Enhancing Structure-aware Encoder with Extremely Limited Data for Graph-based Dependency Parsing

Yuanhe Tian♥, **Yan Song**♠†, **Fei Xia**♥
♥University of Washington ♠University of Science and Technology of China
♥{yhtian, fxia}@uw.edu ♠clksong@gmail.com

Abstract

Dependency parsing is an important fundamental natural language processing task which analyzes the syntactic structure of an input sentence by illustrating the syntactic relations between words. To improve dependency parsing, leveraging existing dependency parsers and extra data (e.g., through semi-supervised learning) has been demonstrated to be effective, even though the final parsers are trained on inaccurate (but massive) data. In this paper, we propose a frustratingly easy approach to improve graph-based dependency parsing, where a structure-aware encoder is pre-trained on auto-parsed data by predicting the word dependencies and then fine-tuned on gold dependency trees, which differs from the usual pre-training process that aims to predict the context words along dependency paths. Experimental results and analyses demonstrate the effectiveness and robustness of our approach to benefit from the data (even with noise) processed by different parsers, where our approach outperforms strong baselines under different settings with different dependency standards and model architectures used in pre-training and fine-tuning. More importantly, further analyses find that only 2K auto-parsed sentences are required to obtain improvement when pre-training vanilla BERT-large based parser without requiring extra parameters.[1]

1 Introduction

Dependency parsing aims to produce the syntactic structure of a sentence by illustrating the syntactic relations between words, where the words with dependency relations are connected by directed and labeled arcs. It is an important fundamental natural language processing (NLP) task that is widely used to enhance downstream NLP tasks (Cai et al., 2009; Strubell et al., 2018; Huang and Carley, 2019; Zhang et al., 2019; Guo et al., 2019; Nie et al., 2020; Zhou et al., 2020b; Chen et al., 2020; Tian et al., 2022) such as coreference resolution, relation extraction, and sentiment analysis.

To produce the dependency structure of a sentence, the contextual information is of great importance to achieve good model performance. Thus, most recent studies (Dozat and Manning, 2017; Zhou and Zhao, 2019; Zhou et al., 2020a,b; Mrini et al., 2020; Zhang et al., 2021) leverage advanced encoders (e.g., bi-LSTM, Transformer (Vaswani et al., 2017)) to model the contextual information of the input and obtain outstanding performance. In addition, because leveraging different models to obtain better results is an important technique for many NLP tasks (Juraska et al., 2018; Kobayashi, 2018; Kuwabara et al., 2020; Qin et al., 2021), many previous studies apply this technique to dependency parsing to further improve model performance. Under this paradigm, many studies utilize semi-supervised methods (e.g., self-training) to benefit from auto-processed extra data which is used to extract useful features (Smith and Eisner, 2007; Koo et al., 2008; Bansal and Klein, 2011; Ma and Xia, 2013; Kiperwasser and Goldberg, 2015; Yu and Bohnet, 2017) or training data (Spreyer and Kuhn, 2009; Rybak and Wróblewska, 2018; Rotman and Reichart, 2019). However, since the auto-generated parse tree is not always accurate, semi-supervised methods need to handle the noise with care to achieve better performance (Søgaard and Rishøj, 2010; Chen et al., 2018).

To address the noise issue, in this paper, we propose to apply pre-training and fine-tuning to enhance dependency parsing with the auto-parsed data generated by existing parsers. Although the effectiveness of pre-training and fine-tuning paradigm has been demonstrated to leverage extra data in many NLP tasks, it is still worth studying whether this paradigm works well for dependency

†Corresponding author.
[1]Our code is available at https://github.com/synlp/DMPar.

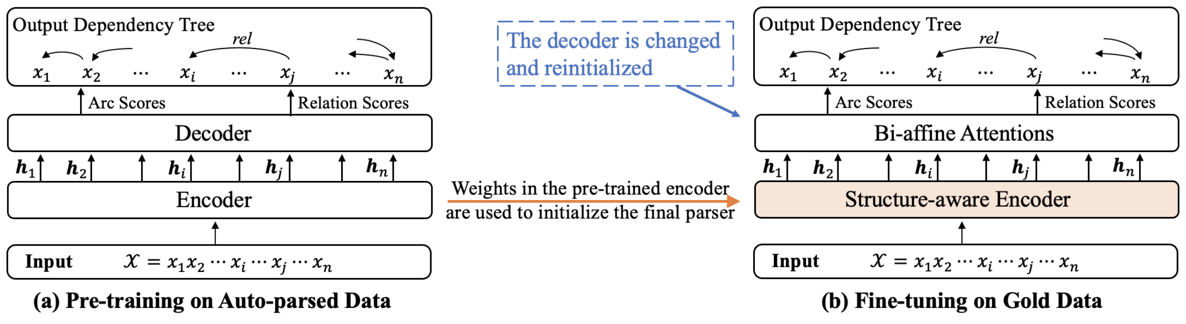
Output Dependency Tree
rel
x_1 x_2 ⋯ x_i ⋯ x_j ⋯ x_n
Arc Scores
Relation Scores
Decoder
h_1 h_2 h_i h_j h_n
Encoder
Input $\mathcal{X} = x_1x_2 \cdots x_i \cdots x_j \cdots x_n$
(a) Pre-training on Auto-parsed Data
The decoder is changed and reinitialized
Weights in the pre-trained encoder are used to initialize the final parser
Output Dependency Tree
Bi-affine Attentions
Structure-aware Encoder
(b) Fine-tuning on Gold Data

Figure 1: Our parser is trained in two stages: pre-training with auto-parsed data (Figure 1(a)) and fine-tuning with gold dependency trees (Figure 1(b)). Fine-tuning uses the same encoder architecture as in pre-training but further adjusts its weights. In contrast, the decoder for fine-tuning is different from the one in pre-training and its weights are randomly initialized.

