, especially for the treebanks with lesser training data. We also try our architecture across multiple languages, i.e., a high-resource language with a large treebank, such as English,

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 71–79,
Vancouver, Canada, August 3-4, 2017.

and a low-resource language with a small data set. Interestingly, even though the mixed languages are completely different, we observe some score improvements in low-resource languages with this approach. Finally we rank the 6th place on the main result of the shared task.

2 System overview

The CoNLL 2017 shared task aims at parsing Universal Dependencies 2.0 (Nivre et al., 2017). While its concern is parsing in the wild, i.e., from the raw input text, in this work we focus only on the dependency parsing layer that receives the tokenized and POS tagged input text. For all preprocessing from sentence splitting to POS tagging, we use the provided UDPipe pipeline. For obtaining the training treebanks, we keep the gold word segmentation while assign the predicted POS tags with the UDPipe.[1]

We did a simple model selection with the development data for choosing the final submitted models. First, we trained our baseline graph-based LSTM parsing model (Section 3) independently for each treebank. Then for some languages with more than one treebank, or low-resource languages with a small treebank alone, we applied our proposed domain adaptation technique (Section 4) and obtained additional models. For treebanks for which we trained several models, we selected the best performing model on the development set in terms of LAS. For the other treebanks, we submitted our baseline graph-based parser.

For the parallel test sets (e.g., *en_pub*) with no training data, we use the model trained on the largest treebank of a target language. We did not pay much attention to the surprise languages. For Buryat (*bxr*), we just ran the model of Russian (*ru*). For the other three languages, we ran the model of English (*en*).

3 Biaffine Attention model

Our baseline model is the *biaffine* attention model (Dozat and Manning, 2017), which is an extension to the recently proposed dependency parsing method calculating the score of each arc independently from the representations of two tokens obtained by Bi-LSTMs (Kiperwasser and Goldberg, 2016). For labeled dependency parsing, this model first predicts the best *unlabeled* dependency structure, and then assigns a label to each predicted arc with another classifier. For the first step, receiving the word and POS tag sequences as an input, the model calculates the score of every possible dependency arc. To obtain a well-formed dependency tree, these scores are given to the maximum spanning tree (MST) algorithm (Pemmaraju and Skiena, 2003), which finds the tree with the highest total score of all arcs. The overview of the biaffine model is shown in Figure 1.

Let w_t be the t-th token in the sentence. As an input the model receives the word embedding $\mathbf{w}_t \in \mathbb{R}^{d_{word}}$ and POS embedding $\mathbf{p}_t \in \mathbb{R}^{d_{pos}}$ for each w_t, which are concatenated to a vector $\mathbf{x}_t$. This input is mapped by Bi-LSTMs to a hidden vector $\mathbf{r}_t$, which is then fed into an extension of bilinear transformation called a *biaffine* function to obtain the score for an arc from w_i (head) to w_j (dependent):

$$
\begin{aligned}
\mathbf{r}_t &= \text{Bi-LSTM}(\mathbf{x}_t), \qquad (1)\\
\mathbf{h}_i^{(arc-head)} &= \text{MLP}^{(arc-head)}(\mathbf{r}_i),\\
\mathbf{h}_j^{(arc-dep)} &= \text{MLP}^{(arc-dep)}(\mathbf{r}_j),\\
s_{i,j}^{(arc)} &= \mathbf{h}_i^{\mathrm{T}(arc-head)} U^{(arc)} \mathbf{h}_j^{(arc-dep)}\\
&\quad + \mathbf{h}_i^{\mathrm{T}(arc-head)} \mathbf{u}^{(arc)},
\end{aligned}
$$

where MLP is a multi layer perceptron. A weight matrix $U^{(arc)}$ determines the strength of a link from w_i to w_j while $\mathbf{u}^{(arc)}$ is used in the bias term, which controls the prior headedness of w_i.

After obtaining the best unlabeled tree from these scores, we assign the best label for every arc according to $\mathbf{s}_{i,j}^{(label)}$, in which the k-th element corresponds to the score of k-th label:

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

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

4 Domain Adaptation Techniques with Adversarial Training

Here we describe our network architectures for domain adaptation. We present two different net-

[1] We were not aware of the jack-knifed training data provided by the organizer at submission time.

