Dependency Parsing via Sequence Generation

Boda Lin[1‡], **Zijun Yao**[2,3‡], **Jiaxin Shi**[4], **Shulin Cao**[2,3]
Binghao Tang[1], **Si Li**[1*], **Yong Luo**[5], **Juanzi Li**[2,3], **Lei Hou**[2,3]
[1]School of Artificial Intelligence, Beijing University of Posts and Telecommunications
[2]Department of Computer Science and Technology, BNRist;
[3]KIRC, Institute for Artificial Intelligence Tsinghua University, Beijing 100084, China
[4]Huawei Cloud Computing Technologies [5]School of Computer Science, Wuhan University
{linboda, lisi}@bupt.edu.cn
{yaozj20@mails., houlei@}tsinghua.edu.cn

Abstract

Dependency parsing aims to extract syntactic dependency structure or semantic dependency structure for sentences. Existing methods for dependency parsing include transition-based method, graph-based method and sequence-to-sequence method. These methods obtain excellent performance and we notice them belong to labeling method. Therefore, it may be very valuable and interesting to explore the possibility of using generative method to implement dependency parsing. In this paper, we propose to achieve Dependency Parsing (DP) via Sequence Generation (SG) by utilizing only the pre-trained language model without any auxiliary structures. We first explore different serialization designing strategies for converting parsing structures into sequences. Then we design *dependency units* and concatenate these units into the sequence for DPSG. We verify the DPSG is capable of parsing on widely used DP benchmarks, i.e., PTB, UD2.2, SDP15 and SemEval16. In addition, we also investigate the astonishing low-resource applicability of DPSG, which includes unsupervised cross-domain conducted on CODT and few-shot cross-task conducted on SDP15. Our research demonstrates that sequence generation is one of the effective methods to achieve dependency parsing. Our codes are available now.[1].

1 Introduction

Dependency Parsing (DP), which aims to extract the structural information beneath sentences, is fundamental in understanding natural languages. It benefits a wide range of Natural Language Processing (NLP) applications, such as machine translation (Bugliarello and Okazaki, 2020), question answering (Teney et al., 2017), and information retrieval (Chandurkar and Bansal, 2017). As shown in Figure 1, dependency parsing predicts for each word the existence and dependency relation with other words according to a pre-defined formation. Such dependency structure is represented in tree or directed acyclic graph, which can be converted into flattened sequence, as presented in this paper.

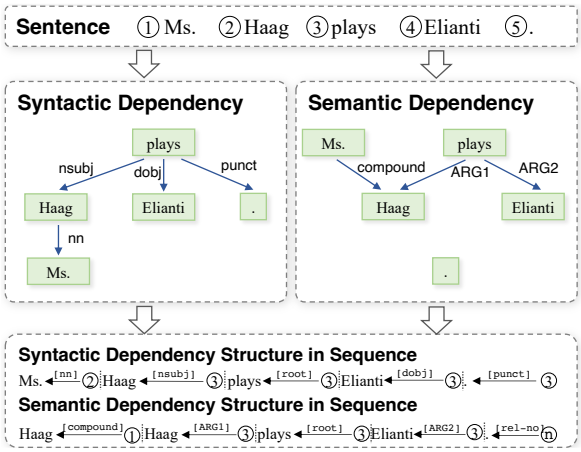

Figure 1: Parsing "*Ms. Haag plays Elianti .*" according to the Stanford syntactic dependency structure (Manning et al., 2014) and the DM semantic dependency structure (Oepen et al., 2014). They are further converted into unified serialized representations. We use the position ID to replace the head word in flattened sequence.

Previous models for dependency parsing mostly predict the labeling of each node in the parsing tree and parsing graph, which include graph-based methods (Dozat and Manning, 2017), transition-based methods (Ma et al., 2018), and sequence-to-sequence methods (Li et al., 2018). While prospering with these methods, dependency parsing shows three trends now. 1) New Representation. Recent works extend dependency parsing from syntactic DP (SyDP) to semantic DP (SeDP) with many new representations (Oepen

[1]https://github.com/TimeLessLing/DPSG-code/tree/main
‡Boda Lin and Zijun Yao make equal contribution
*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 7339–7353
December 7-11, 2022

et al., 2014; Che et al., 2012). 2) Low-resource. Corpora from different domains facilitate the research on cross-domain dependency parsing (Peng et al., 2019; Li et al., 2019). 3) PLM. With the development of pre-trained language models (PLMs), researchers manage to enable PLMs on dependency task and achieve the new state-of-the-art (SOTA) results (Fernández-González and Gómez-Rodríguez, 2020; Gan et al., 2021). However, PLMs used in most previous research are encoder-only PLM such as BERT (Devlin et al., 2019). Recently, more and more research demonstrates the power of generative encoder-decoder PLM (Du et al., 2021), which inspires us to think about an interesting question: can dependency parsing be implemented in a generative sequence-to-sequence method? If this method works, can it be used well for new representations and low-resource scenarios?

In order to verify the feasibility of generative-parsing and explore suitable design solutions, we propose **D**ependency **P**arsing via **S**equence **G**eneration (DPSG). The core idea is to find a unified unambiguous serialized representation for both syntactic and semantic dependency structures. Then an encoder-decoder PLM is learned to generate the parsing results following the serialized representation, without the need for an additional decoder. That is, our parser can achieve its function using one original PLM (without any modification).

In particular, DPSG consists of three key components. The *Serializer* is responsible for converting between the dependency structure and the serialized representation. The *Positional Prompt* pattern provides supplementary word position information in the input sentence to facilitate the sequence generation process. The *encoder-decoder PLM* with added special tokens performs the parsing task via sequence generation.

We conduct experiments on 5 popular DP benchmarks: PTB, UD2.2, CODT, SDP15, and SemEval16. DPSG performs generally well on different DP tasks. It significantly outperforms the baselines on cross-domain (CODT) and Chinese SeDP (SemEval16) corpora, and achieves comparable results on the other three benchmarks. In addition, the few-shot cross-task experiments also demonstrate the applicability of DPSG . We also design a series of pilot experiments to explore the rationality of the designing of the DPSG. Our research shows that generative sequence-to-sequence method has the potential to be an effective sequence-to-sequence approach for dependency parsing.

2 Preliminaries

We formally introduce the dependency parsing task and the encoder-decoder PLM, and the corresponding notations. This paper uses bold lower case letters, blackboard letters, and bold upper case letters to denote sequences, sets, and functions, respectively. Elements in the sequence and the sets are enclosed in parentheses and braces, respectively.

