A Semi-Autoregressive Graph Generative Model for Dependency Graph Parsing

Ye Ma, Mingming Sun, Ping Li
Cognitive Computing Lab
Baidu Research
No.10 Xibeiwang East Road, Beijing 100193, China
10900 NE 8th St. Bellevue, Washington 98004, USA
{maye811906,sunmingming01, pingli98}@gmail.com

Abstract

Recent years have witnessed the impressive progress in Neural Dependency Parsing. According to the different factorization approaches to the graph joint probabilities, existing parsers can be roughly divided into autoregressive and non-autoregressive patterns. The former means that the graph should be factorized into multiple sequentially dependent components, then it can be built up component by component. And the latter assumes these components to be independent so that they can be outputted in a one-shot manner. However, when treating the directed edge as an explicit dependency relationship, we discover that there is a mixture of independent and interdependent components in the dependency graph, signifying that both aforementioned models fail to precisely capture the explicit dependencies among nodes and edges. Based on this property, we design a Semi-Autoregressive Dependency Parser to generate dependency graphs via adding node groups and edge groups autoregressively while pouring out all group elements in parallel. The model gains a trade-off between non-autoregression and autoregression, which respectively suffer from the lack of target inter-dependencies and the uncertainty of graph generation orders. The experiments show the proposed parser outperforms strong baselines on Enhanced Universal Dependencies of multiple languages, especially achieving 4% average promotion at graph-level accuracy. Also, the performances of model variations show the importance of specific parts.

1 Introduction

Dependency graph in neural parsing is a directed graph representing semantic dependencies between words, with a transitive relation traveling from the rooted node to all words in the sentence phase by phase. As such, transition-based parsing seems to be a natural choice, as it builds up the parsing graph sequentially so that the dependency relationships can be captured. However, graph-based parsing dominates recent competitions on parsing technologies including IWPT 2020 and 2021 (Bouma et al., 2020, 2021), even if using a simple biaffine attention (Dozat and Manning, 2017) only to predict the whole graph at once. To explore a more effective parsing method that can represent these dependency relationships in a rigorous manner, we define and construct Topological Hierarchies for dependency graphs based on the explicit dependencies carried by them. According to the characteristics of topological hierarchies, we proposes a **S**emi-**A**utoregressive Dependency **G**raph Pars**er** (**SAGER**) – a novel graph-based parsing fashion via the semi-autoregressive graph generation.

Generally, autoregressive graph generation indicates that the model dynamically adds nodes and edges based on the generated sub-graph structure until reaching the complete graph. It brings a challenge to determine the generation order of graphic data, because there is no conventional reading order like text data (Chen et al., 2021). Although the dependency graphs stipulate the strict sequential dependencies by directed edges, there is a lack of such topological orders between sibling nodes. For instance, node $\mathbf{b}$ in Figure 1.a depends on the node $\mathbf{a}$ because there is an explicit edge pointing from $\mathbf{a}$ to $\mathbf{b}$. However, it is hard to decide the dependency relationship between node $\mathbf{d}$ and node $\mathbf{e}$ as they are not linked neither directly nor indirectly. Previous works on directed graph generation solve the problem surfacely. Cai and Lam (2019, 2020) sort these sibling nodes randomly at the early stage of training and then change them to a deterministic order (e.g., relation-frequency) at later training steps. Some other works do the sorting by referring to the orders of the known sequences like word order or alphanumerical order (Zhang et al., 2019a,b; Bevilacqua et al., 2021). Such imposed orders would bring more exposure bias (Ranzato et al., 2016) to teacher forcing training. More specifically, these sibling

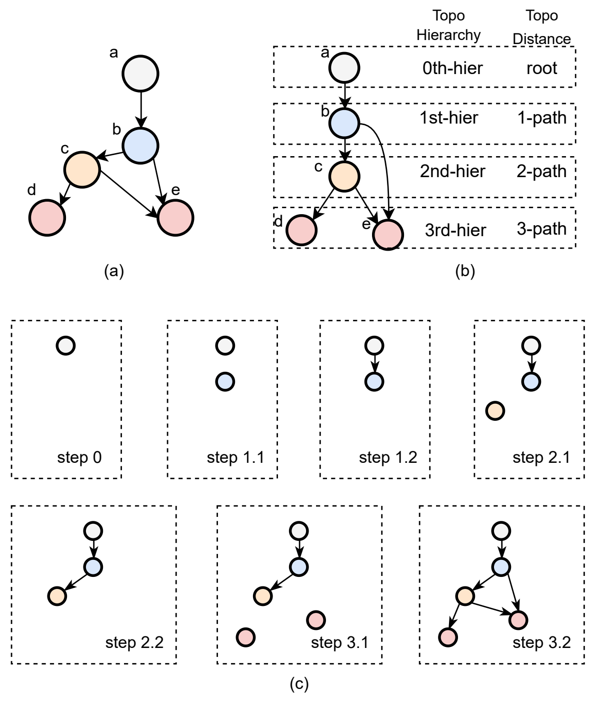

Figure 1: (a) An example graph (b) Divide nodes into different topological hierarchies based on their furthest distances from the root node. (c) Semi-autoregressive graph generation process.

nodes are essentially orderless and non-interlogical. They cannot form stationary and logical patterns like words in sentences. As a result, it is hard for the sibling nodes to be generated in the same order as in the training, causing learned knowledge invalid and later predictions misled. The aforementioned random ordering seems to alleviate the problem to some extent, but it destabilizes and complicates the training process and generally results in inferior models.