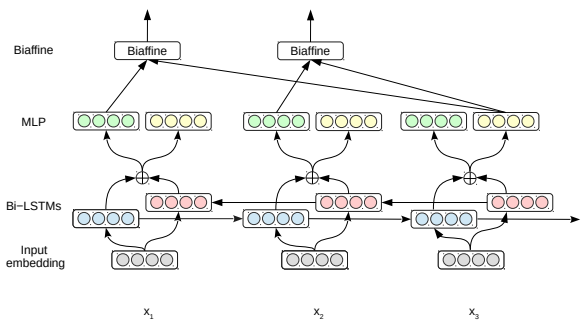

Figure 1: Overview of the biaffine model.

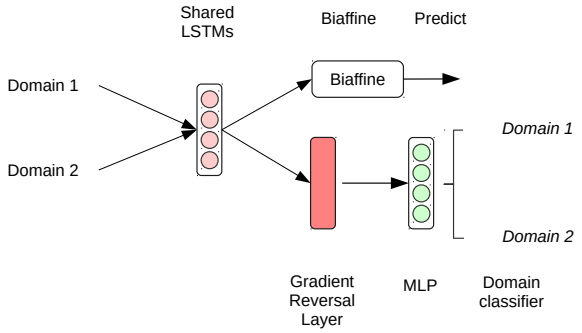

Figure 2: An application of the adversarial training for the biaffine model. In this basic architecture all domains are modeled with a common single Bi-LSTM network (Shared Bi-LSTMs).

works both using adversarial training; the main difference between them is whether we use a domain-specific feature representation for each domain. The basic architecture with adversarial training (Section 4.1) is an application of the existing domain adaptation technique (Ganin and Lempitsky, 2015) that does not employ domain-specific representations. We then extend this architecture to add a domain-specific component with a gated mechanism (Section 4.2).

In Section 5 we compare the empirical performances of these two approaches as well as several ablated settings.

4.1 Adversarial Training

Figure 2 describes the application of adversarial training (Ganin and Lempitsky, 2015) for the biaffine model. In this architecture all models for different domains are parameterized by the same LSTMs (Shared Bi-LSTMs), which output $\mathbf{r}_t$ (Eqn. 1) that are fed into the biaffine model. The key of this approach is a domain classifier, which also receives $\mathbf{r}_t$ and tries to classify the domain of the input. During training the classifier is trained to correctly classify the input domain. At the same time, we train the shared BiLSTM layers so that the domain classification task becomes harder. By this *adversarial* mechanism the model is encouraged to find the shared parameters that are not specific to a particular domain as much as possible. As a result, we expect the target domain model (with lesser training data) is trained to utilize the knowledge of the source domain effectively. Note that the domain classifier in Figure 2 is applied for every token in the input sentence.

This domain adaptation technique can be implemented by introducing the **gradient reversal layer** (GRL). The GRL has no parameters associated with it (apart from the hyper-parameter λ, which is not updated with backpropagation). During the forward propagation GRL acts as an identity transform, while during the backpropagation GRL takes the gradient from the subsequent layer, multiplies it by λ and passes it to the preceding layer.

The parameter λ controls the trade-off between the two objectives (the domain classifier and the biaffine model) that shape the feature representation during training. The GRL layer $R_\lambda(\mathbf{x})$ for the forward and backward propagation is defined as follows:

$$R_\lambda(\mathbf{x}) = \mathbf{x}; \quad \frac{dR_\lambda}{d\mathbf{x}} = -\lambda\mathbf{I}.$$

4.2 Shared Gated Adversarial Networks

Now we present our extension to adversarial training described above, which we call the shared gated adversarial networks (*SharedGateAdvNet*).

Figure 3 shows the overall architecture. The largest difference from Figure 2 is the existence of the domain-specific Bi-LSTMs (Not-shared Bi-LSTMs) that we expect to capture the representations not fitted into the shared LSTMs and specialized to a particular domain. The model comprises of the following three components.

Shared Bi-LSTMs As in the basic model with adversarial training (Figure 2), the shared Bi-LSTMs in this model try to learn the domain invariant feature representation via a domain classifier, which facilitates effective domain adaptation.