2.1 Dependency Parsing

A pre-defined dependency representation is a set of relations $\mathbb{R}$. Dependency parsing takes a sentence $\mathbf{x} = (w_1, w_2, ..., w_n)$ as input, where w_i is the i^{th} word in the sentence. It outputs the set of dependency pairs $\mathbf{y} = (p_1, p_2, ..., p_n)$, where $p_i = \left\{\left(r_i^j, h_i^j\right)\right\}$ denotes the dependency pair of the i^{th} word w_i. We use h_i^j and r_i^j to denote the j^{th} head word of w_i and their relation. $\texttt{PID}(w)$ denotes the position of the specific word w in the input sentence.

Syntactic Dependency Parsing (SyDP) analyses the grammatical dependency relations. The parsing result of SyDP is a tree structure called the syntactic parsing tree. In the SyDP, each non-root word has exactly one head word, which means $|p_i| = 1$ if w_i is the not root word.

Semantic Dependency Parsing (SeDP) focuses on representing the deep-semantic relation between words. Each word in SeDP is allowed to have *multiple* (even no) head words. This leads to the result of SeDP being a directed acyclic graph called Semantic Dependency Graph. Figure 1 shows the difference between SyDP and SeDP, where SyDP produces a tree while SeDP produces a graph.

2.2 Pre-trained Language Model

PLMs are usually stacks of attention blocks of Transformer (Vaswani et al., 2017). Some PLMs that consist of encoder blocks only (e.g., BERT (Devlin et al., 2019)) are not capable of sequence generation. This paper focuses on PLMs having both encoder blocks and decoder blocks, such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2020).

An encoder-decoder PLM takes a sequence $\mathbf{s} = (s_1, ..., s_n)$ as input, and outputs a sequence $\mathbf{PLM}(\mathbf{s}) = \mathbf{o} = (o_1, ..., o_m)$. Each PLM has an associated vocabulary $\mathbb{V}$, which is a set of tokens that can be directly accepted and embedded by the

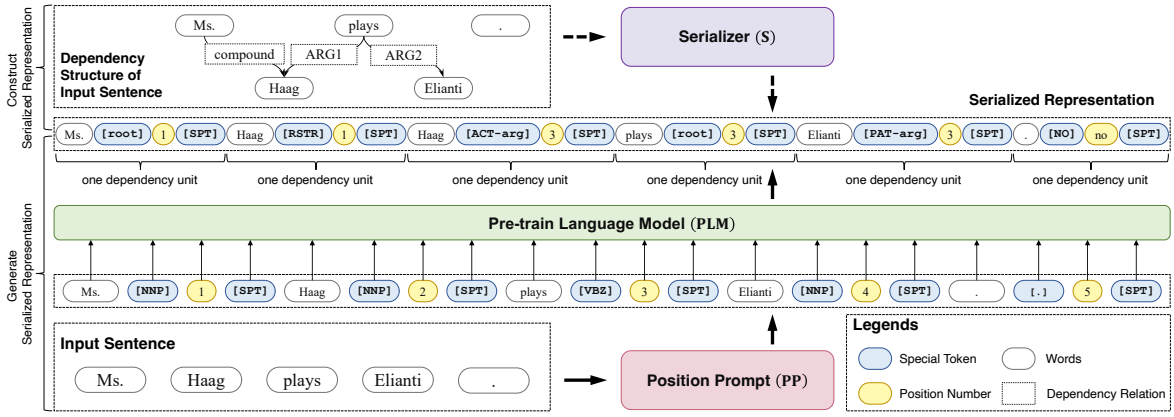

Figure 2: This figure shows the overall framework of DPSG. The DM semantic dependency structure of "*Ms. Haag plays Elianti .*" is converted into the serialized representation by the Serializer. The Positional Prompt module injects positional information into the input sentence, and the PLM is responsible for generating the results.

PLM. The PLM first splits the input sequence into tokens in the vocabulary with a subword tokenization algorithm, such as SentencePieces (Kudo and Richardson, 2018). Then, the tokens are mapped into vectors by looking up the embedding table. The attention blocks digest the embedded sequence and generate the output sequence.

3 Method

DPSG leverages a PLM to parse the dependency relation of a sentence by sequence generation. Therefore, the *Serializer* converts the dependency structure into a serialized representation that meets the output format of the PLM (Section 3.1). The *Positional Prompt* injects word position and part-of-speech tagging into the input sentence so as to avoid numerical reasoning (Section 3.2). The PLM is modified by adding special tokens introduced by the Serializer and the Positional Prompt (Section 3.3). Figure 2 illustrates the overall framework.

3.1 Serializer for Dependency Structure

The Serializer $\mathbf{S} : (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{t}$ is a function that maps sentence $\mathbf{x}$ and its corresponding dependency pairs $\mathbf{y}$ into a serialized representation $\mathbf{t}$, which servers as the target output to fine-tune the language model. The Inverse Serializer $\mathbf{S}^{-1} : (\mathbf{x}, \mathbf{o}) \mapsto \mathbf{y}$ converts the output $\mathbf{o}$ of the PLM into dependency pairs to meet the output requirement of the DP task.

Specifically, the Serializer $\mathbf{S}$ decomposes dependency pairs, $\left\{\left(h_i^j, r_i^j\right)\right\} \in \mathbf{y}$, into smaller dependency units by scattering the dependent word w_i into each of its head word, which forms the following triplets set: $\left\{(w_i, r_i^j, h_i^j)\right\}$. Then, it replaces each relation r_i^j with a special token[2] $\left[\texttt{REL}\left(r_i^j\right)\right] \in \mathbb{R}$, where $\mathbb{R}$ is a set of special tokens for all different relations. The head word h_i^j is substituted by its position in the input sentence $\mathbf{x}$, denoted as $\texttt{PID}\left(h_i^j\right)$. The target serialized representation $\mathbf{t} = \mathbf{S}(\mathbf{x})$ concatenates all the dependency units with split token $[\texttt{SPT}]$ as the following:

$$\Big(\ldots\Big[\texttt{SPT}\Big] \underbrace{w_i \Big[\texttt{REL}\left(r_i^j\right)\Big] \texttt{PID}\left(h_i^j\right)}_{\text{one dependency unit}} \Big[\texttt{SPT}\Big]\ldots\Big)$$

The Inverse Serializer $\mathbf{S}^{-1}$ restores the dependency structure from the serialized representation by substituting the special token $\left[\texttt{REL}\left(r_i^j\right)\right]$ with the original relation and indexing the head with its position $\texttt{PID}\left(h_i^j\right)$ in the input sentence $\mathbf{x}$.

There are two issues in the Serializer designing: **Word Ambiguity.** It is highly possible for words, especially function words, to appear multiple times in one sentence, e.g., there are more than 72% sentences in Penn Treebank (Marcus et al., 1993) with repeated words. We take two measures for word disambiguation in a dependency unit: (1) To disambiguate head word, the Serializer represents the head word by its position, rather than the word itself; (2) To disambiguate dependent word, the Serializer arranges dependency units by order of the dependent word in the input sentence $\mathbf{x}$, rather than topological ordering or depth/breadth first search ordering of the dependency graph. The Inverse Se-

[2]Brackets indicate special tokens out of vocabulary $\mathbb{V}$.

rializer scans $\mathbf{x}$ and $\mathbf{o}$ simultaneously so as to refer the corresponding dependent word to $\mathbf{x}$.