Instead of imposing orders on these sibling nodes, we assume them (including their incoming edges) to be conditionally independent to construct topological hierarchies as the generation orders. As shown in Figure 1.b, we divide nodes into several hierarchies according to their furthest distances from the root node. We can see that there are no explicit dependency relationships between nodes in the same hierarchy. Besides, nodes in the later hierarchies only depends on those in the previous hierarchies, forming a natural generation sequence. For a directed acyclic graph (DAG), it *at least* has one topological ordering but *only* has one topological hierarchy. At each generation step, we firstly predict all new nodes in parallel and then calculate their incoming edges by the biaffine attention (Dozat and Manning, 2017). In a word, our model autoregressively adds node groups and edge groups but non-autoregressively generates elements in these groups. See Figure 1.c for our semi-autoregressive generation process.

Another challenge lies at the probable edge sparsity problem when representing sub-graph structures. Sparse graph representation models like GCN (Kipf and Welling, 2017) heavily rely on the given adjacency to capture context information. That means it may fail to represent historical information completely and efficiently when predicted edges make mistakes. An extreme situation of edge sparsity is that the new nodes have no incoming edges predicted so that the model can only represent its node features rather than the sub-graph structure. To enhance the robustness of the generator, we adopt the dense structure of Transformer (Vaswani et al., 2017). In our Graph Transformer, there are implicit edges linking from the nodes in the previous and current hierarchies to the new node. Then, the predicted explicit edges serve as the bias to adjust the attention distribution over the implicit edges so that the model can adaptively select useful structural information.

Overall, this paper proposes a novel direction – semi-autoregression to deal with dependency parsing problems, distinguished with autoregression and non-autoregression (detailed definitions about them are available in § 2). With the dependencies denoted as the directed edges, the semi-autoregressive pattern unfolds graphs in the order of topological hierarchies, which strictly follows the explicit dependency relationships defined in dependency graphs. Our model strikes a reasonable balance, resulting in it neither ignoring inter-output dependencies like non-autoregressive models, nor suffering from severe exposure bias in generation orders like autoregressive models On the experimental side, we evaluate SAGER on Enhanced Universal Dependencies (EUD) which are non-tree dependency graphs. In addition to the official evaluation metric Enhanced Label Attachment Scores (ELAS), we design a graph-level matching score (GMS) to assess the probability of returning an completely correct graph. Our model outperforms other baselines in both metrics especially GMS, showing that it is good at overcoming some intractable prediction errors. Finally, we introduce multiple model variations to investigate the effect of different model components and show that our model is well-designed, especially the parts of discarding imposed orders and adding implicit edges.

2 Related Work

Autoregressive Parser. Generally, a generator is in an autoregressive fashion provided its generation probability at each step is conditional on items it produces previously. Transition-based parser obviously conforms to the characteristic, as it updates the action probability every step based on the words, tags and label embeddings previously put in the buffer and stack (Chen and Manning, 2014). Meanwhile, we note that some mechanisms commonly used in autoregressive generators are used to improve transition-based parsers like beam search and pointer networks (Weiss et al., 2015; Ma et al., 2018; Fernández-González and Gómez-Rodríguez, 2019). On the other hand, Cheng et al. (2016) proposes a graph-based autoregressive parser by adding arcs sequentially with the considerations of previous parsing decisions. However, it should not be taken as a rigorous graph generative model as it does not generate by extending the sub-graph structures. Actually, instead of dependency graphs, it is more prevalent that leverage the autoregressive graph generators to parse Abstract Meaning Representation (AMR) (Cai and Lam, 2019, 2020; Zhang et al., 2019b,a). They are all in the (fully) autoregressive pattern that an order is imposed to nodes and edges without topological orderings. In this paper, we focus on investigating the effects of these imposed orderings by introducing some variations of the proposed model.

Non-Autoregressive Parser. In contrast, non-autoregression implies that all components factorized from the graph are independent, so their probabilities do not affect each other and can be obtained in parallel at any time. A representative non-autoregressive parser is Deep Biaffine Attention (BiAtt) (Dozat and Manning, 2017) which assuming all edges are independent. For the tree-structure dependency graphs, it is often followed by a searching algorithm for the Maximum Spanning Tree (MST). Some heuristic algorithms (Li et al., 2020; Kiperwasser and Goldberg, 2016) construct the MST step by step, which yet does not mean they are in the autoregressive manner because all edge probabilities are predicted at once and fixed before the searching. As to the higher-order graph-based parsers, Ji et al. (2019) incorporates the second-order knowledge into the word representations and still uses the BiAtt as the final parser. Wang et al. (2019); Zhang et al. (2020) decompose the graph into components of different second-order parts. Different from BiAtt that each component is an edge, here some components consists of two edges whose joint probabilities can be calculated as a whole by a trilinear function. They still belong to non-autoregressive parsers because their components are independent of each other and disable to be subdivided.

3 Proposed Model

3.1 Definitions

Problem Definition. Conditional on the source sentence $S = (w_n)_{n=1}^{N}$, the task is to generate a dependency graph hierarchy by hierarchy. The generation process can be denoted as a sequence of components: $(C^{(t)})_{t=0}^{T}, T \leq N$. We firstly turn dependency graphs to DAGs by deleting the back edges in their cycles. It should be mentioned that there are only a few graphs with cycles and we can add these removed edges back by rules before evaluations. Then we can construct topological hierarchies based on the furthest distance from each node to the root node. The initial component $C^{(0)} = \{v_0\}$ in the 0-th hierarchy only has a root node. When $t > 0$, the component in the t-th hierarchy is defined as $C^{(t)} = \{V^{(t)}, E^{(t)}\}$. Let $\mathcal{V}_t = \bigcup_{j=0}^{t} V^{(j)}$, then $V^{(t)} = \{v_i\}_{i=|\mathcal{V}_{t-1}|}^{|\mathcal{V}_t|-1}$ is a set of nodes in the t-th hierarchy. And, $E^{(t)} = \{(v_j, v_i, z_{ji}) | v_j \in \mathcal{V}_{t-1}, v_i \in V^{(t)}\}$ is a set of edges pointing from nodes in the previous hierarchies to the current nodes, where v_j is the head of v_i and z_{ji} is the label on the arc.