Domain-specific Bi-LSTMs This domain-specific component captures the information that does not fit into the domain-invariant shared

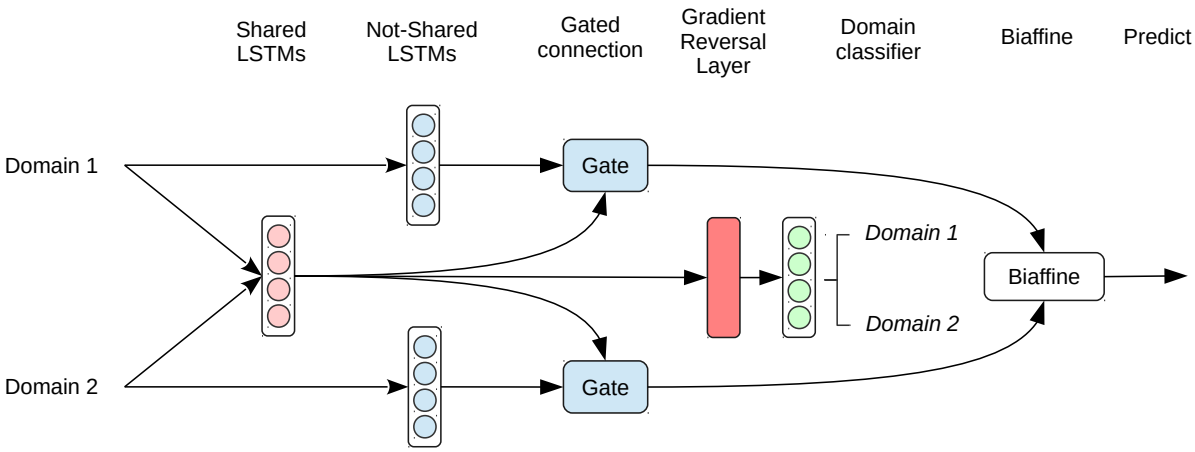

Figure 3: The architecture of SharedGateAdvNet. Each token in the input sentence is passed to the biaffine model through this network.

Bi-LSTMs. The domain-specific LSTMs exist on each domain. Figure 3 shows the case when two treebanks (domains) are trained at the same time. When training with three treebanks, there exist three Bi-LSTMs, one for each domain.

Gated connection The gated connection selects which information to use between the domain-invariant and domain-specific feature representations (the shared Bi-LSTMs and the domain-specific Bi-LSTMs). We get the combined representation $\mathbf{r}_t^{gate}$ from these two vectors as follows:

$$\begin{aligned} \mathbf{g}_t &= \sigma(U^{(gate)}(\mathbf{r}_t^{dom} \oplus \mathbf{r}_t^{share}) + \mathbf{u}^{(gate)}), \\ \mathbf{r}_t^{gate} &= \mathbf{g}_t \cdot \mathbf{r}_t^{share} + (1 - \mathbf{g}_t) \cdot \mathbf{r}_t^{dom}, \end{aligned}$$

where σ is the sigmoid function. $\mathbf{r}^{share}$ is the output of the shared Bi-LSTMs while $\mathbf{r}^{dom}$ is the output of the domain-specific Bi-LSTMs.

5 Experiments

5.1 Settings

For the settings of the biaffine models, we follow the same network settings as Dozat and Manning (2017): 3-layer, 400-dimentional LSTMs for each direction, 500-dimentional MLP for arc prediction, and 100-dimentional MLP for label prediction. We use the 100-dimensional pre-trained word embeddings trained by word2vec (Mikolov et al., 2013) [2] and the 100-dimensional randomly initialized POS tag embeddings. For the model with adversarial training, we fix λ to 0.5.[3] We apply dropout (Srivastava et al., 2014) with a 0.33 rate at the input and output layers. For optimization, we use Adam (Kingma and Ba, 2014) with the batch size of 128 and gradient clipping of 5. We use early stopping (Caruana et al., 2001) based on the performance on the development set.

5.2 Preliminary Experiment

Before selecting the final submitted model for each treebank (Section 5.3) here we perform a small experiment on selected languages (English and French) to see the effectiveness of our domain adaptation techniques.