Isolated Words. There are dependency representations allowing for isolated words which have neither head words nor dependency relations with other words, e.g., the period mark in the SeDP results shown in Figure 1. Note that the isolated words are different from the root word, as the root word is the head word of itself. We use special token [NO] to denote such isolation relation and word no to represent the position of the virtual head word.

3.2 Positional Prompt for Input Sentence

As Section 3.1 mentions, representing the head words by their positions is an important scheme for head word disambiguation. However, PLMs are less skilled at numerical reasoning (Geva et al., 2020). We also empirically find it difficult for the PLM to learn the positional information of each word from scratch. Thus, we inject Positional Prompt (PP) for each word, which converts the positional encoding problem into generating the position number in the input, rather than counting for each word.

In particular, given the input sentence $\mathbf{x}$, the positional prompt is the position number of each word w_i wrapped with two kinds of special tokens [$\texttt{POS}_\texttt{i}$] and [SPT]. [$\texttt{POS}_\texttt{i}$] marks the part-of-speech tagging (POS tagging) of w_i and prevents the tokenization algorithms from falsely taking the positional prompt as part of the previous word. [SPT] separates the position number from the next word. They also provide word segmentation information for some languages, such as Chinese. After the conversion, we have the input sequence in the following form:

$$\mathbf{s} = w_1\ [\texttt{POS}_1]\ 1\ [\texttt{SPT}]\ w_2\ [\texttt{POS}_2]\ 2\ [\texttt{SPT}] \cdots$$

For brevity, we denote the above process as a function $\mathbf{PP} : \mathbf{x} \mapsto \mathbf{s}$ that maps input sentence into sequence with positional prompt.

3.3 PLM for Sequence Generation

Both Serializer and Positional Prompt introduce special tokens that are out of the original vocabulary $\mathbb{V}$, including the relation tokens in $\mathbb{R}$, the separation tokens [PID], [SPT], and the special relation token [NO]. Before training, these tokens are added to the vocabulary, and their corresponding embeddings are randomly initialized from the same distribution as other tokens. As we should notice, these special tokens are expected to undertake different semantic information. PLM thus treats them as trainable variables and learns their semantic information during training.

With all the three components of DPSG, input sentence is first converted into sequence with positional prompt: $\mathbf{s} = \mathbf{PP}(\mathbf{x})$. The sequence is further fed into the PLM and get the sequence output with the maximum probability: $\mathbf{o} = \mathbf{PLM}(\mathbf{s})$. The final predicted dependency structure is recovered via the Inverse Serializer: $\mathbf{y}' = \mathbf{S}^{-1}(\mathbf{o})$.

The training objective aims to maximize the likelihood of the ground truth dependency structure. To do so, we take the serialized dependency structure as the target and minimize the auto-regressive language model loss.

4 Experiments

4.1 Evaluation Setups

4.1.1 Datasets

We evaluate DPSG on the following 5 widely used benchmarks for both SyDP and SeDP. We show more details about datasets in Appendix A.

- **Penn Treebank** (PTB) (Marcus et al., 1993) is the most well-known benchmark for SyDP. We follow Ma et al. (2018) to use the Stanford basic Dependencies representation (de Marneffe et al., 2006) of PTB convert by Stanford parser[3].
- **Universal Dependency Treebanks** (UD) (Nivre et al., 2016) is the most popular multi-lingual dataset for SyDP. We follow previous works (Gan et al., 2021; Ma et al., 2018) to process 12 languages from 2.2 version of UD (UD2.2). They are: Bulgarian (bg), Catalan (ca), Czech (cs), German (de), English (en), Spanish (es), French (fr), Italian (it), Dutch (nl), Norwegian (no), Romanian (ro), and Russian (ru).
- **Chinese Open Dependency Treebank** (CODT) (Li et al., 2019) aims to evaluate the cross-domain SyDP capacity of the parser. It includes a balanced corpus (BC) for training, and three other corpora gathering from different domains for testing: product blogs (PB), popular novel "Zhu Xian" (ZX), and product comments (PC).
- **BroadCoverage Semantic Dependency Parsing** dataset (SDP15) (Oepen et al., 2014) annotates English SeDP sentences with three different

[3]http://nlp.stanford.edu/software/lex-parser.html

representations, named as DM, PAS, and PSD. It provides both in-domain (ID) and out-of-domain (OOD) evaluation datasets. The representation of SDP15 allows for isolated words.

- **Chinese semantic Dependency Parsing** dataset (SDP16) (Che et al., 2012) is a Chinese SeDP benchmark. The sentences are gathered from News (NEWS) and textbook (TEXT). The representation of SemEval16 allows for multiple head words but does not have isolated words.

4.1.2 Evaluation Metrics

Following the conventions, we use unlabeled attachment score (UAS) and labeled attachment score (LAS) for SyDP. We use labeled attachment F1 Score (LF) on SDP15 of SeDP. For SeDP on SemEval16, we use unlabeled attachment F1 (UF) and labeled attachment F1 (LF). All the results are presented in percentages (%).

4.1.3 Implementations

We use T5-base (Raffel et al., 2020) and mT5-base (Xue et al., 2021) as the backbone PLM for datasets in English and datasets in other languages, respectively. In particular, we use V1.1 checkpoints, which are only pre-trained on unlabeled sentences, so as to keep the PLM unbiased.

The PLM is implemented with Huggingface Transformers (Wolf et al., 2020). The learning rate is $5 \times e^{-5}$, weight decay is $1 \times e^{-5}$. The optimizer is AdamW (Loshchilov and Hutter, 2019). The other details are shown in Appendix B.

4.2 Baselines

We divide baselines into three main categories based on their domain of expertise. We supplement more details about baselines in Appendix C.

In-domain and Multi-lingual SyDP. *Biaffine* (Dozat and Manning, 2017), *StackPTR* (Ma et al., 2018), and *CRF2O* (Zhang et al., 2020) introduce specially designed parsing modules without PLM. *CVT* (Clark et al., 2018), *MP2O* (Wang and Tu, 2020), *RNGTr* (Mohammadshahi and Henderson, 2021) and *MRC* (Gan et al., 2021) are recently proposed PLM-based dependency parser. *SeqNMT* (Li et al., 2018), *SeqViable* (Strzyz et al., 2019), and *PaT* (Vacareanu et al., 2020) cast dependency parsing as sequence labeling task, which is closely related to our sequence generation method. Apart from these methods, *HPSG* (Zhou and Zhao, 2019) and *HPSG+LA* (Mrini et al., 2020) combine with information of constituency parsing thus outperform all parsing baselines.