parsing. Specifically, we apply an auto-parser[2] to unlabeled data to obtain the auto-parsed dependencies, and then use the resulting data (with noise) to pre-train a structure-aware encoder, which is finally fine-tuned with the gold labels. The pre-training of the encoder follows exactly the same process of training a dependency parser with the same input and output, except that the labeled data are automatically generated, which significantly differs from the usual pre-training process that aims to predict the context words along the dependency path. In the fine-tuning stage, the weights (with structural information learnt from noisy auto-parsed data) of the pre-trained encoder is used to initialize the encoder of our final parser, whereas the final parser's decoder is initialized randomly before fine-tuning. In doing so, the encoder is able to learn the dependency information from large auto-parsed data (with noise) through pre-training and then use the information to enhance the performance of the final parser when it is fine-tuned on the gold parse trees. Compared with previous studies, our method offers a more flexible way to selectively learn from the auto-parsed data than the methods that take dependency parses (with noise) as fixed extra input features or training instances. Experimental results and further analyses on English benchmark datasets demonstrate the effectiveness and robustness of the proposed approach, which outperforms strong baselines under different settings with different dependency standards and model architectures used in pre-training and fine-tuning. The most interesting finding from this study is that the pre-training step only needs a small amount of data (e.g., two thousand auto-parsed sentences for the BERT-large encoder), to improve the performance of the resulting parser.

[2]E.g., Stanford CoreNLP Toolkits (Manning et al., 2014).

2 Training the Dependency Parser

In this study, we use neural graph-based dependency parsers, because they have achieved state-of-the-art performance on this task (Dozat and Manning, 2017; Zhou and Zhao, 2019; Mrini et al., 2020). Specifically, training our graph-based parser follows a two-stage procedure with pre-training and fine-tuning, where pre-training is performed on auto-parsed data with the same object to train a dependency parser, and the fine-tuning is conducted on the gold dependency trees with the pre-trained encoder (other modules in the final parser are freshly learned in fine-tuning).

Figure 1 shows the model architecture and the two-stage procedure to train our final dependency parser. In the following text, we firstly introduce the neural graph-based dependency parser and then illustrate the process to pre-train the structure-aware encoder.

2.1 Neural Graph-based Dependency Parser

Given the input sentence $\mathcal{X} = x_1 x_2 \cdots x_i \cdots x_n$ (x_i is the i-th word), conventional neural graph-based dependency parsers firstly obtain the hidden vector $\mathbf{h}_i$ for each word x_i from the encoder. Then, based on $\mathbf{h}_i$, for each word pair (x_i, x_j), the decoder of the parser computes $s^{arc}_{i,j}$ and $s^{rel}_{i,j}$ indicating the score for the directional dependency connection (arc) between x_i and x_j and the score for the dependency relation type $rel \in \mathcal{R}$ ($\mathcal{R}$ is the dependency type set) between them, respectively. Next, the parser applies the Eisner algorithm[3] (Eisner, 1996) to all $s^{arc}_{i,j}$ to predict the dependency tree $\widehat{\mathcal{T}}_0$ and assigns the connection between x_i and its

[3]The Eisner algorithm is only applied in inference. In training, the parser is optimized by comparing $s^{arc}_{i,j}$ and $s^{rel}_{i,j}$ with the gold standards using the cross-entropy loss function.

head x_j with the dependency type $\widehat{r}_{i,j}$ having the highest score $s^{rel}_{i,j}$.

It is worth noting that there are many ways to obtain the dependency arc scores $s^{arc}_{i,j}$ and the dependency relation scores $s^{rel}_{i,j}$. In doing so, bi-affine attentions (Dozat and Manning, 2017) (as illustrated in Figure 1(b)) is the most common and effective way to obtain $s^{arc}_{i,j}$ and $s^{rel}_{i,j}$. Specifically, for $s^{arc}_{i,j}$, it is computed by

$$\mathbf{h}^{arc\text{-}d}_i = \mathrm{MLP}^{\,arc\text{-}d}(\mathbf{h}_i) \quad (1)$$

$$\mathbf{h}^{arc\text{-}h}_j = \mathrm{MLP}^{\,arc\text{-}h}(\mathbf{h}_j) \quad (2)$$

$$s^{arc}_{i,j} = (\mathbf{h}^{arc\text{-}d}_i \oplus [1])^{\top} \mathbf{W}^{arc} (\mathbf{h}^{arc\text{-}h}_j \oplus [1]) \quad (3)$$

where MLP $^{arc\text{-}h}$ and MLP $^{arc\text{-}d}$ denote multi-layer perceptrons for the head and dependent representations, respectively; $\mathbf{W}^{arc}$ is a trainable matrix; $\oplus$ is the vector concatenation operation; [1] is a one-dimensional unit vector which serves as a bias term for $\mathbf{h}^{arc\text{-}d}_i$ and $\mathbf{h}^{arc\text{-}h}_j$. Following the aforementioned process, $s^{rel}_{i,j}$ for a particular dependency type $rel \in \mathcal{R}$ is computed in a similar way.

2.2 Pre-training Structure-aware Encoder

To leverage existing parsers and unlabeled data, we replace the original encoder with a structure-aware encoder. The encoder is pre-trained with dependency trees generated from an existing parser, following the same but simplified procedure (as shown in Figure 1(a) without using bi-affine attentions) of training a parser. That is, in the pre-training procedure, we use

$$s^{arc}_{i,j} = \mathbf{h}_i^{\top} \mathbf{W}^{arc} \mathbf{h}_j, \quad \mathbf{s}^{r}_{i,j} = \mathbf{W}^{rel}(\mathbf{h}_i \oplus \mathbf{h}_j) \quad (4)$$

to compute the arc score $s^{arc}_{i,j}$ and the relation score vector $\mathbf{s}^{r}_{i,j}$ over all dependency relation types. Herein, each dimension of $\mathbf{s}^{r}_{i,j}$ corresponds to a particular dependency relation type in $\mathcal{R}$ and $\mathbf{W}^{arc}$ and $\mathbf{W}^{rel}$ denote two trainable matrices.

