Semantic Dependency Parsing with Edge GNNs

Songlin Yang, Kewei Tu*
School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
{yangsl,tukw}@shanghaitech.edu.cn

Abstract

Second-order neural parsers have obtained high accuracy in semantic dependency parsing. Inspired by the factor graph representation of second-order parsing, we propose edge graph neural networks (E-GNNs). In an E-GNN, each node corresponds to a dependency edge, and the neighbors are defined in terms of sibling, co-parent, and grandparent relationships. We conduct experiments on SemEval 2015 Task 18 English datasets, showing the superior performance of E-GNNs [1].

1 Introduction

Traditional syntactic dependency parsing aims to produce a tree structure for a given sentence, which has been well-studied. However, tree-structured representation is ill-suited for producing meaning representation, which motivates the proposal of semantic dependency parsing (SDP) (Oepen et al., 2014). SDP aims to produce a directed acyclic graph (DAG) instead of a tree to enable representing more complex semantic relationships.

Graph-based methods have obtained high accuracy in SDP (Peng et al., 2017; Dozat and Manning, 2018; Wang et al., 2019). Notably, Wang et al. (2019) propose a second-order neural CRF parser and show superior performance compared to the first-order Biaffine Parser (Dozat and Manning, 2018). To optimize the intractable CRF objective, they leverage approximate inference algorithms such as loopy belief propagation (LBP), unrolling several inference steps as recurrent neural networks (Zheng et al., 2015) for end-to-end approximation-aware training (Gormley et al., 2015). However, their model suffers from the following two problems: (i) Second-order dependency parsing can be formulated in terms of factor graphs (Smith and Eisner, 2008). However, the corresponding factor graph for second-order parsing is highly loopy (Fig. 1). It is known that on loopy graphs, LBP can easily get stuck on bad local optima, leading to sub-optimal results and thus undermining the parsing performance. (ii) First-order and second-order scores are produced based solely on contextualized word representations, which is deemed to be sub-optimal (Gan et al., 2022).

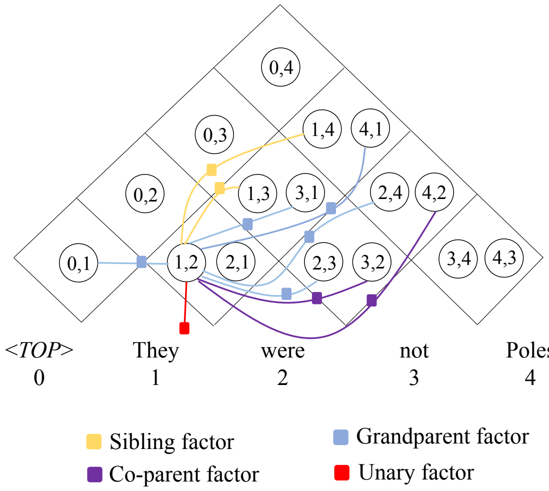


Figure 1: Factor graph representation of second-order semantic dependency parsing. The figure is plotted by Wang et al. (2019). Edge node (i, j) represents an edge from x_i to x_j. Three kinds of second-order relationships: sibling, co-parent, and grandparent are shown in the figure.

In this work, we propose edge GNNs (E-GNNs) to address the aforementioned limitation of Wang et al. (2019). Inspired by factor graph representations of second-order SDP where each variable node corresponds to a dependency edge (Fig. 1), we take edges as GNN nodes and define neighbors in terms of sibling, co-parent, and grandparent relationships. The benefit is shown as follows.

*Corresponding Author
[1] Our code is publicly available at https://github.com/sustcsonglin/gnn-sdp.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6096–6102
December 7-11, 2022 ©2022 Association for Computational Linguistics

(i) Previous work suggests that GNNs outperform LBP on loopy graphs (Yoon et al., 2018; Satorras and Welling, 2021). Thus we can expect that using E-GNNs will improve the inference quality and thereby result in better parsing performance. (ii) E-GNNs are more expressive since they incorporate edge-level features instead of just word-level features as in Wang et al. (2019). Edge nodes propagate features among neighbors via GNN layers, iteratively refining their representations to be more context-aware, and thereby capturing more information regarding long-range dependencies, which is shown experimentally.

We conduct experiments on SemEval 2015 Task 18 English datasets of SDP, showing superior performance compared with Wang et al. (2019).