Param	Method (PLM)	UAS	LAS
-	CRF2O	96.14	94.49
-	Biaffine	95.74	94.08
-	StackPTR	95.87	94.19
340M	†MP2O (BERT-large)	96.91	95.34
335M	†MRC (RoBERTa-large)	**97.24**	95.49
-	†CVT (CVT)	96.60	95.00
110M	†RNGTr (BERT-base)	96.66	95.01
340M	*‡HPSG (BERT-large)	97.20	95.72
340M	*‡HPSG+LA (XLNet-large)	97.42	96.26
-	‡SeqNMT	92.08	94.11
-	‡SeqViable	93.67	91.72
110M	†‡PaT (BERT-base)	95.87	94.66
220M	†‡DPSG (T5-base)	96.64	**95.82**
60M	†‡DPSG (T5-small)	96.13	95.18

Table 1: Results on PTB for SyDP. Param means the number of parameters of the used PLM. The value '-' in this column means this is a method without PLM. ‡ means this method belongs to sequence-tp-sequence methods. † means this method use PLM. * means this method utilize additional constituency parsing information and is not comparable to other methods.

Unsupervised Cross-domain SyDP. Peng et al. (2019) and Li et al. (2019) modify the *Biaffine* for the unsupervised cross-domain DP. *SSADP* (Lin et al., 2021) relies on extra domain adaptation steps. In the PLM era, Li et al. (2019) propose *ELMo-Biaffine* with IFT on unlabeled target domain data.

SeDP. Dozat and Manning (2018) modify *Biaffine* for SeDP. *BS-IT* (Wang et al., 2018) is a transition-based semantic dependency parser with incremental Tree-LSTM. *HIT-SCIR* (Che et al., 2019) solves the SeDP with a BERT based pipeline. *BERT+Flair*[4] (He and D. Choi, 2020) augments the Biaffine model with BERT and Flair (Akbik et al., 2018) embedding. *MFVI* (Wang et al., 2019) and *Pointer* (Fernández-González and Gómez-Rodríguez, 2020) are the recently proposed excellent graph-based method and transition-based method for SeDP.

4.3 Main Results

We first validate the capacity of DPSG on both SyDP and SeDP in different languages and different domains, then we show that DPSG is transferable in low-resource scenarios including unsu-

[4]They use different pre-processing scripts on SDP15, thus are not comparable with DPSG and other baselines on SDP15.

Model	bg	ca	cs	de	en	es	fr	it	nl	no	ro	ru	AVG
CRF2O	90.77	91.29	91.54	80.46	87.32	90.86	87.96	91.91	88.62	91.02	86.90	93.33	89.33
MP2O	91.30	93.60	92.09	82.00	90.75	**92.62**	89.32	93.66	91.21	91.74	86.40	92.61	90.61
PaT	89.56	91.56	90.14	83.80	90.25	90.80	89.87	92.31	89.51	92.95	83.71	92.22	89.72
MRC	93.76	**94.38**	**93.81**	**85.23**	**91.95**	**92.62**	**91.76**	**94.79**	92.97	94.50	88.67	95.00	**92.45**
DPSG	**93.92**	93.75	92.97	84.84	91.49	92.37	90.73	94.59	92.03	**95.30**	**88.76**	**95.25**	92.17

Table 2: Results on 12 languages of UD2.2 in terms of LAS. All these baselines utilize PLM except CRF2O. The scales of used PLM are same with corresponding model in Table 1.

Method	NEWS		TEXT	
	UF	LF	UF	LF
BS-IT	81.14	63.30	85.71	72.92
BERT+Flair	82.92	67.27	**91.10**	80.41
DPSG	**84.31**	**70.82**	90.97	**82.36**

Table 3: Experimental results on SemEval16.

Method (ID/OOD)	DM	PAS	PSD
BS-IT	90.3/84.9	91.7/87.6	78.6/75.9
†HIT-SCIR (BERT-base)	92.9/89.2	94.4/92.4	81.6/81.0
MFVI	94.0/89.7	94.1/91.3	81.4/79.6
†Pointer (BERT-base)	**94.4**/**91.0**	**95.1**/**93.4**	82.6/**82.0**
DPSG (T5-base)	94.3/90.8	**95.1**/93.2	**83.1**/**82.0**

Table 4: Experimental results on SDP15 in terms of LF (including ID and OOD results). † means the model utilizing PLM.

pervised cross-domain DP and few-shot cross-task DP. Especially the few-shot cross-task experiments show that DPSG provides a principled way to explore the intrinsic connection among different structure parsing tasks.

4.3.1 DPSG is capable of DP

Single-lingual DP results are shown in Table 1 and Table 4. DPSG achieves the first-tier on PTB in Table 1 and SDP15 in Table 4. For SyDP, DPSG outperforms MRC, the SOTA model for DP, by 0.33% in LAS. For SeDP, DPSG obtain the comparable performance with Pointer.

In order to verify that the effectiveness of the proposed serialization scheme does not depend too much on the size of the PLM, we also use T5-small as the backbone for PTB. The result shows that the T5-small with only 60M parameters obtains performance comparable to other larger PLMs.

Multi-lingual DP results are shown in Table 2 and Table 3. DPSG obtain comparable performance to the SOTA baselines on UD2.2, and especially get the SOTA in bg, no, ro, and ru. For Chinese SeDP in Table 3, DPSG also obtain the SOTA performance. DPSG outperforms BERT+Flair to a large margin, achieves 3.55% performances gain on NEWS and 1.95% performances on TEXT with regard to LF.

4.3.2 Low-resource ability of DPSG

Unsupervised cross-domain is one of the common low-resource scenarios. Table 5 demonstrates the outstanding cross-domain transfer ability of DPSG. We enhance the unsupervised cross-domain capacity of DPSG with intermediate fine-tuning (IFT) (Pruksachatkun et al., 2020). Before training on the dependency parsing, the intermediate fine-tuning uses the unlabeled sentences in the target domain and continues to train the PLM in the source domain. We implement DPSG with and without IFT on the target domain. DPSG with IFT achieves the new SOTA, with a boosting of 5.06%, 7.21% and 10.49% in terms of LAS on PB, ZX, and PC, compared to ELMo with IFT.

Few-shot cross-task experiments are conducted on SyDP (PTB)→SeDP (SDP15). We randomly extract the 0.1%, 1% and 10% samples from the original SDP train sets to simulate the few-shot setting. The results are shown in Figure 3. Comparing with the vanilla T5, the DPSG with Tuning achieve higher performance in all few-shot settings. Especially in the low-resource scenario, the DPSG with Tuning only uses 35 training samples to achieve the 64.73% performance comparing with the original DPSG. Even in the OOD test, DPSG also get 65.63% performance. These results also demonstrate the intrinsic relationship between SeDP and SyDP. We supplement more detailed statistics in Appendix D.