Once the model is pre-trained, we get rid of the $\mathbf{W}^{arc}$ and $\mathbf{W}^{rel}$ and combine the resulting encoder with a new randomly initialized bi-affine attention module to construct our final dependency parser (illustrated in Figure 1(b)) for fine-tuning.

Through pre-training, the encoder is able to learn dependency information from the auto-parsed data (with noise). Meanwhile, because the decoder (i.e., the bi-affine attentions) of the final parser is changed and randomly initialized without using the decoder parameters (i.e., $\mathbf{W}^{arc}$ and $\mathbf{W}^{rel}$ in Eq. (4)) obtained from pre-training, our final parser is able to optimize its parameters based on the gold standard trees. Therefore, by using the auto-parsed and the gold training data in different stages (i.e., pre-training and fine-tuning, respectively), the noise in the auto-parsed data is carefully addressed: errors learnt from the pre-training stage can be "fixed" in the fine-tuning stage. In contrast, many existing semi-supervised approaches train the final parser on the combination of auto-parsed and gold training data, which could be risky.

Datasets		Sent. #	Token #	ASL
PTB	Train	40K	950K	23.9
	Dev	2K	40K	23.6
	Test	2K	57K	23.5
UD	Train	13K	205K	16.3
	Dev	2K	25K	12.6
	Test	2K	25K	12.1
Brown (Full)		24K	458K	19.0
English Wiki		92M	2,380M	22.3

Table 1: The number of sentences, tokens, and the averaged sentence length (ASL) of PTB, UD, Brown, and English Wiki used in our experiments.

3 Experiments

3.1 Datasets

In the experiments, we use English Wiki (with 92M sentences and 2,380M tokens) as the raw data for pre-training. We follow previous studies (Dozat and Manning, 2017; Zhou and Zhao, 2019; Mrini et al., 2020) to use English Penn Treebank (PTB)[4] (Marcus et al., 1993) converted by version 3.3.0 of the Stanford Dependency converter[5] as the benchmark dataset, which is further split into train/dev/test sets. In addition, we use Brown corpus (Marcus et al., 1993) and the English Web Treebank of Universal Dependencies (UD)[6] (Nivre et al., 2016). Herein, the dependency parses of Brown are obtained in the same process as PTB and the dependency parsing standard used in PTB and Brown is the Stanford typed dependencies (the Stanford standard) (De Marneffe and Manning, 2008); while UD follows a different standard named the UD dependency parsing standard[7]. The statistics (i.e., the number of sentences, tokens, and

[4]https://catalog.ldc.upenn.edu/LDC99T42.

[5]https://stanfordnlp.github.io/CoreNLP.

[6]We use the version 2.9 of UD obtained from https://universaldependencies.org/

[7]https://universaldependencies.org/u/overview/syntax.html

Pre-training	Fine-tuning	Testing
Wiki ($\mathcal{SD}$)	PTB Training ($\mathcal{SD}$)	PTB Test ($\mathcal{SD}$)
		Brown ($\mathcal{SD}$)
	UD Training ($\mathcal{UD}$)	UD Test ($\mathcal{UD}$)

Table 2: The datasets used in pre-training, fine-tuning, and testing. The dependency standard used in the datasets are illustrated in parentheses with $\mathcal{SD}$ and $\mathcal{UD}$ referring to the Stanford dependency standard and the UD standard, respectively.

the averaged sentence length (ASL)) of all datasets, namely, PTB, Brown, UD, and English Wiki are reported in Table 1.

3.2 Obtaining the Auto-parsed Data

In the experiments, we propose to use existing NLP toolkits to obtain the auto-parsed Wiki data, because it not only allows us to benefit from existing tookits, but also is a good approximate of real-world applications where we want to build a good parser with existing toolkits. In addition, given PTB is one of the most widely used benchmark datasets for English dependency parsing, we want the auto-parsed data to follow exactly the same dependency standard as PTB, so that we can explore the effect of our approach when there is no gap between the standards in the auto-parsed and training data. However, many well-known existing dependency parsers (e.g., Stanford CoreNLP Toolkit (SCT) (Manning et al., 2014) and SpaCy[8]) follow a different standard.[9] Therefore, in the experiments, we employ a parsing-conversion process to obtain the dependency trees: we first use Berkeley Neural Parser[10] (Kitaev and Klein, 2018) to obtain the constituency trees of the Wiki text; then we convert them into dependency trees following the same process to obtain PTB (we denote this process as BNP-SD). Since the Berkeley Neural Parser is trained on the training set of PTB, this process ensures that the off-the-shelf dependency parser does not see the test data of PTB in training and the auto-parsed dependency trees follow the same dependency parsing standard as PTB.

[8]`https://spacy.io/`

[9]Specifically, the dependency parser of SCT is trained on UD and follows the UD dependency parsing standard; SpaCy is trained on OntoNotes 5 with the dependency trees converted from the corresponding constituency trees through ClearNLP, which does not exactly follow the Stanford standard.

[10]We use the model named "*benepar_en2*", which is downloaded from `https://github.com/nikitakit/self-attentive-parser`.

Hyper-parameters	Values
Learning Rate	5e-6, **1e-5**, 3e-5
Warmup Rate	**0.1**, 0.2
Dropout Rate	**0.33**
Batch Size	**16**, 32

Table 3: The hyper-parameters tested in tuning our models. The best ones used in our final experiments are highlighted in boldface.