2 Model

Word representations. Given a sentence $w = w_0 \cdots w_n$ (w_0 is the root), we feed it into BERT (Devlin et al., 2019) to obtain contextualized word embeddings and apply mean-pooling to the last layer of BERT to obtain word-level embedding $c = c_0 \cdots c_n$. We concatenate c with POS tag and lemma embeddings

$$e_i = c_i \oplus e_i^{\text{pos}} \oplus e_i^{\text{lemma}}$$

and then feed $e_0 \cdots e_n$ into a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) (BiLSTM):

$$\ldots, (\overrightarrow{b_i}, \overleftarrow{b_i}), \cdots = \text{BiLSTM}([\ldots, e_i, \ldots])$$

The final word representation is $x_i = \overrightarrow{b_i} \oplus \overleftarrow{b_i}$

Initial edge representation. To obtain initial edge representations, we adopt low-rank bilinear pooling (Kim et al., 2017) in order to capture pairwise interactions of parent and child word representations:

$$e_{ij}^0 = U(\sigma(Vx_i) \circ \sigma(Wx_j))$$

where σ is the activation function and we choose `tanh` in this work; $\circ$ is Hadamard (element-wise) product; U, W, V are linear layers (bias terms are omit for brevity).

E-GNN encoding. Inspired by second-order parsing, we take edges as GNN nodes and define neighbors in terms of sibling (sib), co-parent (cop), grand-parent (grd) relationships (Fig. 1). Since grandparent relationship is not symmetric, we consider both grandpa (grdp) and grandson (grds) relationships. We define $rel(i, j)$, the neighbor set[2] of edge (i, j) with respect to relationship $rel \in \{sib, cop, grdp, grds\}$ as follows:

$$\begin{aligned} sib(i,j) &:= \{(i,k)\}_k \quad & cop(i,j) &:= \{(k,j)\}_k \\ grdp(i,j) &:= \{(k,i)\}_k \quad & grds(i,j) &:= \{(j,k)\}_k \end{aligned}$$

For each rel, we use a deep biaffine scoring function (Dozat and Manning, 2017) to compute the un-normalized attention scores from edge (i, j) to its neighbor $(m, n) \in rel(i, j)$:

$$\begin{aligned} e_{ij}^{rel,a/b} &= MLP^{rel,a/b}(e_{ij}) \\ s_{ij,mn}^{rel} &= [e_{ij}^{rel,a}; 1]^T W^{rel} [e_{mn}^{rel,b}; 1]; \end{aligned}$$

Note that we do not need to compute scores for every pair of (i, j) and (m, n), which needs $\mathcal{O}(n^4)$ time. We only need to compute scores for adjacent edges under specific relationship and thereby need only $\mathcal{O}(n^3)$ time. The normalized attention scores for each relation types are computed as follows:

$$\begin{aligned} \alpha_{ij,ik}^{sib} &= \frac{\exp\{s_{ij,ik}^{sib}\}}{\sum_{k'} \exp\{s_{ij,ik'}^{sib}\}} \\ \alpha_{ij,kj}^{cop} &= \frac{\exp\{s_{ij,kj}^{cop}\}}{\sum_{k'} \exp\{s_{ij,k'j}^{cop}\}} \\ \alpha_{ij,jk}^{grds} &= \frac{\exp\{s_{ij,jk}^{grds}\}}{\sum_{k'} \exp\{s_{ij,jk'}^{grds}\}} \\ \alpha_{ij,ki}^{grdp} &= \frac{\exp\{s_{ij,ki}^{grdp}\}}{\sum_{k'} \exp\{s_{ij,k'i}^{grdp}\}} \end{aligned}$$

We compute the feature aggregated from neighbors as:

$$\begin{aligned} t_{ij}^m = \sum_k (&\alpha_{ij,ik}^{sib} e_{ik}^{m-1} + \alpha_{ij,kj}^{cop} e_{kj}^{m-1} \\ &+ \alpha_{ij,jk}^{grds} e_{jk}^{m-1} + \alpha_{ij,ki}^{grdp} e_{ki}^{m-1}) \end{aligned}$$

Next, we update GNN node representations based on their last iteration's representations and the aggregated feature:

$$e_{ij}^m = \text{ReLU}(\text{Linear}(e_{ij}^{m-1}) + t_{ij}^m)$$

[2]We include the edge itself in its neighbor set because during E-GNN computation, an edge may not want to attend to any neighbors and in that case it can put most of the attention weight on itself.