Explicit and Implicit Edge. We define two kinds of edges, namely *explicit* edges and *implicit* edges. The former is what we need to really predict. Let $\mathcal{N}_i$ be the explicit first-order neighbours of the node $v_i \in V^{(t)}$ and $\mathcal{D}_i$ be the implicit neighbours, and $\mathcal{N}_i \cup \mathcal{D}_i = \mathcal{V}_t$. Notably, nodes in $\mathcal{N}_i$ can not appear in $V^{(t)}$ according to the definition of topological hierarchy. They have uni-directional edges pointing to the node v_i with arc labels, and these edge can be found in $E^{(t)}$. On the other hand, nodes in $\mathcal{D}_i$ should not have pointed to v_i, but our model does so because we expect nodes to learn structural information adaptively. It should be mentioned that nodes in the same component or hierarchy also have implicit edges linking to each other, i.e., $V^{(t)} \subseteq \mathcal{D}_i$.

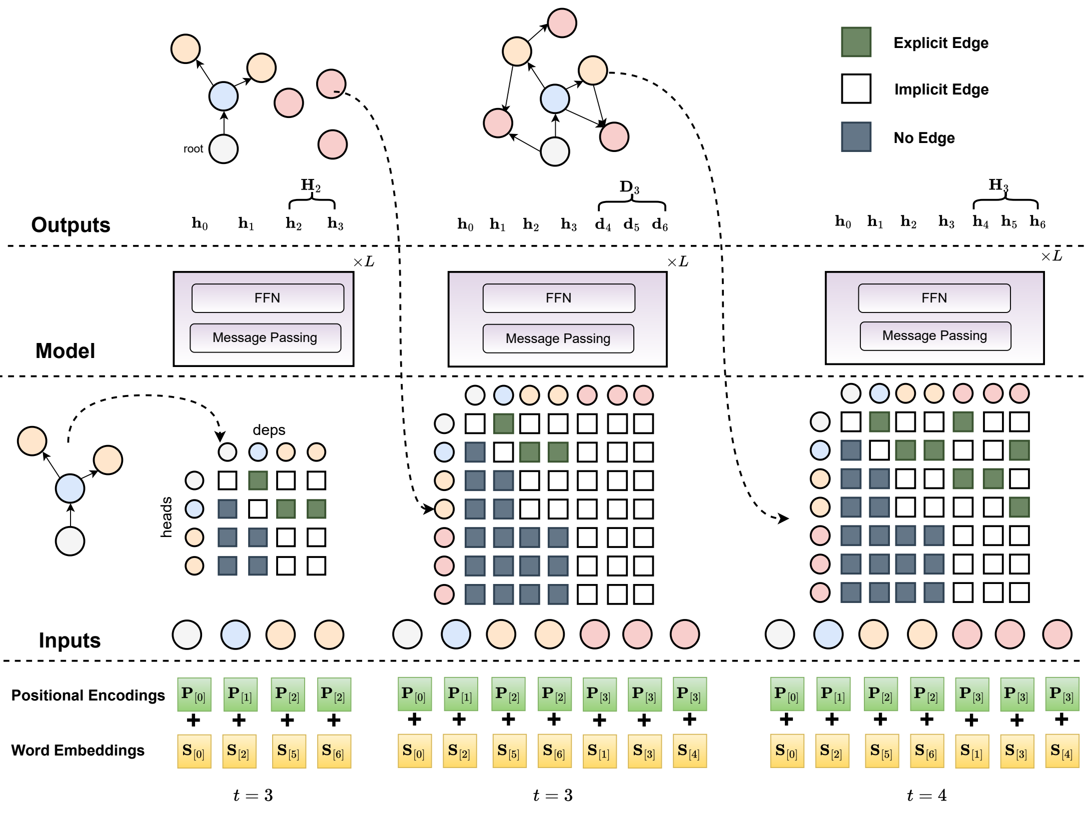

Figure 2: Semi-autoregressive generation process and graph transformer.

Head and Dependent Representation. We define two representations of the same node with different roles, namely the *head* representation and the *dependent* representation. Each generated node will first be used as a dependent node to calculate its incoming arcs, and then as a head node until the end of generation. We define the head vector of a node $v_i \in V^{(t)}$ as $\mathbf{h}_i$ and its dependent vector as $\mathbf{d}_i$. For a component, its head matrix $\mathbf{H}^{(t)} = \mathcal{F}_\theta(\mathcal{V}_t, \mathcal{E}_t, S)$ and dependent matrix $\mathbf{D}^{(t)} = \mathcal{F}_\theta(\mathcal{V}_t, \mathcal{E}_{t-1}, S)$ are the concatenations of multiple corresponding node representation, where $\mathcal{E}_t = \bigcup_{j=0}^{t} E^{(t)}$. We can see that the difference between them is that the latter inputs lack $E^{(t)}$, which means there are no available explicit edges pointing to $V^{(t)}$ nodes when calculating dependent representations. We shall mention that the graph representation model $\mathcal{F}_\theta(\cdot)$ can represent all components, but we only need to focus on the new component at each generation step because the new component does not affect node representations in the previous components.