3.3 Settings

Table 2 summarizes the datasets used in pre-training, fine-tuning, and testing, where the dependency parsing standards for them are also illustrated in parentheses ($\mathcal{SD}$ and $\mathcal{UD}$ stand for the Stanford standard and the UD dependency parsing standard, respectively). Herein, we denote the experiments using Brown and UD in testing as **cross-domain** and **cross-standard** experiments, respectively, because Brown (for testing) and PTB (for fine-tuning) come from different domains whereas UD (for testing) and Wiki (for pre-training) use different dependency standards. Intuitively, the cross-standard setting with UD as the test set is most challenging as the auto-parsed Wiki data used for pre-training and the UD data used for fine-tuning and testing follow different dependency standards.

In addition, since our system design requires the final parser to use a new randomly initialized decoder before fine-tuning, it is interesting to explore the impact of the choice of the final parser decoder. Therefore, in addition to our final parser with biaffine attentions (BF) following the architecture in Figure 1(b) (we denote the final parsers as "**+BF**"), we also try final parsers without BF and following the architecture in Figure 1(a) (we denote it as "**-BF**"). It is worth noting that, the architectures used in pre-training and fine-tuning are different under "+BF" whereas they are the same under "-BF" (the parser used in pre-training does not use BF). Intuitively, "+BF" setting is more challenging because the patterns learned by the encoder from pre-training may not fit into the new architecture (i.e., the BF module) in the final parser and thus result in noise in the final parser.

3.4 Implementation Details

Since pre-trained language models have achieved outstanding performance in many NLP tasks (Devlin et al., 2019; Wu et al., 2019; Yang et al., 2019; Raffel et al., 2019; Chen et al., 2020; Tian et al.,

Models	PTB		Brown		UD
	UAS	LAS	UAS	LAS	UAS
BERT-base	96.43	94.74	94.25	91.43	65.14
BERT-large	96.70	94.96	94.60	91.62	65.31
XLNet-base	96.81	95.02	94.74	91.70	65.57
XLNet-large	96.97	95.15	95.01	91.98	65.86

Table 4: The performance of different dependency parsers obtained after pre-training, **without** fine-tuning. We only report UAS (which does not evaluate the relation types associated with the dependency connections) on UD because UD and the auto-parsed data use different dependency parsing standard.

2020; Diao et al., 2020; Sun et al., 2020; Wang and Tu, 2020; Song et al., 2021; Diao et al., 2021), we use two of them, namely, BERT and XLNet, as the encoder for pre-training. Specifically, we use the base and large versions of them following the default settings: the base models use 12 layers of self-attentions with 768 dimensional hidden vectors and the large models use 24 layers of self-attentions with 1024 dimenstional hidden vectors.[11] We train the models on the auto-parsed Wiki for one epoch (i.e., 2,800K steps in total) with the batch size set to 32. It is worth noting that, since English Wiki is used to train BERT and XLNet, it could be considered that we do not use additional data in experiments. In fine-tuning, we use the parameters in the encoder obtained from pre-training to initialize the encoder in our approach and randomly initialize all other trainable parameters. Following previous studies, we evaluate all models based on unlabeled attachment score (UAS) and labeled attachment score (LAS). Table 3 reports the hyper-parameters tested in training our models. We test all combinations of them for each model and use the one achieving the highest LAS score in our final experiments.

4 Results, Analyses, and Findings

4.1 Overall Performance

Since pre-training also follows the process to train a dependency parser, we report the performance of the dependency parsers obtained after pre-training on the test set of PTB, Brown, and UD in Table 4 for reference. For the final parsers, we report the mean and standard deviation[12] of them on the test set of all datasets in Table 5.

Overall, results on PTB, Brown, and UD demonstrate the effectiveness of our approach under different configurations (i.e., using the base and large versions of BERT and XLNet encoders, with and without BF), where consistent improvements are observed in most cases, even though BERT and XLNet baselines have already achieved good performance. Particularly, it is promising to observe that our approach works well on UD (i.e., the cross-standard setting), where the pre-trained models has rather low performance (e.g., according to Table 4, BERT-large achieves 65.31% UAS on UD test set after pre-training) before they are fine-tuned on the gold standard. This observation demonstrates the effectiveness of our approach in cases where other approaches (e.g., training on the combination of the auto-parsed data and the gold training data) may not work well owing to the poor quality of auto-parsed data. Besides, our final parser with BF also works well in most cases with its architecture differing from the one used in pre-training. It shows the effectiveness and robustness of our approach to leverage the structure-aware encoder obtained from a parser with a different architecture, where the representation obtained from such encoder may not fit into the final parser due to the differences of the architectures in pre-training and fine-tuning.

In addition, we compare our best performing model[13] using BERT-large and XLNet-large and BF with previous studies on the PTB test set and report the results in Table 6. Overall, our approach outperforms all previous graph-based approaches (i.e., the ones without the "†" mark) except Mrini et al. (2020) that leverage auto-generated part-of-speech (POS) tags. Particularly, our model outperforms Zhou and Zhao (2019) and Zhou et al. (2020a) that perform constituency and dependency parsing at the same time through head-driven phrase structure grammar (HPSG) parsing. Besides, compared with Mrini et al. (2020) who proposed label attention layer to enhance the study of Zhou and Zhao (2019) on HPSG parsing, our approach obtain inferior performance because we do not use the label attention layer or the auto-generated POS tags. Given that our approach outperforms Zhou and Zhao (2019) on the test set of PTB, the effectiveness of our approach for dependency parsing is still valid and promising.

[11]We download the cased version of BERT from https://github.com/google-research/bert and XLNet from https://github.com/zihangdai/xlnet.

[12]In experiments, we run (fine-tune) each model five times with different random seeds.

[13]We follow previous studies to compare our best performing model with their models.