Parser	DM ID	DM OOD	PAS ID	PAS OOD	PSD ID	PSD OOD	Avg ID	Avg OOD
Dozat and Manning (2017) **+char+lemma**	93.7	88.9	93.9	90.6	81.0	79.4	89.5	86.3
Kurita and Søgaard (2019) **+lemma**	92.0	87.2	92.8	88.8	79.3	77.7	88.0	84.6
Wang et al. (2019) (MF) **+char+lemma**	94.0	89.7	94.1	91.3	81.4	79.6	89.8	86.9
Wang et al. (2019) (LBP) **+char+lemma**	93.9	89.5	94.2	91.3	81.4	79.5	89.8	86.8
Pointer **+char+lemma**	93.9	89.6	94.2	91.2	81.8	79.8	90.0	86.9
Zhang et al. (2019) +char+**BERT**$_{\text{LARGE}}$	92.2	87.1	-	-	-	-	-	-
Lindemann et al. (2019) **+BERT**$_{\text{BASE}}$	94.1	90.5	94.7	92.8	82.1	81.6	90.3	88.3
Lindemann et al. (2020) **+BERT**$_{\text{LARGE}}$	93.9	90.4	94.7	92.7	81.9	81.6	90.2	88.2
Pointer +char+lemma+**BERT**$_{\text{BASE}}$	94.4	91.0	95.1	93.4	82.6	82.0	90.7	88.8
LBP† (baseline) +lemma+**BERT**$_{\text{BASE}}$	94.9	91.7	95.2	93.5	82.6	82.3	90.9	89.2
E-GNN (ours) +lemma+**BERT**$_{\text{BASE}}$	**95.0**	**92.0**	**95.5**	**93.9**	**82.9**	**82.4**	**91.1**	**89.4**

Table 1: Labeled F1 scores on three formalisms of SemEval 2015 Task 18. $+char$ and $+lemma$ means using character and lemma embeddings. Pointer: Fernández-González and Gómez-Rodríguez (2020). †: our re-implementation.

Training loss. After l iterations of GNN update, we obtain e^l_{ij} for each edge. We use an MLP to map e^l_{ij} into a q-dimensional vector d_{ij}, where q is the label set size (including the special `NULL` label). We can associate each edge with a label index, which is either the index of `NULL` if the edge does not exist in the gold SDP graph, or the index of the gold edge label. We denote this label index as $y^{\star}_{ij}$. Then we use cross-entropy to compute the loss:

$$L = -\sum_{ij} \log \frac{\exp\{d_{ij}(y^{\star}_{ij})\}}{\sum_y \exp\{d_{ij}(y)\}}$$

3 Experiment

3.1 Setup

We conduct experiments on the SemEval 2015 Task 18 English datasets (Oepen et al., 2015). Sentences in the datasets are annotated with three formalisms: DM (Flickinger et al., 2012), PAS (Miyao and Tsujii, 2004), and PSD (Hajič et al., 2012). We use the standard data splitting as used in previous works (Martins and Almeida, 2014; Du et al., 2015), which contains 33,964 sentences in the training set, 1,692 sentences in the development set, 1,410 sentences in the in-domain (ID) test set and 1,849 sentences in the out-of-domain (OOD) test set from the Brown Corpus (Francis and Kucera, 1982). We use `bert-base-cased` (Devlin et al., 2019) to obtain contextualized word embedding. The number of GNN layers is set to 3. Other hyperparameters can be found in App. A. We report the labeled F1 scores (LF1) in the ID and OOD test sets for each formalism. The reported results are averaged over three runs with different random seeds.