English experiment First, for English, we compare the performances of several domain adaptation techniques as well as the baselines without adaptation, to see which technique performs better. We compare the following six systems:

- **UDPipe**: The official baseline parser (Straka et al., 2016). This is a transition-based parser selecting each action using neural networks.
- **Biaffine**: Our reimplementation of the graph-based parser of Dozat and Manning (2017). We use Chainer (Tokui et al., 2015) for our implementation. We train this model independently on each treebank.
- **Biaffine-MIX**: A simple baseline of domain adaptation, which just trains a single biaffine model across different domains. We obtain

[2] The pre-trained word embeddings are provided by the CoNLL 2017 Shared Task organizers. These are trained with CommonCrawl and Wikipedia.

[3] We did not obtain any performance gains by scheduling λ in our preliminary experiments.

Method	en		en_lines		en_partut	
	UAS	LAS	UAS	LAS	UAS	LAS
UDPipe	83.83	80.13	79.00	74.68	78.26	74.23
Biaffine	86.19	82.45	81.94	77.64	80.01	75.52
Biaffine-MIX	86.19	82.28	82.23	76.95	83.34	78.02
Biaffine-MIX + Adv	86.05	82.36	82.47	77.45	83.38	78.37
SharedGateNet	86.17	82.42	82.67	**78.28**	84.06	80.09
SharedGateAdvNet	**86.27**	**82.47**	**82.69**	78.27	**84.32**	**80.35**

Table 1: The result of our preliminary English experiment across multiple domains. UDPipe and Biaffine are trained separately for each language, while the other models are trained across all domains jointly.

Method	en		en_lines		en_partut		fr		fr_partut		fr_sequoia	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Biaffine	86.19	82.45	81.94	77.64	80.01	75.52	**90.39**	**88.11**	86.79	83.70	87.20	84.90
Ours (domain)	**86.27**	**82.47**	**82.69**	**78.27**	**84.32**	**80.35**	90.26	88.06	88.71	86.18	87.64	85.27
Ours (domain, lang)	85.47	81.75	82.03	77.58	83.60	79.51	90.07	87.71	**89.49**	**87.08**	**87.90**	**85.34**

Table 2: The result of our preliminary experiment across different languages (English and French). Ours (domain) is trained for each language across multiple domains by SharedGateAdvNet. Ours (domain, lang) is trained with all six treebanks of two languages jointly. Joint training of two languages brings a small improvement on the smaller French treebanks (fr_partut and fr_sequoia).

this by removing the domain classification component in Figure 2.

- **Biaffine-MIX + Adv**: The model in Figure 2, which shares the same parameters across multiple domains but adversarial training facilitates learning domain-invariant representations.
- **SharedGateNet**: A simpler version of our proposed architecture (Figure 3), which does not have the adversarial component but has the gated unit controlling the strength of the two, domain-invariant and domain-specific Bi-LSTMs.
- **SharedGateAdvNet**: Our full architecture (Figure 3) with both adversarial training and the gated unit.

The result is shown in Table 1. First, we find that our baseline biaffine parser already outperforms the official baseline parser (UDPipe) by a large margin (e.g., for English, 82.45 vs. 80.13 LAS), which suggests the strength of graph-based parsing with Bi-LSTMs that enable the model to capture the entire sentence as a context. By just mixing the training treebanks (Biaffine-MIX), we observe a score improvement for the domains with less data, *en_lines* and *en_parunt*, which only contain 2,738 and 1,090 sentences, respectively. We also observe an additional small gain with adversarial training (Biaffine-MIX Adv). Comparing with this, our proposed architectures (SharedGateNet and SharedGateAdvNet) perform better. This shows the importance of having the domain-specific network layers. Our final architecture SharedGateAdvNet slightly outperforms SharedGateNet, indicating that the adversarial technique also has its own advantage.

Since SharedGateAdvNet consitently outperforms the others in English, we only try this method in the following experiments.