Models	PTB				Brown (cross-domain)				UD (cross-standard)			
	-BF		+BF		-BF		+BF		-BF		+BF	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
BERT-base	94.65	0.06	94.70	0.05	91.26	0.07	91.46	0.08	89.78	0.08	89.09	0.09
+ Dep. Wiki	**95.06**	0.05	**95.30**	0.05	**91.56**	0.07	**91.76**	0.06	**90.57**	0.07	**90.39**	0.08
BERT-large	95.01	0.07	95.11	0.06	91.79	0.08	91.84	0.07	90.70	0.09	90.80	0.08
+ Dep. Wiki	**95.25**	0.06	**95.50**	0.09	**92.00**	0.07	**92.40**	0.08	**91.22**	0.07	**91.01**	0.09
XLNet-base	95.13	0.05	95.19	0.06	91.78	0.06	91.98	0.07	91.02	0.07	**91.50**	0.07
+ Dep. Wiki	**95.49**	0.04	**95.50**	0.05	**92.35**	0.05	**92.38**	0.08	**91.51**	0.06	91.39	0.07
XLNet-large	95.48	0.04	95.54	0.06	92.31	0.05	92.53	0.07	91.38	0.08	**92.30**	0.07
+ Dep. Wiki	**95.71**	0.05	**95.86**	0.05	**92.45**	0.06	**92.54**	0.07	**91.94**	0.07	91.70	0.08

Table 5: The mean μ and standard deviation σ of LAS of our approaches (**with** the fine-tuning of the structure-aware encoder) and the baseline models with different configurations (i.e., the ones using base or large BERT/XLNet with (+BF) and without (-BF) bi-affine attentions) on the test set of PTB, Brown, and UD.

Models	UAS	LAS
Dozat and Manning (2017)	95.74	94.08
*Dozat and Manning (2017) (BERT)	96.64	95.11
*Zhou and Zhao (2019) (BERT)	97.00	95.43
*Zhou and Zhao (2019) (XLNet)	97.20	95.72
*Zhou et al. (2020a) (XLNet)	97.23	95.65
*Zhou et al. (2020b) (LIMIT-BERT)	97.14	95.44
*Mrini et al. (2020) (XLNet + POS)	**97.42**	**96.26**
*Wang and Tu (2020) (BERT)	96.91	95.34
Zhang et al. (2021) (BERT)	96.64	95.09
Mohammadshahi and Henderson (2021) (BERT)	96.66	95.01
*†Fernández-González and Gómez-Rodríguez (2021) (BERT)	97.05	95.47
*†Yang and Tu (2021) (BERT)	97.24	95.73
BNP-SD (Kitaev and Klein, 2018)	96.03	94.03
*Ours (BERT-large)	97.06	95.60
*Ours (XLNet-large)	97.30	95.92

Table 6: Comparison (UAS and LAS) of our approach with previous studies. "*" denotes the models using the large version of BERT and XLNet; "†" marks the parsers that do not use the graph-based approaches.

4.2 The Effect of System Design

Our parser is different from many of the previous studies in two ways: (1) the auto-parsed data is used for pre-training only, (2) the fine-tuning step uses a different decoder from the one used in pre-training, whose weights are initialized randomly.

To determine the impact of those decisions, we build three more parsers for comparison. The first one uses the architecture in Figure 1(a) and is trained with the union of the auto-parsed and gold standard data (we denote this approach as "Union") without the fine-tuning step. The second parser ("Fine-tuning") is pre-trained with the auto-parsed data and fine-tuned with gold dependency trees, but the two stages use the same decoder (as in Figure 1(a)) and the decoder's weights for fine-tuning are initialized with the weights from pre-training. The third one ("Randomize") is the same as the second one but the weights of the decoder derived from pre-training are thrown away before fine-tuning. The "Randomize" system differs from our final parser only in that our final parser uses a different decoder in the fine-tuning stage.

All aforementioned three approaches use BERT-base encoder. For auto-parsed data, we randomly select sentences from English Wiki where the number of selected sentences equals to the number of sentences in the training set of different datasets (i.e., 40K auto-parsed sentences for PTB and Brown, and 13K auto-parsed sentences for UD).[14] For each approach, we run it five times with different random data and report the average results (LAS for PTB and Brown, and UAS[15] for UD) of them, as well as the average results of BERT-base baseline and our final parser, in Table 7.

It is observed that "Randomize" consistently outperforms the other two approaches on the test set of all datasets. Particularly, for cross-standard settings on UD, because the auto-parsed data and the UD data use different dependency standards, the quality of the auto-parsed data can be considered relatively low with respect to the UD standard (this can be confirmed by the low model performance of the parser pre-trained on auto-parsed data on the test set of UD (see Table 4)). Under this setting, the "Union" and "Fine-tuning" even achieve inferior results (the results are underlined) compared with the BERT-base baseline, because they suffer from the gap between the dependency standards of auto-parsed data and gold data. On the contrary,

[14] We also tested other numbers of selected auto-parsed sentences and obtain similar observations.

[15] We report UAS (where the dependency type are ignored in evaluation) for UD because the auto-parsed data and the UD use different dependency standards.

Approach	PTB	Brown	UD
BERT-base	94.65	91.26	92.86
+ Union	94.69	91.30	<u>83.51</u>
+ Fine-tuning	94.76	91.38	<u>92.14</u>
+ Randomize	**94.94**	**91.47**	**93.10**
Our Final Parser	95.19	91.60	92.98

Table 7: The average performance (LAS for PTB and Brown, UAS for UD) of different approaches using BERT-base encoder with the scores lower than the baseline highlighted by underlines. "Union" refers to the model that is trained on the union of auto-parsed and gold data; "Fine-tuning" denotes the model that is pre-trained on auto-parsed data and then further fine-tuned on the gold data; "Randomize" is the same as "Fine-tuning" except that the weights of the decoder is randomly initialized before fine-tuning. The best scores among "Union", "Fine-tuning", and "Randomize" are highlighted by boldface. All three models and the baseline use the architecture in Figure 1(a) and the only difference between "Randomize" and our final parser is the architecture of decoder in fine-tuning.