Training Objective. The objective is to maximize the graph joint probability $\mathcal{J}$:

$$\mathcal{J} = \prod_{t=1}^{T} P(V^{(t)}|\mathcal{V}_{t-1}, \mathcal{E}_{t-1}) P(E^{(t)}|\mathcal{V}_t, \mathcal{E}_{t-1})$$

$$P(V^{(t)}|\mathcal{V}_{t-1}, \mathcal{E}_{t-1}) = \prod_{v_i \in V^{(t)}} P(v_i|\mathcal{V}_{t-1}, \mathcal{E}_{t-1})$$

$$P(E^{(t)}|\mathcal{V}_t, \mathcal{E}_{t-1}) = \prod_{e_i \in E^{(t)}} P(e_i|\mathcal{V}_t, \mathcal{E}_{t-1})$$

$$\mathcal{V}_t = \mathcal{V}_{t-1} \cup V^{(t)}, \ \mathcal{E}_t = \mathcal{E}_{t-1} \cup E^{(t)}$$

It indicates that we autoregressively generate the new node group $V^{(t)}$ and the edge group $E^{(t)}$ based on groups generated previously and the elements in the same group are independent.

3.2 Graph Generation Process

Figure 2 presents the generative process from the 3-rd step to the 4-th step. Specifically, At the generation step t, we firstly update head representations $\mathbf{H}^{(t-1)}$ for the last-step nodes $V^{(t-1)}$ using their network structure information $E^{(t-1)}$. Notably, although there only generates an intermediate sub-graph of the entire structure, the explicit topological information of $V^{(t-1)}$ nodes is completed because they would not have incoming arcs from nodes generated later. On the other hand, these sentence words have been represented as a dense matrix $\mathbf{S}$ by a Transformer-encoder. Then, their probabilities of being selected are calculated by:

$$P(w_{1:N}) = \text{MaxPool}\left[\sigma\left(\mathbf{H}^{(t-1)}\mathbf{W}_1\mathbf{W}_2^\top\mathbf{S}^\top\right)\right]$$
$$V^{(t)} = \{w_n|P(w_n) > 0.5\}$$

where $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a linear transformation matrix. This operation is similar to a multi-label classification. Every source word is assigned with an independent probability, and words with probabilities larger than 0.5 are selected as new nodes $V^{(t)}$. To represent these new nodes as $\mathbf{D}^{(t)}$ when their network structural information are unknown, we suppose that there are implicit edges pointing from previous nodes to these nodes. Besides, these new nodes are connected to each other by implicit edges. Although it is impossible to appear explicit edges among them, this operation can further enrich node representations. Their connections are illustrated by the second adjacency matrix in the middle block of Figure 2. Explicit edge connections and types are then figured out by Deep Biaffine Attention (Dozat and Manning, 2017):

$$E^{(t)} = \text{DeepBiaffine}\left(\|_{j=0}^{t-1}\mathbf{H}^{(j)}, \mathbf{D}^{(t)}\right)$$

where $\|_{j=0}^{t-1}\mathbf{H}^{(j)}$ is achieved by concatenating head representations of all nodes in the previous hierarchies. The generation proceeds via repeating the aforementioned operations until no words can be selected as new nodes.

3.3 Graph Representation Model

Recently, Transformer (Vaswani et al., 2017) has made impressive progress in the graph representation field (Ying et al., 2021). In essence, Transformer regards inputs as an undirected fully-connected graph, thus serving as a special graph representation model that can enjoys global perception at all layers. Previous works focusing on adapting Transformer-encoder to node or graph classification, while this paper modifies Transformer-decoder to conduct graph generation.

Let $\mathbf{x}_i^{(l)}$ denote the node v_i embedding at the l-th layer. If the node v_i is in the component C_t and copied from the source word w_n, its initial node embedding $\mathbf{x}_i^{(0)}$ should be the summation of:

$$\mathbf{x}_i^{(0)} = \mathbf{S}_{[n]} + \mathbf{P}_{[t]}$$

where $\mathbf{S}, \mathbf{P}$ indicate word embeddings and hierarchical positional encodings respectively, as shown in Figure 2. Nodes in the same hierarchy have the same hierarchical positional encodings.

The message passing layer actually takes the position of the masked self-attention layer in the decoder. The original decoder self-attention helps every word to aggregate left-ward contexts. In contrast, every node in our model not only aggregates left-ward contexts (i.e, nodes in previous hierarchies), but also nodes in the same hierarchy. To distinguish explicit edges and implicit edges, the message vector $\mathbf{m}_{ji}$ of the node v_j with an explicit edge pointing to the node v_i should be enriched with prior structural knowledge by

$$\mathbf{m}_{ji}^{(l)} = \begin{cases} \mathbf{x}_j^{(l)} + \text{relu}\left(\mathbf{x}_j^{(l)}\mathbf{U}_{z_{ji}}\right), & v_j \in \mathcal{N}_i \\ \mathbf{x}_j^{(l)}, & v_j \in \mathcal{D}_i \end{cases}$$

where $\mathbf{U}_{z_{ji}} \in \mathbb{R}^{d\times d}$ indicates the parametric embedding matrix of the edge label z_{ji}. These edge embedding metrics are shared across all layers. Notably, we assume that the central node v_i is self-connected implicitly, i.e. $v_i \in \mathcal{D}_i$. The reduction function is then defined as the multi-head attention:

$$\alpha_{ji} = \frac{\exp\left(\mathbf{x}_i\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{m}_{ji}^\top\right)}{\sum_{v_u\in\mathcal{N}_i\cup\mathcal{D}_i}\exp\left(\mathbf{x}_i\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{m}_{ui}^\top\right)}$$

$$\mathbf{x}_i^{(l)\prime} = \left[\Big\|_{h=1}^{H}\left(\sum_{v_j\in\mathcal{N}_i\cup\mathcal{D}_i}\alpha_{ji}^h\mathbf{m}_{ji}^{(l)}\mathbf{W}_V^h\right)\right]\mathbf{W}_O$$

We can see that the query is the node embedding $\mathbf{x}_i$, and the keys and values are those message vectors $\mathbf{m}_{ji}$. Its output $\mathbf{x}_i^{(l)\prime}$ is then fed into the feed-forward layer to enter the next layer:

$$\mathbf{x}_i^{(l+1)} = \text{FFN}\left(\mathbf{x}_i^{(l)\prime}\right)$$

The outputs $\mathbf{x}^{(L)}$ of the final layer are head representations or dependent representations.