English and French experiment To see the effects of our approach when combining completely different data, i.e., different language treebanks, we perform a small experiment using two languages: English and French. French treebanks are also divided into three domains and also are imbalanced: *fr* (14,553 sentences), *fr_partut* (620 sentences), and *fr_sequoria* (2,231 sentences). We compare the models trained within each language (3 domains for each), and the model trained with all six treebanks of English and French. The result is shown in Figure 2. Interestingly, especially for *fr_partut* and *fr_sequoria*, we observe a small score improvement by jointly learning two languages. The best model for *fr* is the biaffine model without joint training. Note also that for *en*, the effect of the adaptation technique is very small, or

Language	UAS			LAS		
	UDPipe	Biaffine	SharedGateAdv	UDPipe	Biaffine	SharedGateAdv
ar	80.13	**82.97**	-	73.04	**76.35**	-
bg	88.02	**90.60**	-	83.22	**86.23**	-
ca	88.37	**90.57**	-	85.28	**87.42**	-
cs	88.14	**91.36**	91.23	84.68	**88.31**	88.29 (i)
cs_cac	86.81	89.03	**89.77**	83.50	85.65	**86.73** (i)
cs_cltt	72.65	75.88	**78.87**	69.01	71.43	**75.23** (i)
cu	80.41	**82.83**	-	73.19	**75.30**	-
da	78.40	**82.04**	-	74.74	**78.42**	-
de	79.40	**84.40**	-	74.11	**80.17**	-
el	82.37	**84.99**	-	78.69	**80.88**	-
en	83.83	86.19	**86.27**	80.13	82.45	**82.47** (i)
en_lines	79.00	81.94	**82.69**	74.68	77.64	**78.27** (i)
en_partut	78.26	80.01	**84.32**	74.23	75.52	**80.35** (i)
es	87.50	89.37	**89.49**	84.29	86.25	**86.33** (i)
es_ancora	87.53	**90.34**	90.26	84.54	**87.76**	87.61 (i)
et	69.26	**70.57**	-	**60.16**	59.01	-
eu	74.59	**77.58**	-	69.23	**70.06**	-
fa	83.89	**87.02**	-	79.81	**83.15**	-
fi	79.97	80.57	**81.74**	75.37	74.62	**75.98** (i)
fi_ftb	80.72	81.60	**82.21**	76.10	75.38	**76.20** (i)
fr	88.72	**90.39**	90.07	86.36	**88.11**	87.71 (ii)
fr_partut	77.82	86.79	**89.49**	73.67	83.70	**87.08** (ii)
fr_sequoia	84.35	87.20	**87.89**	81.93	84.90	**85.33** (ii)
ga	74.57	**80.28**	-	63.47	**72.70**	-
gl	80.66	83.73	**83.81**	77.17	80.08	**80.11** (i)
gl_treegal	72.32	78.48	**83.69**	66.43	73.62	**78.08** (i)
got	76.42	**78.21**	-	68.92	**70.32**	-
grc	62.12	66.13	**69.16**	55.23	58.77	**61.23** (i)
grc_proiel	77.35	80.74	**81.83**	72.03	75.43	**76.22** (i)
he	83.93	**86.12**	-	78.03	**80.13**	-
hi	91.29	**93.03**	-	86.90	**88.97**	-
hr	81.99	**85.66**	-	76.40	**79.79**	-
hu	71.52	**74.09**	-	**65.04**	63.31	-
id	80.76	**83.34**	-	74.08	**76.67**	-
it	87.77	90.09	**90.82**	85.04	87.62	**88.15** (i)
it_partut	82.02	82.66	**90.66**	78.47	78.72	**87.34** (i)
ja	94.31	**95.48**	-	92.94	**94.15**	-
kk	37.08	**64.51**	-	23.60	**37.09**	-
ko	63.71	**67.01**	-	56.41	**59.04**	-
la	56.93	67.41	**73.06**	47.13	58.65	**64.53** (i)
la_ittb	75.95	82.82	**83.29**	71.07	78.24	**78.69** (i)
la_proiel	75.31	79.30	**80.34**	69.11	72.83	**73.74** (i)
lv	56.93	**71.98**	-	47.13	**63.45**	-
nl	79.57	**85.44**	85.38	74.55	80.21	**80.28** (i)
nl_lassysmall	79.59	85.01	**86.59**	75.46	81.17	**82.53** (i)
no_bokmaal	87.52	**90.17**	-	84.38	**87.40**	-
no_nynorsk	85.79	**88.51**	-	82.49	**85.54**	-
pl	85.18	**86.84**	-	79.01	**81.15**	-
pt	88.37	90.75	**91.02**	85.20	**87.84**	87.79 (i)
pt_br	88.37	90.53	**90.93**	86.26	88.36	**88.75** (i)
ro	85.22	**88.04**	-	79.66	**82.35**	-
ru	80.13	82.79	**84.01**	75.07	77.14	**78.98** (i)
ru_syntagrus	89.69	**92.07**	91.86	86.84	**89.48**	89.23 (i)
sk	81.81	**84.09**	-	75.55	**77.41**	-
sl	81.81	87.12	**87.17**	80.72	**83.80**	83.73 (i)
sl_sst	63.71	77.56	**80.17**	55.39	69.67	**72.28** (i)
sv	77.94	80.32	**82.27**	73.64	75.28	**77.91** (i)
sv_lines	79.72	79.81	**82.71**	74.72	74.23	**77.87** (i)
tr	63.41	**64.77**	-	**55.70**	53.88	-
ug	62.50	74.80	**75.57**	38.46	52.67	**52.68** (iii)
uk	62.66	78.05	**79.59**	54.17	71.39	**72.93** (iii)
ur	83.05	**85.69**	-	76.15	**78.87**	-
vi	63.99	**65.98**	-	56.34	**57.91**	-
zh	74.03	**77.09**	-	68.75	**71.38**	-