3.2 Main result

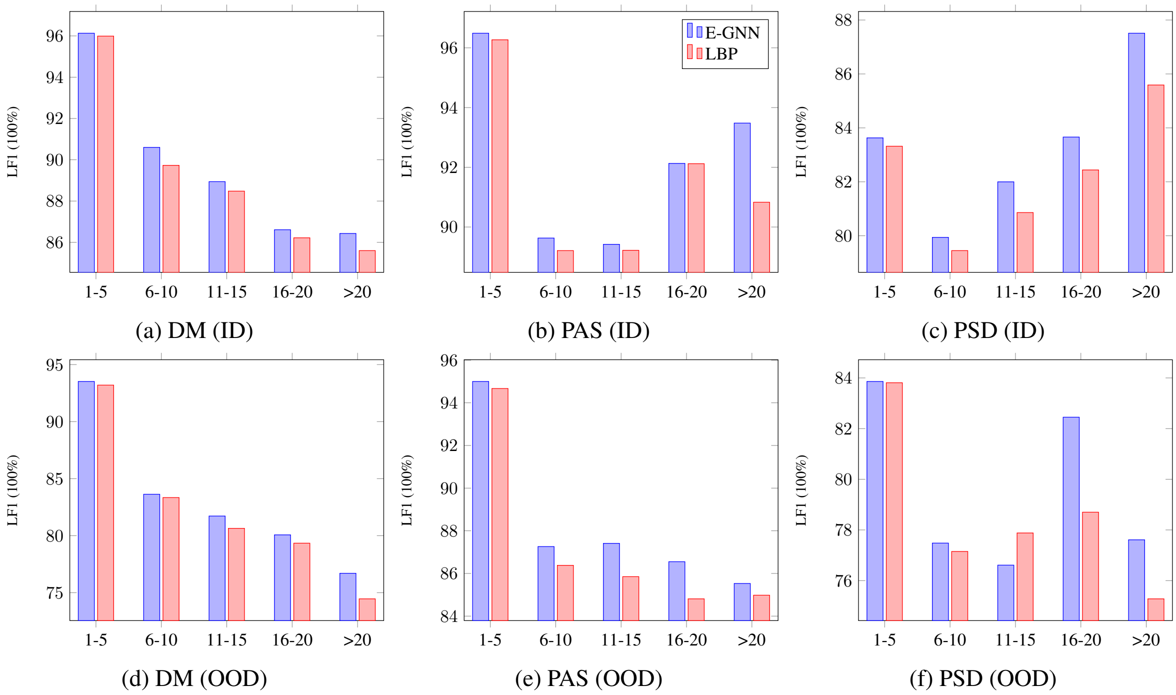
Table 1 shows the experimental results. We reimplement the LBP-based second-order parser of Wang et al. (2019) as the baseline (LBP hereafter for short), using the same neural encoder and the same settings (e.g., hyper-parameters) as E-GNN for fair comparison. As we can see, LBP has already surpassed *Pointer* (Fernández-González and Gómez-Rodríguez, 2020), a strong model, by 0.2 and 0.4 average F1 scores in ID test sets and OOD test sets. E-GNN further outperforms LBP by 0.2 average F1 scores on both ID and OOD test datasets.

3.3 Ablation study

We conduct ablation studies on PAS. First, we study the importance of using different relationship types to define neighbors in GNNs. As we can see from Table 2, removing `sib`/`cop`/`grd` (both `grdp` and `grds`) results in 0.17, 0.20, 0.18 LF1 score drops, respectively, showing that all these relationships are beneficial to the final performance, which is consistent with the intuition in second-order parsing. Second, we conduct an ablation study on the effect of the number of GNN layers. Table 2 shows that using 0/1/2 layers leads to 0.32/0.18/0.15 F1 score drops, validating the importance of using a three-layer E-GNN.

3.4 Error analysis

Fig. 2 shows the change of LF1 scores with the length of dependency edges. We can see that when



(a) DM (ID) (b) PAS (ID) (c) PSD (ID)
(d) DM (OOD) (e) PAS (OOD) (f) PSD (OOD)

Figure 2: LF1 of different edge lengths on three semantic formalisms.

Model	LF
E-GNN	**95.47**
w/o sib	95.30
w/o cop	95.27
w/o grd	95.29
#GNN layer=2	95.32
#GNN layer=1	95.29
#GNN layer=0	95.15

Table 2: Ablation study on PAS test id. set.

the edge length is small (1-5), LBP and E-GNN have almost identical performance. However, when the edge length is large (>10), E-GNN outperforms LBP by a large margin, especially when the edge length is larger than 20. We hypothesize that E-GNN can model long-range dependencies more effectively. Neural encoders such as BiLSTMs have difficulty in capturing long-range dependencies, so relying solely on word representations to produce first/second-order edge scores, as in Wang et al. (2019), would have difficulty in predicting long edges. In comparison, for E-GNN, although the initial edge representation may also have difficulty in capturing long-range dependencies, during iterative GNN update, a long edge can gather information from all its neighbors, refining its representation to be more context-aware and thus capturing more long-range information.

