Levi Graph AMR Parser using Heterogeneous Attention

Han He
Computer Science
Emory University
Atlanta GA 30322, USA
han.he@emory.edu

Jinho D. Choi
Computer Science
Emory University
Atlanta GA 30322, USA
jinho.choi@emory.edu

Abstract

Coupled with biaffine decoders, transformers have been effectively adapted to text-to-graph transduction and achieved state-of-the-art performance on AMR parsing. Many prior works, however, rely on the biaffine decoder for either or both arc and label predictions although most features used by the decoder may be learned by the transformer already. This paper presents a novel approach to AMR parsing by combining heterogeneous data (tokens, concepts, labels) as one input to a transformer to learn attention, and use only attention matrices from the transformer to predict all elements in AMR graphs (concepts, arcs, labels). Although our models [1] use significantly fewer parameters than the previous state-of-the-art graph parser, they show similar or better accuracy on AMR 2.0 and 3.0.

1 Introduction

Abstract Meaning Representation (AMR) has recently gained lots of interests due to its capability in capturing abstract concepts (Banarescu et al., 2013). In the form of directed acyclic graphs (DAGs), an AMR graph consists of nodes as concepts and edges as labeled relations. To build such a graph from plain text, a parser needs to predict concepts and relations in concord.

While significant research efforts have been conducted to improve concept and arc predictions, label prediction has been relatively stagnated. Most previous models have adapted the biaffine decoder for label prediction (Lyu and Titov, 2018; Zhang et al., 2019a; Cai and Lam, 2019; Zhou et al., 2020; Lindemann et al., 2020). These models assign labels from the biaffine decoder to arcs predicted by another decoder, which can be misled by incorrect arc predictions during decoding.

[1]Resources are publicly available at https://github.com/emorynlp/levi-graph-amr-parser.

The enhancement of message passing between decoders for arc and label predictions has shown to be effective. Among these works, Cai and Lam (2020) emerge with an iterative method to exchange embeddings between concept and arc predictions and feed the enhanced embeddings to the biaffine decoder for label prediction. While this approach greatly improves accuracy, it complicates the network architecture without structurally avoiding the error propagation from the arc prediction.

This paper presents an efficient transformer-based (Vaswani et al., 2017) approach that takes a mixture of tokens, concepts, and labels as inputs, and performs concept generation, arc prediction, and label prediction jointly using only attentions from the transformer without using a biaffine decoder. Its compact structure (§3.3) enables cross-attention between heterogeneous inputs, providing a complete view of the partially built graph and a better representation of the current parsing state. A novel Levi graph decoder (§3.4) is also proposed that reduces the number of decoder parameters by 45% (from 5.5 million to 3.0 million) yet gives similar or better performance. To the best of our knowledge, this is the first text-to-AMR graph parser that operates on the heterogeneous data and adapts no biaffine decoder.

2 Related Work

Recent AMR parsing approaches can be categorized into four classes: (i) transition-based parsing which casts the parsing process into a sequence of transitions defined on an abstract machine (e.g., a transition system using a buffer and a stack) (Wang et al., 2016; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Peng et al., 2017; Guo and Lu, 2018; Liu et al., 2018; Naseem et al., 2019; Fernandez Astudillo et al., 2020; Lee et al.,

Proceedings of the 17th International Conference on Parsing Technologies (IWPT 2021), pages 50–57
Bangkok, Thailand (online), August 6, 2021. ©2021 Association for Computational Linguistics

2020), (ii) seq2seq-based parsing [2] which transduces raw sentences into linearized AMR graphs in text form (Barzdins and Gosko, 2016; Konstas et al., 2017; van Noord and Bos, 2017; Peng et al., 2018; Xu et al., 2020; Bevilacqua et al., 2021), (iii) seq2graph-based parsing which incrementally and directly builds a semantic graph via expanding graph nodes without resorting to any transition system (Cai and Lam, 2019; Zhang et al., 2019b; Lyu et al., 2020). (iv) graph algebra parsing which translates an intermediate grammar structure into AMR (Artzi et al., 2015; Groschwitz et al., 2018; Lindemann et al., 2019, 2020).

Our work is most closely related to seq2graph paradigm while we extend the definition of node to accommodate relation labels in a Levi graph. We generate a Levi graph which is a linearized form originally used in seq2seq models for AMR-to-text (Beck et al., 2018; Guo et al., 2019; Ribeiro et al., 2019). Our Levi graph approach differs from seq2seq approaches in its attention based arc prediction, where arc is directly predicted by attention heads instead of brackets in the target sequence.