Table 3: The result of our experiment for model selection on the development data. (i), (ii), and (iii) correspond to the differnt domain adaptation strategies found in the body.

negative, and these suggest our approach may be ineffective for a treebank that already contains sufficient amount of data.

Due to time constraints, we were unable to try many language pairs for joint training, but this result suggests the parser may benefit from training across different languages. For the final experiment for model selection below, we try some other pairs for some languages, and select those models when they perform better.

5.3 Model Selection

As we summarize in Section 2 we perform a simple model selection for each language with the development data in order to select the final submitted models. Besides a biaffine model with a single treebank, for some treebanks we additionally train other models with our SharedGateAdvNet. Our approach is divided into the following three strategies according to the languages:

(i) Training multiple domains within a single language. We try this for many languages, such as English (*en*), Czech (*cs*), Spanish (*es*), etc.

(ii) Training multiple domains *across* different languages. Based on the positive result of our preliminary experiment, we try this only for obtaining French models (training jointly with English treebanks).

(iii) Training two treebanks in different languages. We only try this for two very small treebanks: Ukrainian (*uk*), which we train with *en*, and Uyghur (*ug*), which we train with Russian (*ru*) that we find performs better than training with *en*.

See Table 3 for the results. Again, our baseline biaffine parser outperforms UDPipe in most treebanks. Training multiple domains in one language often brings performance gains, in particular for smaller treebanks, as in the case for English. For example, for Galician, LAS in *gl_treegal* largely improves from 73.63 to 78.08 with our joint training. The largest gain is obtained in Italian (*it_partut*), from 78.72 to 87.34, about 10 points improvement in LAS.

From these results, for example, we select our SharedGateAdvNet model for *cs_cac* while select the biaffine model for *cs*, which does not benefit from joint training.