4 Related work

Dependency parsing with GNNs. Ji et al. (2019) used GNNs for dependency parsing. However, they view words instead of edges as GNN nodes. Consequently, it is tricky to define neighbors and thus tricky to design node vector update schemes. In our model, we view edges as GNN nodes, so we can define neighbors and design node vector update schemes more naturally by following second-order dependency relationships. In addition, our model captures edge-level features and thus is more expressive.

Algorithmic alignment. One can view the GNN layers of our model as a learnable inference decoder, which mimics the behavior of LBP. Xu et al. (2020) propose the concept of algorithmic alignment, finding that neural networks—whose structures resembling classical algorithms for certain problems—are easier to train and have better performance. The design of our model follows the principle of algorithmic alignment, as E-GNN nodes resemble variable nodes in the factor graph of second-order parsing, and the message passing mechanism of the GNN resembles LBP inference steps. We can find other successful models complying with the algorithmic alignment principle in the field of NLP. Taking DIORA (Drozdov et al., 2019) for example, it mimics the classical inside-outside algorithm to design the network and achieves good performance in unsupervised constituency parsing.

5 Conclusion

We proposed E-GNNs in the spirit of the factor graph representation of second-order dependency parsing. Experiments and ablation studies on SemEval 2015 Task 18 English datasets of SDP validated the effectiveness of E-GNNs.

Limitations

E-GNN needs $\mathcal{O}(n^3)$ time to update edge representations in each GNN layer, while the Biaffine Parser only needs $\mathcal{O}(n^2)$ time to score all edges. Besides, E-GNN needs to store $\mathcal{O}(n^2)$ edge embeddings in each GNN layer, consuming more GPU memories than the Biaffine Parser.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61976139).

References

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.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. 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 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 927–931, Denver, Colorado. 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.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank. a dynamically annotated treebank of the wall street journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–96.

Winthrop Nelson Francis and Henry Kucera. 1982. *Frequency analysis of English usage: Lexicon and usage*. Houghton Mifflin.

Leilei Gan, Yuxian Meng, Kun Kuang, Xiaofei Sun, Chun Fan, Fei Wu, and Jiwei Li. 2022. Dependency parsing as MRC-based span-span prediction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2427–2437, Dublin, Ireland. Association for Computational Linguistics.

Matthew R. Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics*, 3:489–501.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Jin-Hwa Kim, Kyoung Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Hadamard product for low-rank bilinear pooling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon,* France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2020. Fast semantic parsing with well-typedness guarantees. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3929–3951, Online. Association for Computational Linguistics.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476, Dublin, Ireland. Association for Computational Linguistics.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

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.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Victor Garcia Satorras and Max Welling. 2021. Neural enhanced belief propagation on factor graphs. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 685–693. PMLR.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 145–156, Honolulu, Hawaii. Association for Computational Linguistics.

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.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2020. What can neural networks reason about? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard S. Zemel, and Xaq Pitkow. 2018. Inference in probabilistic graphical models by graph neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. 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 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. 2015. Conditional random fields as recurrent neural networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1529–1537. IEEE Computer Society.

Architecture hyper-parameters	
POS/Lemma dimension	100
Embeddings dropout	0.33
BiLSTM encoder size	1000
BiLSTM layers dropout	0.33
MLP layers dropout	0.33
BiAffine hidden size	300
GNN hidden size	500
GNN layer	3
Training-related hyper-parameters	
BERT learning rate	5e-5
Other learning rate	1.5e-3
Optimizer	AdamW
Scheduler	linear warmup
Warmup rate	0.5
Gradient clipping	5.0
Tokens per batch	3000
Maximum training sentence length	150

Table 3: Summary of hyper-parameters.

A Hyperparameter details

The hyperparamter configuration is summarized in Table 3. Besides, the number of total training epoch is set to 30 for DM; 20 for PAS and PSD. The number of BiLSTM encoder layer is 1 for DM, and 2 for PAS and PSD.