5 Analysis

This section studies whether there is better implementation for DPSG. We are particularly interested

Category	Model	BC→ PB		BC→ ZX		BC→ PC		Average	
		UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
w/o PLM	Biaffine	67.75	60.95	69.41	61.55	39.95	26.96	59.04	49.82
	SSADP	68.55	61.59	70.82	63.61	41.10	27.67	60.16	50.96
w/ PLM	ELMo-Biaffine w/ IFT	77.15	71.54	74.68	67.51	53.04	39.48	68.29	59.51
	DPSG w/o IFT	78.86	73.28	75.74	69.42	54.00	41.98	69.53	61.56
	DPSG w/ IFT	**81.74**	**76.60**	**80.73**	**74.77**	**62.44**	**49.97**	**74.97**	**67.11**

Table 5: Results on CODT for unsupervised cross-domain SyDP.

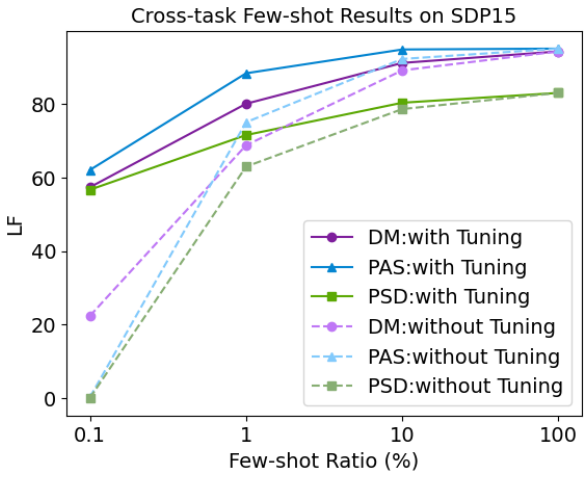

Figure 3: Compare curves of the DPSG with tuning on PTB and the DPSG without tuning on PTB. The training set sizes corresponding to 0.1%, 1%, 10% and 100% are 35, 356, 3565 and 35656, respectively.

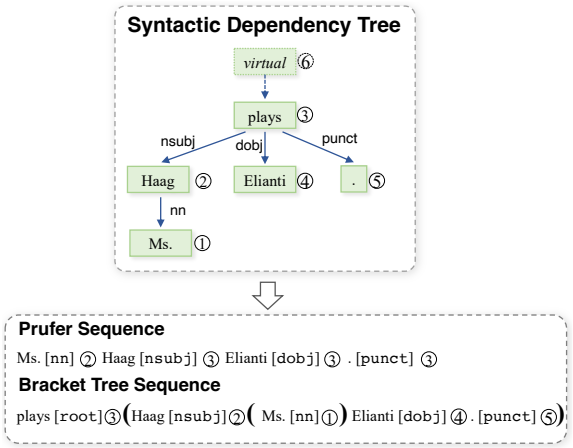

Figure 4: Prufer sequence and Bracket Tree sequence of the same sentence "*Ms. Haag plays Elianti .*".

Metric	DPSG	Prufer	Bracket	DPSG$_{\text{-pos}}$
UAS	96.64	88.62$_{\downarrow 7.98}$	95.37$_{\downarrow 1.27}$	96.48$_{\downarrow 0.16}$
LAS	95.82	86.77$_{\downarrow 11.32}$	93.76$_{\downarrow 2.06}$	95.04$_{\downarrow 0.78}$
Metric	DPSG$_{\text{-pid}}$	DPSG$_{\text{-pid-pos}}$	DPSG$_{\text{-rel}}$	DPSG$_{\text{BART}}$
UAS	95.20$_{\downarrow 1.42}$	93.12$_{\downarrow 3.52}$	96.45$_{\downarrow 0.19}$	86.35$_{\downarrow 10.13}$
LAS	93.17$_{\downarrow 2.65}$	92.30$_{\downarrow 3.52}$	95.54$_{\downarrow 0.28}$	79.45$_{\downarrow 15.59}$

Table 6: Results on PTB for exploratory experiment

in: 1) the designing of the Serializer, 2) the effect of the introduced special tokens, and 3) the choice of the PLM model. We use PTB as the benchmark and compare DPSG introduced in Section 3 with many other possible choices. The results of these exploratory experiments are shown in Table 6.

5.1 Serializer Designing

Tree, as the well-studied data structure for syntactic dependency parsing, has several other serialization methods to be converted into serialized representations. We explore the serializer designing of the tree structure in DPSG with two other widely used serialized representation—Prufer sequence and Bracket Tree, which are shown in Figure 4. Note that both Prufer sequence and Bracket Tree face the same word ambiguity issues; we associate each word with a unique position number as well.

Prufer Sequence is a unique sequence associated with the labeled tree in combinatorial mathematics.

Bracket Tree is one of the most commonly used serialization methods to represent the tree structure (Vinyals et al., 2015; Cross and Huang, 2016). More details about the Prufer sequence and the bracket tree are shown in Appendix E.

We denote the experimental results of Prufer sequence and bracket tree as Prufer and Bracket, respectively, in Table 6. Both Prufer sequence and bracket tree undermine the performance of DPSG to a large margin, which indicates that our proposed Serializer provides a better serialized representation for the PLM to generate. This is because our Serializer guarantees the dependency units in the output have the same order of the words in the input sentences, while Prufer sequence and bracket tree do not preserve the order. Thus, our

proposed DPSG *expands* the input sentence to generate the output sequence, while Prufer sequence and bracket tree based DPSG *reconstruct* the syntax dependency structure. As expansion strategy has smaller generation space than reconstruction, the serialization representation proposed in Section 3.1 eases the learning complexity of the PLM, and further brings better performance.

5.2 Special Tokens Designing

We further investigate whether the additionally introduced special tokens are useful.

Relation Tokens. There are two different ways to represent the dependency relations in the serialized representation: adding a special token for each dependency relation, or let the PLM directly generate the original relation. The later is denoted as DPSG$_{\text{-rel}}$ in Table 6. DPSG$_{\text{-rel}}$ is inferior than DPSG, which indicates that the special tokens for relations are useful. The reason is that adding the special token for dependency relation enables the PLM to learn embedding for the whole relation.

Positional Prompt. We are also particularly interested in the effectiveness of the positional prompts. We conduct experiments where the position and POS tagging are removed respectively and the experiment of removing both of them. The results are denoted as DPSG$_{\text{-pid}}$, DPSG$_{\text{-pos}}$ and DPSG$_{\text{-pid-pos}}$ in Table 6. DPSG$_{\text{-pid}}$ undermines the performance of DPSG because it requires the PLM to perform numerical reasoning, that is, to count for the position of each head word. DPSG$_{\text{-pos}}$ undermines the performance of DPSG because it leaks the POS tagging information.