Language	UAS			LAS		
	UDPipe	Ours	Stanford	UDPipe	Ours	Stanford
ar	71.19	**73.34**	76.59	65.30	**67.78**	71.96
ar_pub	53.55	**55.66**	58.87	43.14	**45.56**	49.50
bg	87.79	**89.95**	92.89	83.64	**85.97**	89.81
bxr	**46.97**	44.07	51.19	**31.50**	27.20	30.00
ca	88.62	**90.59**	92.88	85.39	**87.47**	90.70
cs	86.46	**89.74**	92.62	82.87	**86.50**	90.17
cs_cac	86.49	**90.09**	93.14	82.46	**86.41**	90.43
cs_cltt	76.26	**81.10**	86.02	71.64	**77.14**	82.56
cs_pub	84.42	**87.22**	89.11	79.80	**82.30**	84.42
cu	69.68	**72.19**	77.10	62.76	**65.13**	71.84
da	76.94	**80.90**	85.33	73.38	**77.08**	82.97
de	74.27	**78.43**	84.10	69.11	**74.04**	80.71
de_pub	73.64	**77.20**	80.88	66.53	**70.74**	74.86
el	83.00	**85.48**	89.73	79.26	**81.79**	87.38
en	78.87	**80.96**	84.74	75.84	**77.93**	82.23
en_lines	77.39	**81.87**	85.16	72.94	**77.53**	82.09
en_partut	77.83	**83.17**	86.10	73.64	**79.10**	82.54
en_pub	82.74	**84.75**	88.22	78.95	**81.18**	85.51
es	84.84	**87.80**	90.01	81.47	**84.25**	87.29
es_ancora	86.97	**89.93**	92.11	83.78	**87.27**	89.99
es_pub	84.71	**86.91**	88.14	77.65	**79.66**	81.05
et	67.71	**69.00**	78.08	**58.79**	57.72	71.65
eu	74.39	**77.94**	85.28	69.15	**70.71**	81.44
fa	83.36	**86.06**	89.64	79.24	**82.01**	86.31
fi	77.90	**80.17**	87.97	73.75	**74.71**	85.64
fi_ftb	78.77	**80.43**	89.24	74.03	**74.42**	86.81
fi_pub	82.24	**82.40**	90.60	**78.65**	77.11	88.47
fr	84.13	**85.88**	88.57	80.75	**82.43**	85.51
fr_partut	81.69	**84.79**	88.64	77.38	**80.31**	85.05
fr_pub	78.62	**80.32**	83.45	73.63	**75.20**	78.81
fr_sequoia	82.62	**85.85**	88.48	79.98	**83.10**	86.53
ga	72.08	**73.29**	78.50	61.52	**62.25**	70.06
gl	80.66	**83.51**	85.87	77.31	**80.13**	83.23
gl_treegal	71.17	**73.60**	78.28	65.82	**66.84**	73.39
got	67.13	**67.84**	73.10	59.81	**60.20**	66.82
grc	62.74	**68.85**	78.42	56.04	**61.28**	73.19
grc_proiel	70.42	**74.33**	78.30	65.22	**69.23**	74.25
he	61.54	**64.16**	67.70	57.23	**59.56**	63.94
hi	90.97	**92.64**	94.70	86.77	**88.70**	91.59
hi_pub	63.43	**65.29**	67.24	50.85	**52.81**	54.49
hr	83.20	**85.47**	90.11	77.18	**79.32**	85.25
hsb	**61.70**	49.38	67.83	**53.83**	41.32	60.01
hu	71.46	**71.87**	82.35	**64.30**	60.30	77.56
id	80.91	**83.11**	85.17	74.61	**76.50**	79.19
it	88.03	**89.93**	92.51	85.28	**87.39**	90.68
it_pub	87.04	**88.61**	91.08	83.70	**85.30**	88.14
ja	73.52	**74.46**	75.42	72.21	**73.27**	74.72
ja_pub	77.13	**77.65**	78.64	76.28	**76.78**	77.92
kk	**41.92**	40.12	43.51	**24.51**	22.49	25.13
kmr	**46.20**	31.68	47.71	**32.35**	23.18	35.05
ko	66.40	**71.48**	85.90	59.09	**64.46**	82.49
la	54.35	**63.60**	72.56	43.77	**52.19**	63.37
la_ittb	80.78	**85.53**	89.44	76.98	**82.20**	87.02
la_proiel	63.50	**67.68**	73.71	57.54	**61.34**	69.35
lv	67.14	**68.83**	79.26	59.95	**60.20**	74.01
nl	74.94	**79.16**	85.17	68.90	**73.22**	80.48
nl_lassysmall	81.37	**87.74**	89.56	78.15	**85.03**	87.71
no_bokmaal	86.14	**88.55**	91.60	83.27	**86.05**	89.88
no_nynorsk	84.88	**87.22**	90.75	81.56	**84.39**	88.81
pl	85.08	**86.89**	93.98	78.78	**80.68**	90.32
pt	85.77	**87.70**	89.90	82.11	**84.35**	87.65
pt_br	87.75	**90.06**	92.76	85.36	**87.73**	91.36
pt_pub	80.10	**82.58**	83.27	73.96	**76.35**	77.14
ro	85.50	**87.28**	90.43	79.88	**81.66**	85.92
ru	79.28	**82.75**	87.15	74.03	**77.63**	83.65
ru_syntagrus	89.30	**91.68**	82.31	86.76	**89.31**	75.71
ru_pub	75.67	**77.93**	94.00	68.31	**70.51**	92.60
sk	78.14	**81.11**	89.58	72.75	**75.28**	86.04
sl	84.68	**87.35**	93.34	81.15	**84.06**	91.51
sl_sst	53.79	**57.47**	61.71	46.45	**50.16**	56.02
sme	**46.06**	37.74	51.13	**30.60**	23.54	37.21
sv	80.78	**83.71**	88.50	76.73	**79.68**	85.87
sv_lines	79.18	**81.49**	86.51	74.29	**76.63**	82.89
sv_pub	75.09	**77.47**	81.90	70.62	**72.89**	78.49
tr	60.48	**62.48**	69.62	**53.19**	51.44	62.79
tr_pub	**55.01**	52.95	58.72	**34.53**	31.17	37.72
ug	**53.58**	51.45	56.86	**34.18**	33.19	39.79
uk	69.78	**70.22**	81.44	**60.76**	60.73	75.33
ur	83.67	**86.40**	87.98	76.69	**79.38**	82.28
vi	42.12	**44.01**	46.14	37.47	**38.99**	42.13
zh	61.50	**63.87**	68.95	57.40	**59.99**	65.88
AVG.	74.40	**76.35**	81.30	68.35	**70.13**	76.29