The edge embedding matrices $\mathbf{U}$ give the model access to prior structural knowledge and enable it to select useful prior knowledge adaptively. When all structural knowledge is useless (i.e, parameters in $\mathbf{U}$ are trained to be zeros) and each hierarchy only contains one node, the graph model degrades to a vanilla Transformer decoder.

4 Experiment

Datasets. We tune our models primarily on 15 languages that appear in IWPT 2020 dataset and IWPT

	IWPT 2021	bg	cs	en	et	fi	fr	it	lt	lv	nl	pl	ru	sk	sv	uk	avg
ELAS	**BiAtt**	92.7	91.0	87.2	87.2	90.6	88.4	92.1	81.9	88.3	90.5	90.2	93.2	91.5	87.3	89.1	89.4
	Tree-G	92.8	**91.1**	87.3	87.1	**90.7**	88.6	92.3	81.9	88.2	90.5	**90.4**	93.2	91.6	87.5	89.0	89.5
	SAGER	**92.9**	90.9	**87.9**	**87.3**	**90.7**	**89.5**	**92.8**	**83.5**	**88.5**	**90.9**	**90.4**	**93.5**	**92.1**	**87.9**	**89.6**	**89.9**
GMS	**BiAtt**	47.4	44.8	36.3	37.1	38.7	40.3	43.8	21.0	38.4	46.9	40.8	50.3	51.0	32.2	34.6	40.2
	Tree-G	47.8	45.3	36.9	37.0	39.1	41.1	44.7	21.0	37.9	47.0	41.9	50.8	51.5	33.4	34.2	40.6
	SAGER	**48.8**	**45.6**	**40.3**	**39.2**	**41.4**	**45.4**	**47.1**	**28.2**	**42.8**	**51.3**	**43.6**	**54.2**	**57.8**	**38.2**	**39.2**	**44.2**
	IWPT 2020	**bg**	**cs**	**en**	**et**	**fi**	**fr**	**it**	**lt**	**lv**	**nl**	**pl**	**ru**	**sk**	**sv**	**uk**	**avg**
ELAS	**Sec-order**	91.5	90.1	87.1	86.0	89.0	85.3	91.5	78.9	87.6	86.2	84.0	92.3	87.6	84.7	88.0	87.3
	UDify	90.7	87.5	87.2	84.5	89.5	85.9	91.5	77.6	84.9	84.7	84.6	90.7	88.6	85.6	87.2	86.7
	SAGER	**92.6**	**90.4**	**88.2**	**86.9**	**90.1**	**87.4**	**92.6**	**82.5**	**88.5**	**86.7**	**86.7**	**93.2**	**91.0**	**87.0**	**89.0**	**88.9**
GMS	**Sec-order**	43.1	37.7	35.7	31.8	34.4	29.2	44.4	15.1	35.3	31.0	28.6	47.1	38.7	26.5	30.5	33.9
	UDify	41.4	31.4	34.1	31.2	34.5	33.2	41.5	17.8	31.6	23.8	26.1	40.6	43.1	27.1	31.2	32.6
	SAGER	**48.3**	**43.5**	**41.6**	**36.6**	**38.4**	**38.7**	**47.1**	**25.6**	**42.0**	**34.1**	**32.9**	**53.7**	**55.2**	**36.4**	**38.3**	**40.8**

Table 1: Average ELAS and GMS results of 3 calculations on IWPT 2021 and IWPT 2020 datasets. We use $L = 2$ according to ELAS on the English dev-set.

2021 dataset (Bouma et al., 2020, 2021). The two shared tasks focus on EUD (Schuster and Manning, 2016) which are non-tree graphs with reentrancies, empty nodes and sparsity cycles. To construct the topological hierarchy, we need to delete the back edges in cycles firstly and add them back by rules at inference time. For the language that has multiple treebanks, we simply concatenate all of its treebanks. Besides, we use gold tokenization and gold sentence segmentation during training and development. At test time, we use the results of tokenization and segmentation provided by the top ranked models.

Baseline Models. Our comparison experiments aim to investigate the performances of models themselves, without considering some learning techniques like ensembling (Grünewald et al., 2021), two-stage training (Shi and Lee, 2021) and automated concatenation of embeddings (Wang et al., 2021). We conclude four strong baselines from top-ranked systems in IWPT 2021 and IWPT 2020, namely Deep Biaffine Attention (Dozat and Manning, 2017), Tree-Graph Parser (Shi and Lee, 2021), Second-order Parser (Wang et al., 2019, 2020) and Language-specific UDify (Kondratyuk and Straka, 2019; Kanerva et al., 2020). Their results are reported after eliminating the effects of learning techniques.

Evaluation Metrics. ELAS results are evaluated by the official script provided by IWPT 2021. Besides, we also define a graph-level matching score (GMS) to investigate whether the model can deal with a few arcs that are difficult to be predicted properly in a sample. Since we segment UD sentences from raw texts, the numbers of sentences are different for each system. Therefore, GMS is an F_1 score around the number of absolutely matched graphs,

$$Recall = \frac{\#matched\ graph}{\#gold\ sentences}$$

$$Precision = \frac{\#matched\ graph}{\#system\ sentences}$$

$$GMS = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

where $\#$ represents $The\ number\ of$.