5.3 Model Choosing

Both BART and T5 are widely used encoder-decoder PLMs. We try BART-base as the backbone PLM in DPSG. Table 6 shows that BART undermines the performance. In addition, BART has a significant performance drop after achieving the best performance, as shown in Appendix G.

5.4 Legality

There are two different legalities in DPSG. *Formation Legality* focus on whether the sequence has the correct formation (see Section 3.1) and *Structural Legality* focus on the legality of the corresponding parsing structure. The statistics on PTB show that the formation legality of DPSG is 100%, and the structure legality of DPSG is 99.7%, which is acceptable in practical usage.

Besides, the constraint decoding techniques can be incorporated to force the model to output legal output. During the beam search process, the constraint decoding exam the legality of current predicated sequence for all propoble head tokens and remove all head tokens which make the sequence become illegal.

6 Related Work

6.1 Syntactic Dependency Parsing

In-domain SyDP. Transition-based methods and graph-based methods are widely used in SyDP. Dozat and Manning (2017) introduce biaffine attention into the graph-based methods. Ma et al. (2018) adopt pointer network to alleviate the drawback of local information in transition-based methods. Fernández-González and Gómez-Rodríguez (2021) develop a bottom-up-oriented Hierarchical Pointer Network for the left-to-right dependency parser. As for the sequence-to-sequence methods, Strzyz et al. (2019) improve Li et al. (2018)'s method and explore more representation of predicated labeling sequence of dependency parsing. Vacareanu et al. (2020) use BERT to augment the sequence labeling methods.

Unsupervised Cross-domain SyDP. The labeling of parsing data requires a wealth of linguistics knowledge and this limitation facilitates the research of unsupervised cross-domain DP. Yu et al. (2015) introduce pseudo-labeling unsupervised cross-domain SyDP via self-training. Li et al. (2019) propose a cross-domain datasets CODT for SyDP and build baselines for unsupervised cross-domain SyDP.

6.2 Semantic Dependency Parsing

Buys and Blunsom (2017) accomplish the first transition-based parser for Minimal Recursion Semantics (MRS). Zhang et al. (2016) present two novel transition-systems to generate arbitrary directed graphs in an incremental manner. Wang et al. (2019) consider the secord-order information for SeDP. Fernández-González and Gómez-Rodríguez (2020) improve the pointer network with transition-based method.

6.3 Comparing with Point Network

Although DPSG is the first parser to achieve parsing completely rely on PLM via sequence generation, its mechanism has some similarities with some previous transition-based methods.

Fernández-González and Gómez-Rodríguez (2019) and Fernández-González and Gómez-Rodríguez (2020) both use the left-to-right point network to achieve parsing by auto-regressive "generation", which imply some idea of serialization. Comparing with these methods, DPSG remove the pointer network within stack and buffers, which makes the architecture more simple. However, due to the existence of specially designed pointer networks and action sequences, such methods have the possibility of further improving the network structure to improve the parsing performance. On the contrary, DPSG relies more on the design of the serialization scheme to improve the performance, and it is relatively difficult to improve the performance via improving the PLM structure. In addition, the inference speed of pointer network is faster than that of DPSG based on PLM for inference.

7 Conclusion

This paper proposes DPSG, a generative sequence-to-sequence dependency parsing method. By serializing the parsing structure to a flattened sequence, PLM can directly generate the parsing results in serialized representation. DPSG not only achieves good results in different parsing representations, but also performs surprisingly well on unsupervised cross-domain DP. The few-shot transfer experiments also suggest that DPSG is capable of investigating the inner connection between SeDP and SyDP. The exploratory experiments and analyses demonstrate the rationality of the designing of DPSG. Considering the unity, indirectness, and effectiveness of DPSG, we believe the sequence generative is one of the effective sequence-to-sequence methods for dependency parsing.

8 Limitations

Considering the powerful learning ability of pre-trained language model (PLM) and the simplicity of the overall process, we do not add constraint decoding or post-processing to DPSG, which makes the output of the model possibly illegal. In section 5.4, we have introduce some statistics of legality. Although the statistics show that only a small percentage of outputs is illegal, considering the complexity of the data in practical applications, we still consider that legality problem is a limitation of our work.

In the future research, we consider adding the constrained decoding in the generation process of DPSG.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (61702047, 62276195), Natural Science Foundation of China Youth Project (62006136) and a grant from the Institute for Guo Qiang, Tsinghua University (2019GQB0003).

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing machine translation with dependency-aware self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627, Online. Association for Computational Linguistics.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Avani Chandurkar and Ajay Bansal. 2017. Information retrieval from a structured knowledgebase. In *11th IEEE International Conference on Semantic Computing, ICSC 2017, San Diego, CA, USA, January 30 - February 1, 2017*, pages 407–412. IEEE Computer Society.

Wanxiang Che, Longxu Dou, Yang Xu, Yuxuan Wang, Yijia Liu, and Ting Liu. 2019. HIT-SCIR at MRP 2019: A unified pipeline for meaning representation parsing via efficient training and effective encoding. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 76–85, Hong Kong. Association for Computational Linguistics.

Wanxiang Che, Meishan Zhang, Yanqiu Shao, and Ting Liu. 2012. SemEval-2012 task 5: Chinese semantic dependency parsing. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 378–384, Montréal, Canada. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings*

of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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*. OpenReview.net.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All nlp tasks are generation tasks: A general pretraining framework. *ArXiv preprint*, abs/2103.10360.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. 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 710–716. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Transition-based semantic dependency parsing with pointer networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7035–7046, Online. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021. Dependency parsing with bottom-up hierarchical pointer networks. *ArXiv preprint*, abs/2105.09611.

Leilei Gan, Yuxing Meng, Kun Kuang, Xiaofei Sun, Chun Fan, Fei Wu, and Jiwei Li. 2021. Dependency parsing as mrc-based span-span prediction. *ArXiv preprint*, abs/2105.07654.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

Han He and Jinho D. Choi. 2020. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with BERT. In *Proceedings of the Thirty-Third International Florida Artificial Intelligence Research Society Conference, Originally to be held in North Miami Beach, Florida, USA, May 17-20, 2020*, pages 228–233. AAAI Press.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zhenghua Li, Xue Peng, Min Zhang, Rui Wang, and Luo Si. 2019. Semi-supervised domain adaptation for dependency parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2386–2395, Florence, Italy. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Boda Lin, Mingzheng Li, Si Li, and Yong Luo. 2021. Unsupervised domain adaptation method with semantic-structural alignment for dependency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2158–2167, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland. Association for Computational Linguistics.