"Randomize" is able to carefully address the noise in the auto-parsed data by using the auto-parsed and the gold standard data in different stages and getting rid of the pre-trained decoder (which may learn the noise) and randomly initializing a new one before fine-tuning.

4.3 The Size of Auto-parsed Data

An essential question for evaluating our approach is that how many data (auto-parsed with noise) are required to improve the parsers with various model sizes. To answer its question, we pre-train four models with variant sizes, i.e., 6-layer BERT-base (55M parameters), 12-layer BERT-base (110M parameters), 18-layer BERT-large (252M parameters), and 24-layer BERT-large (336M parameters), on different amount of randomly selected auto-parsed Wiki data. Figure 2 illustrates the averaged[16] improvement (LAS) of four different models over their corresponding baselines on three test sets: (a) PTB, (b) Brown, and (c) UD, with respect to the ratio of the pre-training sentence number to the model size. It is interesting that, for the in-domain setting (i.e., when both training and test sets are from the PTB), the zero points of different curves for all models are roughly the same, i.e., around 0.3, which means that the pre-training only needs a little bit over 300 sentences for every 50M parameters in a parser to ensure an improvement (e.g., for 24-layer BERT-large, it only requires two thousand auto-parsed sentences to obtain a better-than-original parser).

[16]We run the experiment for each model ten times with different random data to guarantee the results are trustworthy.

Data Auto-parsed by	PTB	Brown	UD
N/A	94.78	91.56	90.10
BNP-SD (94.03)	95.37	91.83	90.48
Parser I (94.78)	94.83	91.58	90.42
Parser II (95.26)	95.38	91.84	91.70

Table 8: Performance (LAS) comparison of a model (using BERT-base with BF) pre-trained on auto-parsed English Wiki from different parsers. N/A refers to no pre-training, Parser I and Parser II are the parsers (with BF) trained on PTB using BERT-base and XLNet-base, respectively. The LAS from all parsers on the PTB test set are reported in parentheses for reference.

This finding is highly encouraging since it only needs a small amount of auto-parsed data (compared to 92M sentences in English Wiki) to improve a large model. Similar observations can be drawn for cross-domain and cross-standard settings on Brown and UD datasets, where more (since there are gaps between the pre-training and the fine-tuning data) but still limited auto-parsed data is required to ensure that improvement. Particularly, for each dataset, we found there exists a rather stable ratio for different models, e.g., 0.3 for PTB, 0.6 for Brown, etc., which is a meaningful guidance to improve parsers' performance regarding to their parameters. An explanation to this observation is that structural data is useful to update representation models (Gubbins and Vlachos, 2013; Levy and Goldberg, 2014; Zhou et al., 2020b) so that a limited amount (w.r.t. model size) could greatly affect model performance especially when they are applied on structure-prediction tasks such as parsing.

4.4 The Choice of Existing Parsers

Another factor that may affect the performance of our systems is the off-the-shelf parser used to produce auto-parsed data. To assess the effect of the parser on the performance of the final systems, we experimented with two more parsers from our baselines, i.e., the ones using BERT-base (**Parser I**) and XLNet-base (**Parser II**) trained on PTB, in addition to BNP-SD as described in Section 3.2: Table 8 reports the LAS of models with or without pre-training. While pre-training improves the performance with auto-parsed data from all four

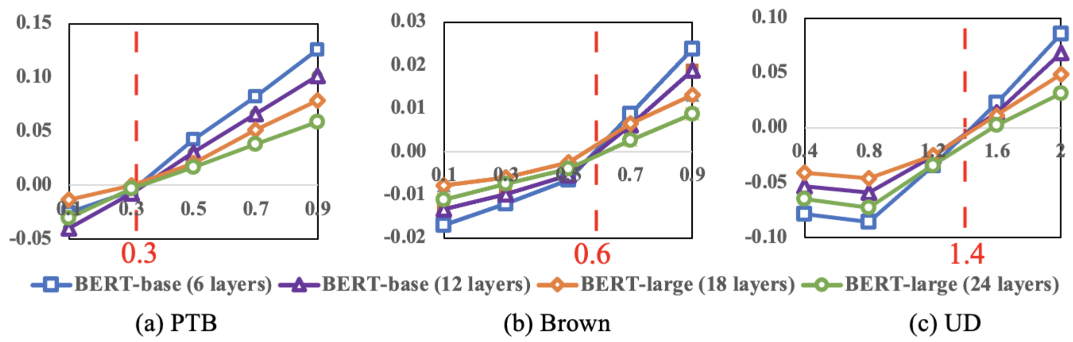

Figure 2: The improvement (LAS) of four models over their baselines on three test sets. The X-axis is the number of auto-parsed sentences used for pre-training divided by the number of model parameters, and then multiplied by 50,000 (to make the scale more readable).

Models	PTB	Brown	UD
BERT-base	94.70	91.46	89.09
+ Dep. Wiki (-BF)	95.30	91.76	90.39
+ Dep. Wiki (+BF)	**95.35**	**91.80**	**90.43**
XLNet-base	95.19	91.98	91.50
+ Dep. Wiki (-BF)	95.50	92.38	91.39
+ Dep. Wiki (+BF)	**95.54**	**92.40**	**91.97**

Table 9: The average LAS of final parsers (with BF) using BERT-base and XLNet-base encoders, with (+) and without (-) using BF in pre-training.

parsers, the improvement with Parser I is less significant than the other parsers because it is more similar to the final parser in terms of the model architecture.