Word Embeddings. Similar to the operations in most top-ranked systems, our word embeddings $\mathbf{S}_{[n]}$ are initialized as the weighted summation of the corresponding hidden states in XLM-R layers (Conneau et al., 2020), where the weights are the learned attention distribution over all XLM-R layers. For the word composed of multiple subwords, we extract the hidden states of the last one. We set up the dimension in the graph representation model as $d = 1024$, the same as that in the pre-trained models.

Other Details. We train models directly on each language with Teacher Forcing Training to output all head (dependent) representations at once. Besides, we truncate the input sentences to 100 words at training time. We totally run 100 epochs with 16 batch size and select the model parameters base on the ELAS on the development sets. We train our models on a single v100 with a speed of about 10000 samples in 10 minutes. We use ReZero (Bachlechner et al., 2021) in our graph transformer, instead of LayerNorm operations commonly used in Transformer. In this case, we do not need to use the warm-up learning schedule, and we use Adam optimizer with the 0.97 decay ratio of the learning rate. We set up the initial learning rate of pre-trained embeddings as $2e-5$, and that of others as $1e-3$. Besides, dropout rates in the part of pre-training and graph representation are set to 0.1, while the output layers of nodes and edges are set to 0.3. We build up the vocabulary on the arc labels for each language respectively. To shrink the size of edge vocabularies, we follow the delexicalization operations of arc labels (Grünewald and Friedrich, 2020) and re-lexicalize them before evaluations.

5 Results and Analysis

5.1 Main Results

The official evaluation metrics ELAS of our models and baselines are shown in Table 1. We note that SAGER achieves at least comparable results on all languages. In IWPT 2021, in addition to obtaining the best average ELAS performance (average $\sim 0.4\%$ points), our model brings significant improvements over multiple languages like Lithuanian ($\sim 1.6\%$ points), French ($\sim 0.9\%$ points), English ($\sim 0.6\%$ points). This enhancement is more significant when comparing our model with the top two models in IWPT 2020 (average $\sim 1.6\%$ points). Besides, sharper increases appears in GMS of IWPT 2021 (average $\sim 3.6\%$ points) and IWPT 2020 (average $\sim 6.9\%$ points), where our model achieves an amazing rising against the baselines in all languages.

It should be mentioned that a higher ELAS does not mean a higher GMS, as shown in the results of Czech (*cs*) language. In other words, some obstinate errors are fixed to make more dependency graphs completely correct, but there appear some samples where more mistakes concentrate. This situation derives from the inherent characteristics of autoregressive generation that the prediction accuracy at one certain step is heavily dependent on that at historical steps. In ideal states, the historical information can calibrate some obstinate errors by the learned dependencies. However, once deviation occurs in an intermediate step, it may lead to mistakes that should not have been made. This is the essential reason that autoregressive parsers are weaker than non-autoregressive parsers. By comparison, our semi-autoregressive parser mitigates the negative impact of this characteristic by removing some unreasonable dependency relationships, thus resulting in better performances in both ELAS and GMS.

5.2 Model Variant Ablation Studies

To investigate the importance of different model components and input features, we evaluated the following variations of our model.

A. Autoregressive generation with random orders. We impose random orders to the sibling nodes, so the model is converted to a fully-autoregressive generator. At each step, the model only generates a new node and its all incoming edges. The sibling nodes will be re-ordered after a training epoch.

B. Autoregressive generation with word orders. The sibling nodes are sorted by the positions of the node words in the sentence.

C. Combine random orders and word orders. The sibling nodes are firstly randomly sorted at the early stage of training and fixed to the word orders at later training.

D. No implicit edges. Without the implicit edges, the graph representation model is similar to GAT (Velickovic et al., 2018) but the messages are additionally enriched with the arc label information.

E. No implicit edges in the same hierarchy. We remove the implicit edges between nodes in the same topological hierarchy. In this case, each node only has the incoming arcs from the nodes in the previous hierarchies.

F. No explicit edges. We replace all explicit edges by implicit edges, which is equal to forcing the edge embedding matrix $\mathbf{U}$ to zeros.

G. No hierarchical positional encodings. In this case, the model would lose the sequential relationships between hierarchies and fail to locate nodes of different hierarchy.

The ablation results of 6 languages are summarized in Figure 3. We firstly focus on the fully-autoregressive variations, namely the model A, B and C. We can see that there are significant declines in performances when imposing orderings to sibling nodes, indicating that the autoregressive mode heavily suffers from exposure bias in terms of generation orderings. Besides, the extent of declines varies a lot in different languages, ranging from over 30% in the Slovak (*sk*) dataset and within 1% in the English (*en*) and Italian (*it*) datasets. This proves that the impact of imposed sorting is quite unstable.