Xue Peng, Zhenghua Li, Min Zhang, Wang Rui, Yue Zhang, and Luo Si. 2019. Overview of the nlpcc 2019 shared task: Cross-domain dependency parsing. In *Natural Language Processing and Chinese Computing - 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9-14, 2019, Proceedings, Part II*, volume 11839, pages 760–771. Springer.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. 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 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Damien Teney, Lingqiao Liu, and Anton van den Hengel. 2017. Graph-structured representations for visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3233–3241. IEEE Computer Society.

Robert Vacareanu, George Caique Gouveia Barbosa, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2020. Parsing as tagging. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5225–5231, Marseille, France. European Language Resources Association.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A neural transition-based approach for semantic dependency graph parsing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5561–5568. AAAI Press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Juntao Yu, Mohab Elkaref, and Bernd Bohnet. 2015. Domain adaptation for dependency parsing via self-trainging. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 1–10, Bilbao, Spain. Association for Computational Linguistics.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics*, 42(3):353–389.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Set	Section	Sentences	Words
Train	[2-21]	39,832	95,0028
Dev	[22]	1,700	40,117
Test	[23]	2,416	56,684

Table 7: Data statistics of PTB.

Domain	Train Set	Dev Set	Test Set	Unlabeled Set
BC	16.3K	1K	2K	–
PB	5.1K	1.3K	2.6K	291K
PC	6.6K	1.3K	2.6K	349K
ZX	1.6K	0.5K	1.1K	33K

Table 8: Data statistics of CODT.

Schema	Train Set	ID Test Set	OOD Test Set
DM	35,656	1,410	1,849
PAS	35,656	1,410	1,849
PSD	35,656	1,410	1,849

Table 9: Data statistics of SDP15.

Domain	Train Set	Dev Set	Test Set
NEWS	8,301	534	1,233
TEXT	128,095	1,546	3,096

Table 10: Data statistics of SemEval16.

A Dataset Statistics

The details about the statistics of datasets used in this paper are shown on Table 7, Table 8, Table 9 and Table 10.

B More Details on Implementations

We conduct all the experiments on Tesla V100. The majority of our experiments on DPSG consume about 22GiB GPU memory. Generally speaking, it takes about one and a half day for DPSG to reach the best performance in Tesla V100. But the datasets with larger dev sets and test sets need more time for implementation, such as *cs* in the UD2.2.

C More Details on Baseline

Baselines for in-domain SyDP.

- [5] **Biaffine:** Dozat and Manning (2017) adopt biaffine attention mechanism into the graph-based method of dependency parsing.
- **StackPTR:** Ma et al. (2018) introduce the pointer network into the transition-based methods of dependency parsing.
- **CRF:** Zhang et al. (2020) improve the CRF to capture more high-order information in dependency parsing.
- [6]**SeqNMT:** Li et al. (2018) use an Encoder-Decoder architecture to achieve the Seq2Seq dependency parsing by sequence tagging. The BPE segmentation from Neural Machine Translation (NMT) and character embedding from AllenNLP (Gardner et al., 2018) are applied to argument their model.
- **SeqViable:** Strzyz et al. (2019) explore four encodings of dependency trees and improve the performance comparing with Li et al. (2018).
- **PaT:** Vacareanu et al. (2020) use a simple tagging structure over BERT-base to achieve sequence labeling of dependency parsing.

+ [7] **CVT:** Clark et al. (2018) propose another pre-train method named cross-view training, which can be used in many sequence constructing task including SyDP. The best results of CVT is achieved by the multi-task pre-training of SyDP and part-of-speech tagging.

+ **MP2O:** Wang and Tu (2020) use message passing GNN based on BERT to capture second-order information in SyDP.

+ **MRC:** Gan et al. (2021) use span-based method to construct the edges at the subtree level. The Machine Reading Comprehension (MRC) is applied to link the different span. RoBERTa-large (Liu et al., 2019) is applied to enhance the representation of parser.

∗ [8] **HPSG:** Zhou and Zhao (2019) view Head-driven Phrase Structure Grammar (HPSG) as the combination of dependency parsing and constituency parsing and achieve both the state-of-the-art performance in dependency parsing and constituency parsing.

∗ **HPSG-LA:** Mrini et al. (2020) propose Label Attention Layer and utilize the training procedure of HPSG parser to obtain the best performance for dependency parsing until now.

Baselines for cross-domain SyDP.

∗ **Biaffine:** Peng et al. (2019); Li et al. (2019) use Biaffine trained on source domain and test on

[5] - means model without PLM
[6] • means sequence-based methods
[7] + means model utilizing PLM
[8] ∗ means model utilizing HPSG information

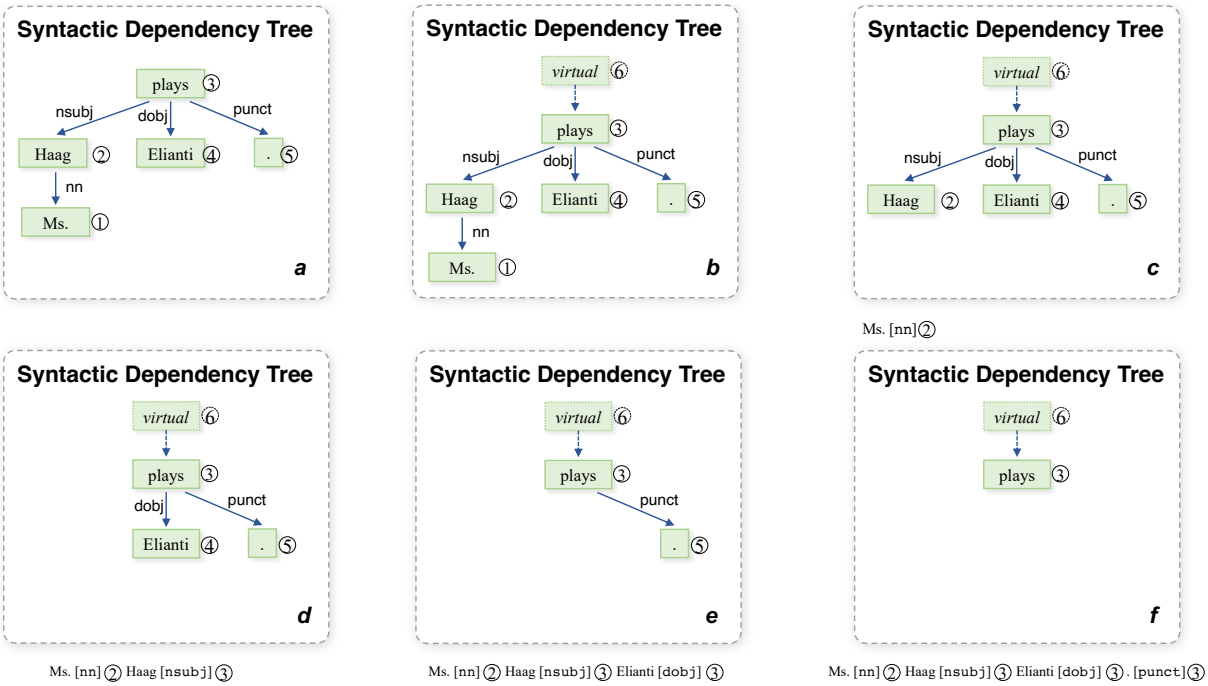

Figure 5: The Prufer Sequence of sentence "*Ms. Haag plays Elianti .*" is constructed from ***a*** to ***f***.