3 Approach

3.1 Text-to-Graph Transducer

Figure 1 shows the overview of our Text-to-Graph Transduction model. Let $W = \{w_0, w_1, \ldots, w_n\}$ be the input sequence where w_0 is a special token representing the target node and w_i is the i'th token. W is fed into a *Text Encoder* creating embeddings $\{e^w_0, e^w_1, \ldots, e^w_n\}$. In parallel, *NLP Tools* produce several features for w_i and pass them to a *Feature Encoder* to generate $\{e^f_0, e^f_1, \ldots, e^f_n\}$. Embeddings $\{e^w_i \oplus e^f_i : i \in [0, n]\}$ are put to a *Text Transformer*, which generates $E^t = \{e^t_0, e^t_1, \ldots, e^t_n\}$.[3]
Let $V = \{v_0, v_1, \ldots, v_m\}$ be the output sequence where v_0 is a special token representing the root and v_i is the i'th predicted node. V is fed into a *Graph Encoder* to create $E^v = \{e^v_0, e^v_1, \ldots, e^v_m\}$. Finally,

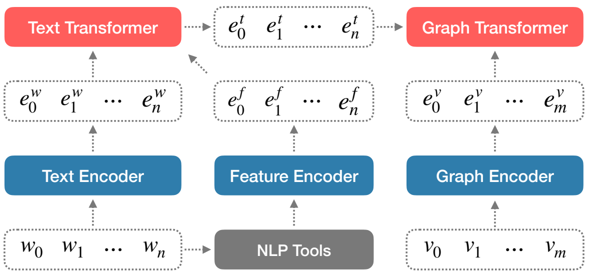


Figure 1: Overview of our Text-to-Graph Transducer.

E^t and E^v are fed into a *Graph Transformer* that predicts the target node as well as its relations to all nodes in V. The target node predicted by the *Graph Transformer* gets appended to V afterwards.[4]

3.2 Concept + Arc-Biaffine + Rel-Biaffine

Our first graph transformer generates $\{v_1, \ldots, v_m\}$ where v_i is a concept in the target graph, and predicts both arcs and labels using a biaffine decoder. Given E^t and E^v (§3.1), three matrices are created, $\mathcal{Q} = e^t_0 \in \mathbb{R}^{1\times d}, \mathcal{K}|\mathcal{V} = [e^t_1, .., e^t_n, e^v_0, e^v_1, .., e^v_m] \in \mathbb{R}^{k\times d}$ $(k = n+m+1)$. These matrices are put to multiple layers of multi-head attention (MHA) producing $\{\alpha^i : i \in [1, h]\}$ and $\{\beta^i : i \in [1, h]\}$ from the last layer, where h is the total number of heads in MHA ($\mathrm{W}^{\mathcal{Q}|\mathcal{K}|\mathcal{V}}_i \in \mathbb{R}^{d\times d}, \mathrm{W}^{\oplus} \in \mathbb{R}^{(h\cdot d)\times d}$):

$$\begin{aligned}
\alpha^i &= \mathrm{softmax}(\frac{(\mathcal{Q}\mathrm{W}^{\mathcal{Q}}_i)(\mathcal{K}\mathrm{W}^{\mathcal{K}}_i)^\top}{\sqrt{d}}) && \in \mathbb{R}^{1\times k} \\
\beta^i &= \alpha^i \cdot \mathcal{V} \cdot \mathrm{W}^{\mathcal{V}}_i && \in \mathbb{R}^{1\times d} \\
\alpha^{\oslash} &= [\alpha^1_j : j \in [1, n]] && \in \mathbb{R}^{1\times n} \\
\beta^{\oplus} &= (\beta^1 \oplus \ldots \oplus \beta^h) \cdot \mathrm{W}^{\oplus} && \in \mathbb{R}^{1\times d}
\end{aligned}$$

$\alpha^{\oslash}_j$ indicates the probability of w_j being aligned to the target node, and $\beta^{\oplus}$ is the embedding representing the node. Let C be the list of all concepts in training data and L be the list of lemmas for tokens in W such that $|W| = |L|$. Given $X = C^\frown W^\frown L$, $\alpha^{\oslash}$ and $\beta^{\oplus}$ are fed into a *Node Decoder* estimating the score of each $x_i \in X$ being the target node:

$$\begin{aligned}
g(C|W|L) &= \mathrm{softmax}(\beta^{\oplus} \cdot \mathrm{W}^{C|W|L}) \\
p(x_i) &= g(C) \cdot [\mathrm{softmax}(\beta^{\oplus} \cdot \mathrm{W}^{G})]_i \\
&\quad + g(W) \sum_{j\in W(x_i)} \alpha^{\oslash}_j + g(L) \sum_{j\in L(x_i)} \alpha^{\oslash}_j
\end{aligned}$$

$g(C|W|L)$ is the gate probability of the target node being in $C|W|L$, respectively ($\mathrm{W}^{C|W|L} \in \mathbb{R}^{d\times 1}$).