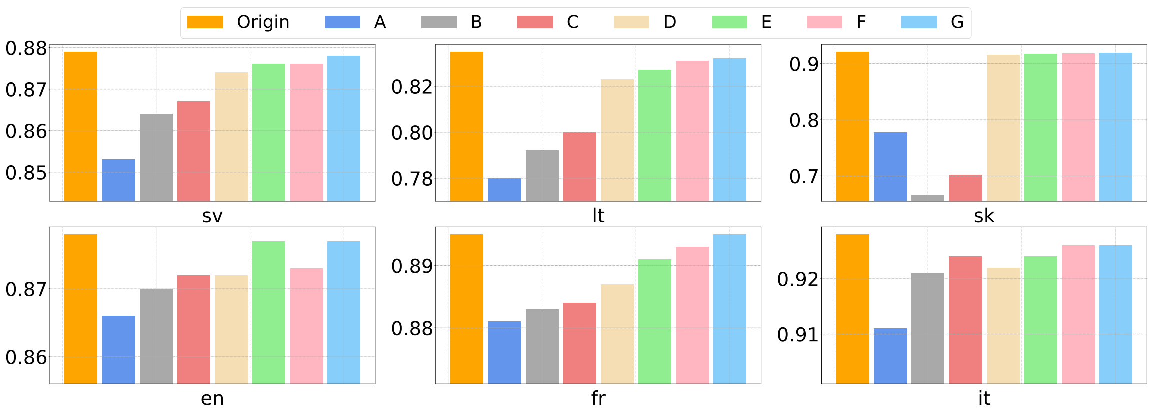

Figure 3: Test-set ELAS results, comparing the origin model with different model variations

Moving to the model D, E and F which are variations with respect to explicit and implicit edges. Although it is not as significant as the negative effects of using autoregressive modes, that of removing implicit or explicit edges cannot be ignored. Generally, implicit edges play a more important role than explicit edges as the performances of model D are often lower than those of model E and F. This validates that the edge sparsity is the major problem of graph generation after the uncertainty of generation orderings. Besides, the implicit edges in the same hierarchy (see model E) and the historical arc label information (see model F) are both compulsory model components because the model always performs worse when dropping them. The final is the model without hierarchical positional encodings. Compared with other variations, its performance is the closest to the origin model, implying that our graph representation model is not very sensitive to the sequential relationships between hierarchies.

5.3 Error analysis of Topological Hierarchy

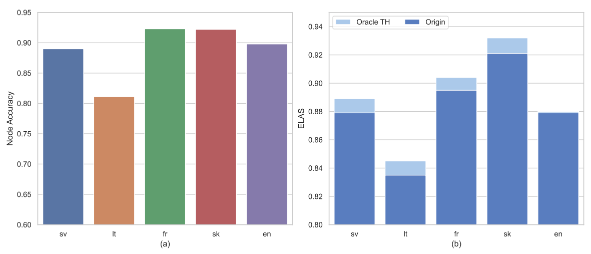

Figure 4: (a) Accuracy of nodes in the correct hierarchy. (b) ELAS results using oracle Topological Hierarchy (TH).

Since a topological hierarchy regulates the rough topological structure of a dependency graph, its prediction accuracy is crucial for the whole model. We investigate the node accuracy on 5 languages (see Figure 4.a), and find that about 90% nodes can fall into correct hierarchies. Even the language performing worst under the ELAS evaluation can reach 80% node accuracy. We then provide the model with the oracle node group at each generation step and plot the comparison results against the origin model in Figure 4.b. There is about a 1% increase of ELAS on most languages when using oracle TH. It is surprising that oracle TH does not bring about improvement to the English dataset, indicating that corrupt topological hierarchies do not always lead to incorrect arcs. Actually, it would cause bad results only when the dependent node of an arc is generated before its head nodes. It does not matter for corruptions that do not shuffle the orders of heads and dependents.

5.4 Sensitive analysis of Layers

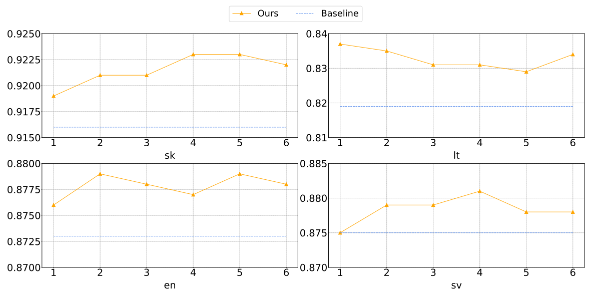

Figure 5: Sensitive analysis of layers on test sets.

We test the sensitivity of ELAS results to different L. As shown in Figure 5, We select four languages whose ELAS are significantly higher than baselines' when $L = 2$. We find that our model still outperforms the baseline whichever L is used. Besides, it is hard to disclose a trend between model performance and the number of layers from the four plots. This is possibly because graph transformer can capture context information of high-order neighbours even with one layer. Overall, SAGER is insensitive to the number of layers.

6 Case Study

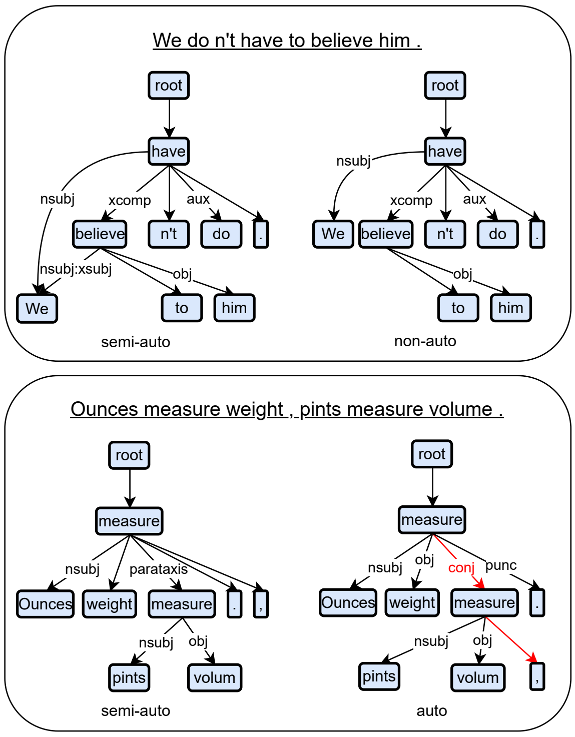

Figure 6: Case study, auto is the model B in § 5.2.