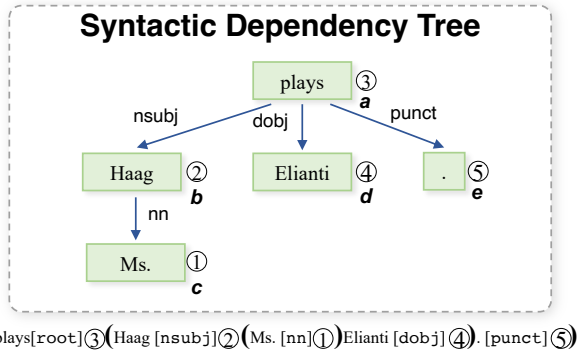

Figure 6: The Bracket Tree Sequence of sentence "*Ms. Haag plays Elianti .*" is constructed following the topological order from ***a*** to ***e***.

target domain as the baseline of unsupervised cross-domain SyDP.

* **SSADP:** Lin et al. (2021) use both semantic and structural feature to achieve the domain adaptation of unsupervised cross-domain parsing.

\+ **ELMo:** Li et al. (2019) use ELMo with intermediate fine-tuning in unlabeled text of target domain to achieve the SOTA on unsupervised cross-domain SyDP.

Baselines for SeDP.

* **Biaffine:** Dozat and Manning (2018) transfer the Biaffine model from SyDP to SeDP.

* **BS-IT:** Wang et al. (2018) use graph-based method for SeDP.

- **HIT-SCIR:** Che et al. (2019) propose a BERT-based pipeline model for SeDP.
- **BERT+Flair:** He and D. Choi (2020) use BERT and flair embedding (Akbik et al., 2018) to argument their modificated Biaffine.
- **MFVI:** Wang et al. (2019) consider the interactions between pairs of edges in semantic graph, which apply mead field variational inference and loopy belief propagation into neural network.
- **Pointer:** Fernández-González and Gómez-Rodríguez (2020) improve the transition-based method with pointer network for semantic dependency parsing.

D Few-shot Transfer

The details of the statistics about the cross-task few-shot transfer experiments on SDP15 are shown on Table 11.

E Construction of Prufer Sequence

E.1 Prufer Sequence

The prufer algorithm which converts labeled tree into Prufer sequence does not preserve the root node, while in dependency parsing, the root is a

Shot / Ratio	Method (ID)	DM	PAS	PSD	Avg(%)
(35656 / 100%)	DPSG	94.3	95.1	83.1	100%
(3565 / 10%)	with Tuning / without Tuning	91.1/89.1	94.8/92.2	80.3/78.6	97.64%/95.34%
(356 / 1%)	with Tuning / without Tuning	80.0/68.8	88.3/75.0	71.5/63.0	87.91%/75.81%
(35 / 0.1%)	with Tuning / without Tuning	57.3/22.4	62.0/0.3	56.7/0.1	64.73%/8.05%
Shot / Ratio	Method (OOD)	DM	PAS	PSD	Avg(%)
(35656 / 100%)	DPSG	90.8	93.2	82.0	100%
(3565 / 10%)	with Tuning / without Tuning	86.6/84.1	90.4/89.1	78.6/76.9	96.07%/94.00%
(356 / 1%)	with Tuning / without Tuning	76.0/63.6	83.8/69.6	69.8/59.8	86.24%/72.55%
(35 / 0.1%)	with Tuning / without Tuning	57.4/20.0	61.4/0.8	55.6/0.1	65.63%/7.66%

Table 11: Few-shot transfer (PTB → SDP15) experimental results in terms of LF.

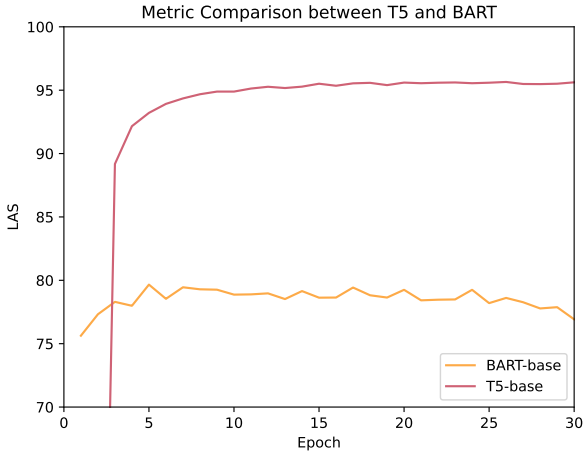

Figure 7: The LAS curves on dev sets of PTB between of T5-base and BART-base.

unique word. To bridge this inconsistency, we introduce an additionally added virtual node to the dependency tree to mark the root word.

The principle of construction is deleting the leaf node with minimum index and adding the index of its farther node into the prufer sequence. This process is repeated more times until there are only two nodes left in the tree.

E.2 Prufer for Parsing Tree

The arc in parsing tree is directed and thus is a rooted tree. When all the son nodes with smaller index are deleted, the root node will be treated as a leaf node then deleted in the next step. To address this problem, we add a virtual node with the maximum index and build a arc from virtual node to the real root. This virtual root force the root node always being a leaf node in the whole construction of prufer sequence. The overall construction process as shown on Figure 5 (a)~(f).

F Construction of Bracket Tree

The Bracket Tree uses *Bracket* to indicate levels of nodes. By recursively putting the sub-tree nodes in a pair of brackets from left-to-right, bracket tree can build a bijection between parsing tree and bracket tree. All the nodes belonging to the same level are wrapped in the same pair of brackets. The process of construction is shown on Figure 6.

G Comparison between T5 and BART

Figure 7 shows the LAS comparison on dev sets of PTB between the T5 and BART in first 30 epochs. After the first two epochs, the performance of T5 raise rapidly and can better maintain performance in the later stages of training. Although BART achieves a better performance in the first two round, but there is not much room for performance improvement. To make matters worse, it can be clearly seen that after achieving the best performance, BART is very unstable, and even a significant performance drop has occurred.