[2]Seq2seq-based parsing is sometimes categorized into "translation-based methods" (Koller et al., 2019) possibly due to the prevalence of seq2seq model in Neural Machine Translation, while we believe that translation refers more to the transduction between languages while AMR is neither a language nor an interlingua.

[3]In our case, BERT (Devlin et al., 2019) is used as the *Text Encoder* and $\forall_i . e^f_i = e^{\text{LEMMA}}_i \oplus e^{\text{POS}}_i \oplus e^{\text{NER}}_i \oplus e^{\text{CHAR}}_i$ is created by the *Feature Encoder* using predictions (lemmas, part-of-speech tags and named-entities) from the *NLP Tools* and character level features from a Convolutional Neural Network. In this work, we use CoreNLP (Manning et al., 2014) for a fair comparison with existing approaches.

[4]*Graph Encoder* creates $\forall_i . e^v_i = \mathrm{transformer}(e^{\text{NODE}}_i \oplus e^{\text{CHAR}}_i)$.

$p(x_i)$ is estimated by measuring the probabilities of x_i being the target if $x_i \in C$ ($\mathrm{W}^G \in \mathbb{R}^{d \times |C|}$), and if $x_i \in W|L$ where $W|L(x_i) = \{j : (x_i = y_j) \wedge y_j \in W|L\}$, respectively. Finally, the output layer $o^{\text{node}} = [p(x_i) : x_i \in X] \in \mathbb{R}^{1\times(|C|+|W|+|L|)}$ gets created and $\arg\max_{x_i}(o^{\text{node}})$ is taken as the target.

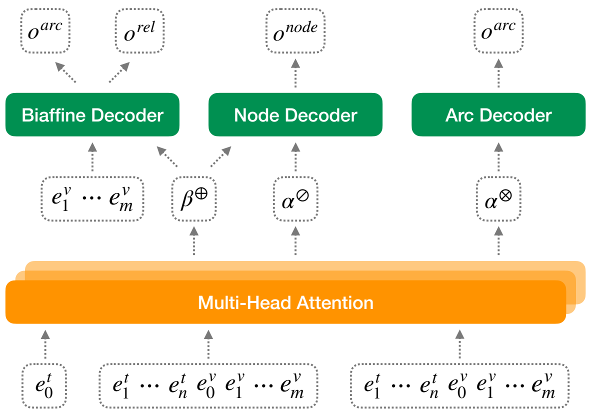


Figure 2: Overview of our Graph Transformer models. ND/BD/AD: node/biaffine/arc decoder. §3.2: ND for concept generation and BD for arc and label predictions; §3.3: ND for concept generation, AD for arc prediction, and BD for label prediction; §3.4: ND for concept and label generations and AD for arc prediction.

For arc and label predictions, the target embedding $\beta^{\oplus}$ is used to represent a head and the embeddings of previously predicted nodes, $\{e_1^v, \ldots, e_m^v\}$, are used to represent dependents in a *Biaffine Decoder*, which creates two output layers, $o^{\text{arc}} \in \mathbb{R}^{1\times m}$ and $o^{\text{rel}} \in \mathbb{R}^{1\times m\times|R|}$, to predict the target node being a head of the other nodes, where $|R|$ is the list of all labels in training data (Dozat and Manning, 2017).