Table 4: The main result on the test data.

5.4 Evaluation on Test data

The main result of CoNLL 2017 shared task on the test data is shown in Table 4. In addition to the official baseline (UDPipe) and our system, we also report the scores of the winning system by the Stanford team. See Zeman et al. (2017) for the overview of the other participating systems.

Our system outperforms UDPipe in many test treebanks, 69 out of 81 treebanks. We find many cases that UDPipe performs better are when the training teebank is very small, e.g., Kazakh (*kk*), Ukrainian (uk), and Uyghur (ug), or not available at all, i.e., surprise languages: Buryat (*bxr*), Kurmanji (*kmr*), Upper Sorbian (*hsb*), and North Sàmi (*sme*), for which our approach is somewhat naive (Section 2) and UDPipe performs always better. We can also see that for some treebanks (e.g., *et*, *fi_pub* and *hu*), our system performs better in UAS while worse in LAS. This may be due to the design of the baseline biaffine model, which determines the best unlabeled tree before assigning the labels (Section 3), i.e., does not perform labeled parsing as a single task.

Our system (NAIST-SATO) achieves the overall average LAS of 70.13, which is the 6th rank among 33 participants in the shared task. UDPipe (68.35) is the 13th rank.

6 Related Work

A related approach to us in parsing is Ammar et al. (2016), where a single multilingual dependency parser parses sentences in several languages. Differently from our final architecture their model shares all parameters across different languages. In this study, we found the importance of modeling language-specific syntactic features explicitly with the separate Bi-LSTMs.

Our network architecture for domain adaptation is an extension of Ganin and Lempitsky (2015), which applies adversarial training (Goodfellow et al., 2014) for the domain adaptation purpose. There is little prior work applying this adversarial domain adaptation technique to NLP tasks; Chen et al. (2016) use it for cross-lingual sentiment classification, in which the adversarial component has a classifier that tries to classify the language of an input sentence. To our knowledge, this is the first study applying adversarial training for parsing. In addition to the simple application, we also proposed an extended architecture with the domain specific LSTMs and demonstrated the importance of them.

7 Conclusion

We have proposed a domain adaptation technique with adversarial training for parsing. By applying it on the recent state-of-the-art graph-based dependency parsing model with Bi-LSTMs, we obtained a consistent score improvement, especially for the treebanks having less training data. For the architecture design, we found the importance of preparing the network layer capturing the domain specific representation. We also performed a small experiment for training across multiple languages and had an encouraging result. In this work, we have not investigated incorporating information on language families, so a natural future direction would be to investigate whether typological knowledge helps to select good combinations of languages for training multilingual models.

Acknowledgments

We thank two anonymous reviewers for helpful comments. This work was in part supported by JSPS KAKENHI Grant Number 16H06981.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics (TACL)* 5:431–444.

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, MIT Press, pages 402–408.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614* .

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *Proc. International Conference on Learning Representations (ICLR)* .

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* .

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague. http://hdl.handle.net/11234/1-1983.

Sriram Pemmaraju and Steven S Skiena. 2003. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica®*. Cambridge university press.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Paris, France.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.