To better understand the drawbacks of non-autoregressive and autoregressive models, we present two examples in Figure 6. In the first example, semi-auto performs better because it can predict the arc *nsubj:xsubj* with the help of the previously predicted arc *xcomp*. The two edges are heavily dependent. In contrast, non-auto fails because it assumes independency between edges. In the graph produced by the auto, the edge between the first *measure* and the comma should have been predicted after linking *measure* to *weight*, but the model actually skips the step and predicts a wrong arc *conj*. This is probably because *(nsubj, obj, conj)* is a very highly-frequent dependent relationship in autoregressive graph sequences.

7 Conclusion and Limitation

This paper explores a semi-autoregressive dependency graph parser (SAGER) that learns the explicit dependencies in dependency graphs. This generation pattern captures the edge dependencies while reducing exposure bias, resulting in a more effective parser and two insights about graph generation, namely the ordering uncertainty and edge sparsity. We believe that the proposed algorithm can be applied in other language parsing problems such as OIA (Open Information Annotation) (Sun et al., 2020; Wang et al., 2022) and graph generation problems (Sun and Li, 2019).

The limitations of our work are inference speed and decoding strategy. The efficiency of semi-autoregressive inference is lower than that of non-autoregressive algorithms, so currently it cannot be applied to scenarios with high frequency requests. Furthermore, this paper only introduces greedy search as a decoding strategy. Overcoming the challenge of introducing many complex decoding strategies such as beam/tree search (Liu et al., 2020; Ma et al., 2021) belongs to our future work.

References

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Gary Cottrell, and Julian J. McAuley. 2021. ReZero is all you need: fast convergence at large depth. In Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI), pages 1352–1361, Virtual Event.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI), pages 12564–12573, Virtual Event.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2020. Overview of the IWPT 2020 shared task on parsing into enhanced universal dependencies. In Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 151–161, Online.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2021. From raw text to enhanced Universal Dependencies: The parsing shared task at IWPT 2021. In Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 146–157, Online.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. 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 3797–3807, Hong Kong, China.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1290–1301, Online.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 740–750, Doha, Qatar.

Xiaohui Chen, Xu Han, Jiajing Hu, Francisco J. R. Ruiz, and Li-Ping Liu. 2021. Order matters: Probabilistic modeling of node sequence for graph generation. In Proceedings of the 38th International Conference on Machine Learning (ICML), pages 1630–1639, Virtual Event.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2204–2214, Austin, TX.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), pages 8440–8451, Online.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France.

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 (NAACL-HLT), pages 710–716, Minneapolis, MN.

Stefan Grünewald and Annemarie Friedrich. 2020. Robertnlp at the IWPT 2020 shared task: Surprisingly simple enhanced UD parsing for english. In Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 245–252, Online.

Stefan Grünewald, Frederik Tobias Oertel, and Annemarie Friedrich. 2021. RobertNLP at the IWPT 2021 shared task: Simple enhanced UD parsing for 17 languages. In Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), Online.

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL), pages 2475–2485, Florence, Italy.

Jenna Kanerva, Filip Ginter, and Sampo Pyysalo. 2020. Turku enhanced parser pipeline: From raw text to enhanced graphs in the IWPT 2020 shared task. In Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 162–173, Online.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree LSTMs. Trans. Assoc. Comput. Linguistics, 4:445–461.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France.

Daniel Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. 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 2779–2795, Hong Kong, China.

Zuchao Li, Hai Zhao, and Kevin Parnow. 2020. Global greedy dependency parsing. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI), pages 8319–8326, New York, NY.

Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, and Ping Li. 2020. Extracting knowledge from web text with monte carlo tree search. In Proceedings of the Web Conference (WWW), pages 2585–2591, Taipei.

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

Ye Ma, Zixun Lan, Lu Zong, and Kaizhu Huang. 2021. Global-aware beam search for neural abstractive summarization. In Advances in Neural Information Processing Systems (NeurIPS), pages 16545–16557, virtual.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In Proceedings of the Tenth

International Conference on Language Resources and Evaluation (LREC), Portorož, Slovenia.

Tianze Shi and Lillian Lee. 2021. TGIF: Tree-graph integrated-format parser for enhanced UD with two-stage generic- to individual-language finetuning. In Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 213–224, Online.

Mingming Sun, Wenyue Hua, Zoey Liu, Xin Wang, Kangjie Zheng, and Ping Li. 2020. A Predicate-Function-Argument Annotation of Natural Language for Open-Domain Information eXpression. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2140–2150, Online.

Mingming Sun and Ping Li. 2019. Graph to graph: a topology aware approach for graph structures learning and generation. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), pages 2946–2955, Naha, Okinawa, Japan.

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 (NIPS), pages 5998–6008, Long Beach, CA.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, Canada.

Xin Wang, Minlong Peng, Mingming Sun, and Ping Li. 2022. OIE@OIA: an adaptable and efficient open information extraction framework. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL), pages 6213–6226, Dublin, Ireland.

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

Xinyu Wang, Zixia Jia, Yong Jiang, and Kewei Tu. 2021. Enhanced Universal Dependency parsing with automated concatenation of embeddings. In Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 189–195, Online.

Xinyu Wang, Yong Jiang, and Kewei Tu. 2020. Enhanced universal dependency parsing with second-order inference and mixture of training data. In Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT), pages 215–220, Online.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL), pages 323–333, Beijing, China.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In Advances in Neural Information Processing Systems (NeurIPS), pages 28877–28888, virtual.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL), pages 80–94, Florence, Italy.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. 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 3784–3796, Hong Kong, China.

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 (ACL), pages 3295–3305, Online.