3.3 Concept + Arc-Attention + Rel-Biaffine

Our second graph transformer is similar to the one in §3.2 except that it uses an *Arc Decoder* instead of the *Biaffine Decoder* for arc prediction. Given $A = \{\alpha^1, \ldots, \alpha^h\}$ in §3.2, $\alpha^{\otimes} \in \mathbb{R}^{1\times(m+1)}$ is created by first applying dimension-wise maxpooling to A and slicing the last $m+1$ dimensions as follows:

$$\alpha^{\otimes} = [\max(\alpha_j^1, \ldots, \alpha_j^h) : j \in [n+1, n+m+1]]$$

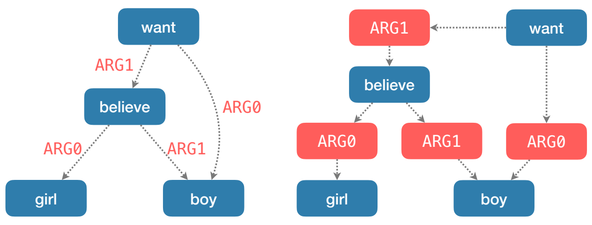
Notice that values in $\alpha^{\otimes}$ are derived from multiple heads; thus, they are not normalized. Each head is expected to learn different types of arcs. During decoding, any $v_i \in V$ whose $\alpha_i^{\otimes} \geq 0.5$ is predicted to be a dependent of the target node. During training, the negative log-likelihood of $\alpha^{\otimes}$ is optimized.[5]

[5]This model still uses the *Biaffine Decoder* for label prediction.

The target node, say v_t, may need to be predicted as a dependent of v_i, in which case, the dependency is reversed (so v_t becomes the head of v_i), and the label is concatenated with the special tag `_R` (e.g., $\texttt{ARG0}(v_i, v_t)$ becomes $\texttt{ARG0\_R}(v_t, v_i)$).

3.4 Levi Graph + Arc-Attention

Our last graph transformer uses the *Node Decoder* for both concept and label generations and the *Arc Decoder* for arc prediction. In this model, $v_i \in V'$ can be either a concept or a label such that the original AMR graph is transformed into the Levi graph (Levi, 1942; Beck et al., 2018) (Figure 3).



(a) AMR graph (b) Levi graph

Figure 3: AMR and Levi graphs for the input, "*The boy wants the girl to believe him*".

Unlike the node sequence containing only concepts in the AMR graph ordered by breadth-first traverse, used as the output sequence for the models in §3.2 and §3.3, the node sequence in this model is derived by inserting the label of each edge after head concept during training. This concepts-labels alternation has two advantages over a strict topological order: (i) it can handle erroneous cyclic graphs, (ii) it is easier to restore relations as each label is connected to its closest concept. The heterogeneous nature of node sequences from Levi graphs allows our Graph Transformer to learn attentions among 3 types of input, tokens, concepts, and labels, leading to more informed predictions.

Let V' be the output sequence consisting of both predicted concepts and labels. Let C' be the set of all concepts and labels in training data. Compared to V and C in §3.2, V' is about twice larger than V because every concept has one or more associated labels that indicate relations to its heads. However, C' is not so much larger than C because the addition from the labels is insignificant to the number of concepts that are already in C. By replacing $V|C$ with $V'|C'$ respectively, the *Node Decoder* in §3.2 can generate both concepts and labels. $\alpha^{\otimes}$ in §3.3 then gives attention scores among concepts and labels that can be used by the *Arc Decoder* to find arcs among them.

	SMATCH		Fine-grained Evaluation						
	Labeled	Unlabeled	No WSD	Concept	SRL	Reent.	Neg.	NER	Wiki
Lindemann et al. (2019)	75.3	-	-	-	-	-	-	-	-
Naseem et al. (2019)	75.5	80	76	86	72	56	67	83	80
Zhang et al. (2019a)	76.3	79.0	76.8	84.8	69.7	60.0	75.2	77.9	85.8
Zhang et al. (2019b)	77.0	80	78	86	71	61	77	79	86
Cai and Lam (2020)	80.2	82.8	80.8	88.1	74.2	64.6	78.9	81.1	86.3
Xu et al. (2020)†	80.2	83.7	80.8	87.4	78.9	66.5	71.5	85.4	75.1
Lee et al. (2020)‡	81.3	85.3	81.8	88.7	88.7	66.3	79.2	71.9	79.4
Bevilacqua et al. (2021)§	84.5	86.7	84.9	89.6	79.7	72.3	79.9	83.7	87.3
CL20	**80.0**±0.2	82.5±0.3	**80.5**±0.3	88.0±0.1	73.7±0.4	63.8±0.7	79.2±0.3	**81.1**±0.3	**86.2**±0.1
ND + BD + BD	79.4±0.1	82.3±0.1	80.0±0.2	87.9±0.2	73.1±0.2	62.5±0.2	**79.8**±0.3	80.7±1.0	85.8±0.5
ND + AD + BD	**80.0**±0.1	**82.6**±0.1	**80.5**±0.1	**88.2**±0.1	73.6±0.4	63.3±0.4	79.4±1.0	80.8±0.8	**86.2**±0.3
ND + AD + LV	**80.0**±0.1	82.2±0.2	**80.5**±0.1	87.7±0.2	**74.5**±0.2	**64.1**±0.3	78.4±1.0	80.5±0.8	**86.2**±0.3

(a) Results on AMR 2.0 results. Supervised†/unsupervised§ pre-training and self-learning‡ are orthogonal to our work.

	SMATCH		Fine-grained Evaluation						
	Labeled	Unlabeled	No WSD	Concept	SRL	Reent.	Neg.	NER	Wiki
Lyu et al. (2020)	75.8	-	-	**88.0**	72.6	-	-	-	-
CL20	76.8±0.2	79.9±0.2	77.3±0.2	86.3±0.2	73.2±0.2	63.4±0.2	72.3±1.4	73.0±0.5	79.5±0.2
ND + BD + BD	75.8±0.2	79.0±0.1	76.2±0.1	84.6±0.2	72.1±0.3	61.7±0.4	72.6±0.7	71.6±0.3	78.7±0.2
ND + AD + BD	76.8±0.1	**80.1±0.1**	77.3±0.1	86.5±0.2	73.1±0.2	**63.6±0.2**	**73.2±0.9**	73.0±0.2	**79.6±0.1**
ND + AD + LV	**77.0±0.2**	79.8±0.2	**77.5±0.2**	86.1±0.1	**73.6±0.3**	62.6±0.6	71.3±0.4	**73.3±0.7**	79.5±0.3

(b) Results on AMR 3.0.

Table 1: Averages ± standard deviations on AMR 2.0 and 3.0 . CL20: results by running the original implementation of Cai and Lam (2020) 3 times, ND+BD+BD: §3.2, ND+AD+BD: §3.3, ND+AD+LV: §3.4.

4 Experiments

4.1 Experimental Setup

All models are experimented on both the AMR 2.0 (LDC2017T10) and 3.0 datasets (LDC2020T02). AMR 2.0 has been well-explored by recent work, while AMR 3.0 is the latest release about 1.5 times larger than 2.0 that has not yet been explored much. The detailed data statistics are shown in Table A.1.2. The training, development, and test sets provided in the datasets are used, and performance is evaluated with the SMATCH (F1) (Cai and Knight, 2013) as well as fine-grained metrics (Damonte et al., 2017). The same pre- and post-processing suggested by Cai and Lam (2020) are adapted. Section A.2 gives the hyper-parameter configuration of our models.

4.2 Results

All our models are run three times and their averages and standard deviations are reported in Table 1. Compared to CL20 using 2 transformers to decode arcs & concepts then apply attention across them, our models use 1 transformer for the *Node Decoder* achieving both objectives simultaneously. All models except for ND+BD reaches the same SMATCH score of 80% on AMR 2.0. ND+AD+LV shows a slight improvement over the others on AMR 3.0, indicating that it has a greater potential to be robust with a larger dataset. Considering that this model uses about 3M fewer parameters than CL20, these results are promising. ND+BD+BD consistently shows the lowest scores, implying the significance of modeling concept generation and arc prediction coherently for structure learning. ND+AD+LV shows higher scores for SRL and Reent whereas the other models show advantage on Concept and NER on AMR 2.0, although the trend is not as noticeable on AMR 3.0, implying that the Levi graph helps parsing relations but not necessarily tagging concepts.

Case Study We study the effect of our proposed two improvements: heterogeneous Graph Transformer and Levi graph, from the view of attention in Figure 4. Figure 4a shows that the core verb "*wants*" is heavily attended by every token, suggesting that our Graph Transformer successfully grasps the core idea. Figure 4b presents the soft alignment between nodes and tokens, which surprisingly overweights " *boy*", "*girl*" and "*believe*" possibly due to their dominance of semantics. Figure 4c illustrates the arc prediction, which is a lower triangular matrix obtained by zeroing out the upper triangle of stacked $\alpha^{\otimes}$. Its diagonal suggests that self-loop is crucial for representing each node.

5 Conclusion

We presented two effective approaches which achieve comparable (or better) performance comparing with the state-of-the-art parsers with significantly fewer parameters. Our text-to-graph trans-

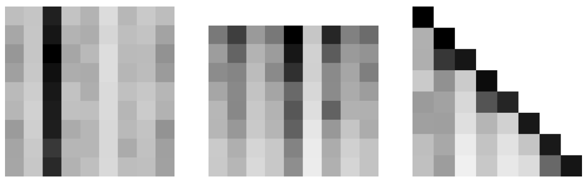

(a) Token⇆token (b) Node⇆token (c) Node⇆node

Figure 4: Self- and cross-attention for tokens "*The boy wants the girl to believe him*" and nodes "*want believe* `ARG1` *boy* `ARG1` `ARG0` *girl* `ARG0`".

ducer enables self- and cross-attention in one transformer, improving both concept and arc prediction. With a novel Levi graph formalism, our parser demostrates its advantage on relation labeling. An interesting future work is to preserve benefits from both approaches in one model. It is also noteworthy that our Levi graph parser can be applied to a broad range of labeled graph parsing tasks including dependency trees and many others.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147, San Diego, California. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

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 AAAI*.

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 3799–3809, Hong Kong, China. Association for Computational Linguistics.

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*, pages 1290–1301, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

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 *Proceedings of the 5th International Conference on Learning Representations*, ICLR'17.

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online. Association for Computational Linguistics.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1831–1841, Melbourne, Australia. Association for Computational Linguistics.

Zhijiang Guo and Wei Lu. 2018. Better transition-based AMR parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722, Brussels, Belgium. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Alexander Koller, Stephan Oepen, and Weiwei Sun. 2019. Graph-based meaning representations: Design and processing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 6–11, Florence, Italy. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Young-Suk Lee, Ramón Fernandez Astudillo, Tahira Naseem, Revanth Gangi Reddy, Radu Florian, and Salim Roukos. 2020. Pushing the limits of AMR parsing with self-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3208–3214, Online. Association for Computational Linguistics.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*. University of Calcutta.

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 welltypedness guarantees. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3929–3951, Online. Association for Computational Linguistics.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430, Brussels, Belgium. Association for Computational Linguistics.

Chunchuan Lyu, Shay B Cohen, and Ivan Titov. 2020. A differentiable relaxation of graph segmentation and alignment for amr parsing. *arXiv preprint arXiv:2010.12676*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose 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.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Xiaochang Peng, Linfeng Song, Daniel Gildea, and Giorgio Satta. 2018. Sequence-to-sequence models for cache transition systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1842–1852, Melbourne, Australia. Association for Computational Linguistics.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375, Valencia, Spain. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. 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 3183–3194, Hong Kong, China. Association for Computational Linguistics.

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

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178, San Diego, California. Association for Computational Linguistics.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. Improving AMR parsing with sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511, Online. Association for Computational Linguistics.

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

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 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020. AMR parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4306–4319, Online. Association for Computational Linguistics.

A Appendix

A.1 Datasets and Pre/Post-Processing

Table 2 describes statistics of the AMR 2.0[6] and the AMR 3.0[7] datasets used in our experiments.

	Sentences	Tokens	Concepts	Relations
TRN	36,521	624,750	422,655	426,712
DEV	1,368	27,713	19,890	20,111
TST	1,371	28,279	26,513	27,175

(a) AMR 2.0.

	Sentences	Tokens	Concepts	Relations
TRN	55,635	965,468	656,123	667,577
DEV	1,722	34,696	25,171	25,568
TST	1,898	37,225	34,903	35,572

(b) AMR 3.0.

Table 2: Statistics of AMR 2.0 and 3.0. TRN/DEV/TST: training/development/evaluation set.

Tokenization, lemmatization, part-of-speech and named entity annotations are generated by the Stanford CoreNLP tool (Manning et al., 2014). Most frequent word senses are removed and restored during pre- and post-processing. The same graph recategorization is performed to assign specific subgraphs to a single node as in Cai and Lam (2020). Wikification is done using the DBpedia Spotlight (Daiber et al., 2013) during post-processing.

A.2 Hyper-Parameter Configuration

The hyper-parameters used in our models are described in Table 3.

Embeddings	
lemma	300
POS tag	32
NER tag	16
concept	300
char	32
Char-level CNN	
#filters	256
ngram filter size	[3]
output size	128
Text Encoder	
#transformer layers	4
Graph Encoder	
#transformer layers	2
Transformer Layer	
#heads	8
hidden size	512
feed-forward hidden size	1024
Graph Transformer	
feed-forward hidden size	1024
Biaffine	
hidden size	100

Table 3: Hyper-parameters settings.

[6]AMR 2.0: https://catalog.ldc.upenn.edu/LDC2017T10

[7]AMR 3.0: https://catalog.ldc.upenn.edu/LDC2020T02