This finding is quite intuitive and matches the observation in Surdeanu and Manning (2010) that the diversity between different parsers plays an important role in improving model performance when combining them. Specifically, when being evaluated on PTB and Brown, Parser I tends to make the same mistakes as our final parser since they use exactly the same architecture (i.e., Transformer), which reduces the benefits of pre-training with auto-parsed data. In contrast, BNP-SD (which uses ELMo embeddings (Peters et al., 2018)) and parser II (which is based on Transformer-XL (Dai et al., 2019)) use different resources and architectures so that our approach with BERT-base encoder can learn from their auto-parsed data. In addition, when our final parser is evaluated on UD, since UD uses a different dependency standard, Parser I and our final parser no longer make the "same" mistakes, which results in more improvement on UD.

4.5 The Decoder Used in Pre-training

In the main experiments, we pre-train the parser without using BF. To explore its effect, we conduct an ablation study where BF is used in pre-training. Table 9 reports the average LAS of different final parsers (with BF) using BERT-base and XLNet-base encoders, where BF is used (i.e., "+BF") or not used (i.e., "-BF") in pre-training. It is observed that using BF in pre-training results in similar performance compared with the settings where BF is not used. It demonstrates the robustness of our approach where the architecture of the final parser (with BF) does not need necessary to be identical to the one (without BF) in pre-training to obtain promising improvement over the baselines. In addition, it is worth-noting that for experiments on UD with XLNet-base, "+ Dep. Wiki (+BF)" outperforms the baseline model whereas "+ Dep. Wiki (-BF)" fails to do so. The explanation could be the following. For "+ Dep. Wiki (+BF)", the only gap between pre-training and fine-tuning is the dependency standard, whereas "+ Dep. Wiki (-BF)" faces an additional gap that the architectures used in pre-training and fine-tuning are different. Therefore, "+ Dep. Wiki (-BF)" fails to overcome the two gaps and thus results in inferior results compared with the XLNet baseline. On the other hand, when BF is also used in pre-training, the gap between the architectures does not exist, which allows our final parser to obtain a higher performance.

5 Related Work

Recent studies for dependency parsing use advanced architectures (e.g., bi-LSTM, BERT) to capture contextual information so as to achieve outstanding performance (Shen et al., 2021; Zhang et al., 2021; Yang and Tu, 2021; Li et al., 2021). To further improve dependency parsing, approaches such as bi-affine attentions (Dozat and Manning, 2017; Attardi et al., 2021; Xu and Koehn, 2021), HPSG parsing (Zhou and Zhao, 2019; Zhou et al., 2020a; Mrini et al., 2020), TreeCRF (Zhang et al., 2020) are further applied. Besides, to improve model performance, there are studies that use existing dependency parsers and auto-parsed data through model ensemble (Attardi and Dell'Orletta, 2009; Surdeanu and Manning, 2010; Che et al., 2018) or semi-supervised approaches (Sagae and Lavie, 2006; Chen et al., 2009; Prokopidis and Papageorgiou, 2014; Yu and Bohnet, 2017; Zhang et al., 2021; Wagner and Foster, 2021).

Compared to previous studies that use auto-parsed data, our approach differs in several ways. First, our encoder is structure-aware as it is pre-trained with dependency trees. Second, because auto-parsed data is noisy and may use dependency standard different from that of the test data (in the cross-standard setting), it is used in pre-training only. In contrast, training a parser on the union of auto-parsed data and gold data would not work well, especially in the cross-standard setting, as shown in Table 7. Third, the decoder of the fine-tuning stage starts with randomly initialized weights, instead of with the weights learned from the pre-training stage, thus ensuring that the decoder in the final parser will not be affected by the noisy auto-parsed data.

6 Conclusion

In this study, we propose a simple and effective solution to improve dependency parser through pre-training on auto-parsed data. In doing so, the encoder is able to learn structural information from the auto-parsed data in pre-training. During fine-tuning, a different decoder is used and its weights initialized randomly, thus reducing the impact of errors in the auto-parsed data. We have run a large number of experiments under different settings (e.g., cross-domain vs. cross-standard, -BF vs. +BF, different parsers used to parse Wiki) and shown that our approach outperforms strong baselines and many previous studies under those settings. Furthermore, pre-training needs only a small amount of auto-parsed data (e.g., 2K sentences for a BERT-large based parser on the PTB test set) to ensure improvement over strong baselines.

References

Giuseppe Attardi and Felice Dell'Orletta. 2009. Reverse Revision and Linear Tree Combination for Dependency Parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 261–264, Boulder, Colorado.

Giuseppe Attardi, Daniele Sartiano, and Maria Simi. 2021. Biaffine Dependency and Semantic Graph Parsing for Enhanced Universal Dependencies. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 184–188, Online.

Mohit Bansal and Dan Klein. 2011. Web-scale Features for Full-scale Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 693–702, Portland, Oregon, USA.

Dongfeng Cai, Yonghua Hu, Xuelei Miao, and Yan Song. 2009. Dependency Grammar Based English Subject-Verb Agreement Evaluation. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, pages 63–71, Hong Kong.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards Better UD Parsing: Deep Contextualized Word Embeddings, Ensemble, and Treebank Concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium.

Dong-Dong Chen, Wei Wang, Wei Gao, and Zhi-Hua Zhou. 2018. Tri-net for Semi-Supervised Deep Learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2014–2020.

Guimin Chen, Yuanhe Tian, and Yan Song. 2020. Joint Aspect Extraction and Sentiment Analysis with Directional Graph Convolutional Networks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 272–279.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving Dependency Parsing with Subtrees from Auto-Parsed Data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 570–579, Singapore.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford Typed Dependencies Manual. Technical report, Technical report, Stanford University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Shizhe Diao, Jiaxin Bai, Yan Song, Tong Zhang, and Yonggang Wang. 2020. ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4729–4740.

Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. 2021. Taming Pre-trained Language Models with N-gram Representations for Low-Resource Domain Adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3336–3349, Online.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jason M. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021. Dependency Parsing with Bottom-up Hierarchical Pointer Networks. *arXiv preprint arXiv:2105.09611*.

Joseph Gubbins and Andreas Vlachos. 2013. Dependency Language Models for Sentence Completion. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1405–1410, Seattle, Washington, USA.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention Guided Graph Convolutional Networks for Relation Extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251.

Binxuan Huang and Kathleen M Carley. 2019. Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5472–5480.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A Deep Ensemble Model with Slot Alignment for Sequence-to-sequence Natural Language Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162, New Orleans, Louisiana.

Eliyahu Kiperwasser and Yoav Goldberg. 2015. Semi-supervised Dependency Parsing using Bilexical Contextual Features from Auto-Parsed Data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1348–1353, Lisbon, Portugal.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia.

Hayato Kobayashi. 2018. Frustratingly Easy Model Ensemble for Abstractive Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4165–4176, Brussels, Belgium.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio.

Ryosuke Kuwabara, Jun Suzuki, and Hideki Nakayama. 2020. Single Model Ensemble using Pseudo-Tags and Distinct Vectors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3006–3013, Online.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland.

Ying Li, Meishan Zhang, Zhenghua Li, Min Zhang, Zhefeng Wang, Baoxing Huai, and Nicholas Jing Yuan. 2021. APGN: Adversarial and Parameter Generation Networks for Multi-Source Cross-Domain Dependency Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1724–1733, Punta Cana, Dominican Republic.

Xuezhe Ma and Fei Xia. 2013. Dependency Parser Adaptation with Subtrees from Auto-Parsed Target Domain Data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 585–590, Sofia, Bulgaria.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language

Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Alireza Mohammadshahi and James Henderson. 2021. Recursive Non-Autoregressive Graph-to-Graph Transformer for Dependency Parsing with Iterative Refinement. *Transactions of the Association for Computational Linguistics*, 9:120–138.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking Self-Attention: Towards Interpretability in Neural Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving Named Entity Recognition with Attentive Ensemble of Syntactic Information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4231–4245.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Prokopis Prokopidis and Haris Papageorgiou. 2014. Experiments for Dependency Parsing of Greek. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 90–96, Dublin, Ireland.

Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving Arabic Diacritization with Regularized Decoding and Adversarial Training. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-text Transformer. *arXiv preprint arXiv:1910.10683*.

Guy Rotman and Roi Reichart. 2019. Deep Contextualized Self-training for Low Resource Dependency Parsing. *Transactions of the Association for Computational Linguistics*, 7:695–713.

Piotr Rybak and Alina Wróblewska. 2018. Semi-Supervised Neural System for Tagging, Parsing and Lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 45–54, Brussels, Belgium.

Kenji Sagae and Alon Lavie. 2006. Parser Combination by Reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. StructFormer: Joint Unsupervised Induction of Dependency and Constituency Structure from Masked Language Modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7196–7209, Online.

David A. Smith and Jason Eisner. 2007. Bootstrapping Feature-rich Dependency Parsers with Entropic Priors. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 667–677, Prague, Czech Republic.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised Dependency Parsing using Generalized Tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1065–1073, Beijing, China.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*.

Kathrin Spreyer and Jonas Kuhn. 2009. Data-driven Dependency Parsing of New Languages Using Incomplete and Noisy Training Data. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 12–20, Boulder, Colorado.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble Models for Dependency Parsing: Cheap and Good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652, Los Angeles, California.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020. Improving Biomedical Named Entity Recognition with Syntactic Information. *BMC Bioinformatics*, 21:1471–2105.

Yuanhe Tian, Yan Song, and Fei Xia. 2022. Improving Relation Extraction through Syntax-induced Pre-training with Dependency Masking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

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

Joachim Wagner and Jennifer Foster. 2021. Revisiting Tri-training of Dependency Parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9457–9473, Online and Punta Cana, Dominican Republic.

Xinyu Wang and Kewei Tu. 2020. Second-Order Neural Dependency Parsing with Message Passing and End-to-End Training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99, Suzhou, China.

Zhaofeng Wu, Yan Song, Sicong Huang, Yuanhe Tian, and Fei Xia. 2019. WTMED at MEDIQA 2019: A Hybrid Approach to Biomedical Natural Language Inference. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 415–426, Florence, Italy.

Haoran Xu and Philipp Koehn. 2021. Zero-Shot Cross-Lingual Dependency Parsing through Contextual Embedding Transformation. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 204–213, Kyiv, Ukraine.

Songlin Yang and Kewei Tu. 2021. Headed Span-based Projective Dependency Parsing. *arXiv preprint arXiv:2108.04750*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763.

Juntao Yu and Bernd Bohnet. 2017. Dependency Language Models for Transition-based Dependency Parsing. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 11–17, Pisa, Italy.

Hongming Zhang, Yan Song, Yangqiu Song, and Dong Yu. 2019. Knowledge-aware Pronoun Coreference Resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 867–876, Florence, Italy.

Xudong Zhang, Joseph Le Roux, and Thierry Charnois. 2021. Strength in Numbers: Averaging and Clustering Effects in Mixture of Experts for Graph-Based Dependency Parsing. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 106–118, Online.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient Second-Order TreeCRF for Neural Dependency Parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online.

Junru Zhou, Zuchao Li, and Hai Zhao. 2020a. Parsing All: Syntax and Semantics, Dependencies and Spans. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4438–4449, Online.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020b. LIMIT-BERT : Linguistics Informed Multi-Task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4450–4461, Online.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar Parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy.