# Leveraging AMR Graph Structure for Better Sequence-to-Sequence AMR Parsing

**Linyu Fan♣, Wu Yiheng♣, Jun Xie♠, Junhui Li♣, Fang Kong♣, Guodong Zhou♣**
♣School of Computer Science and Technology, Soochow University, Suzhou, China
♠Alibaba DAMO Academy
lyfannlp@163.com, wuyiheng@aian.com, lijunhui@suda.edu.cn

**Abstract**

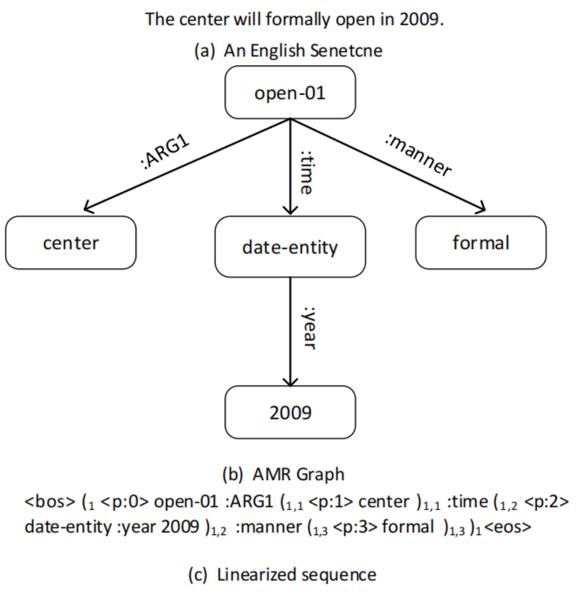
Thanks to the development of pre-trained sequence-to-sequence (seq2seq) models (e.g., BART), recent studies on AMR parsing often regard this task as a seq2seq translation problem by linearizing AMR graphs into AMR token sequences in pre-processing and recovering AMR graphs from sequences in post-processing. Seq2seq AMR parsing is a relatively simple paradigm but it unavoidably loses structural information among AMR tokens. To compensate for the loss of structural information, in this paper we explicitly leverage AMR structure in the decoding phase. Given an AMR graph, we first project the structure in the graph into an AMR token graph, i.e., structure among AMR tokens in the linearized sequence. The structures for an AMR token could be divided into two parts: structure in prediction history and structure in future. Then we propose to model structure in prediction history via a graph attention network (GAT) and learn structure in future via a multi-task scheme, respectively. Experimental results show that our approach significantly outperforms a strong baseline and achieves performance with 85.5 $\pm$0.1 and 84.2 $\pm$0.1 Smatch scores on AMR 2.0 and AMR 3.0, respectively.

**Keywords:** AMR parsing, sequence-to-sequence, Graph Structure

## 1. Introduction

Abstract meaning representation (AMR) is a semantic formalism encoded in the form of a directed acyclic graph, comprising concept nodes and edges that denote the semantic relations between these nodes (Banarescu et al., 2013). Figure 1 presents an English sentence and its corresponding AMR graph. AMR parsing is the task of translating a sentence into an AMR semantic graph automatically. AMR has been applied to a broad range of natural language processing (NLP) tasks such as abstractive summarization (Liu et al., 2015; Hardy and Vlachos, 2018), question answering (Mitra and Baral, 2016), machine translation (Song et al., 2019), and sentiment analysis (Jiang et al., 2022).

With the rapid development of sequence-to-sequence (seq2seq) learning paradigm, recent studies in this literature tend to view AMR parsing as a seq2seq translation problem (Konstas et al., 2017; Xu et al., 2020; Bevilacqua et al., 2021). To apply seq2seq learning to AMR parsing, we need to linearize AMR graphs into AMR token sequences in pre-processing (Figure 1(c)) and then recover AMR graphs from sequences in post-processing. However, treating an AMR graph as a sequence inevitably loses structural information that is encoded in the graph. Taking the AMR graph in Figure 1 as an example, the seq2seq decoder does not explicitly make use of the structural relationship between AMR tokens during the decoding phase, e.g., the parent-child relation between "formal" and "open-01" and the sibling relation between "formal" and "center", which in principle should be beneficial to AMR parsing.


The center will formally open in 2009.

(a) An English Senetcne

<bos> $(_1$ <p:0> open-01 :ARG1 $(_{1,1}$ <p:1> center $)_{1,1}$ :time $(_{1,2}$ <p:2> date-entity :year 2009 $)_{1,2}$ :manner $(_{1,3}$ <p:3> formal $)_{1,3}$ $)_1$ <eos>

(c) Linearized sequence

Figure 1: An example of an AMR graph and its depth-first search (DFS)-based linearization sequence (Bevilacqua et al., 2021). For the presentation convenience, all the parentheses are indexed in (c).

This paper aims to tackle the aforementioned challenge by incorporating AMR structure into the decoding phase of seq2seq AMR parsing. To in-

Corresponding author: Junhui Li.

troduce structure among AMR tokens in a target-side sequence, we initially construct an AMR token graph by transferring the structure from the corresponding AMR graph using heuristic rules. For an AMR token $y_t$, we can divide the structure in the AMR token graph into two parts: structure in prediction history $y_{<t}$ and structure in future $y_{>t}$. Specifically, we properly capture $y_t$'s structural information in $y_{<t}$ with a graph attention network. Meanwhile, we learn $y_t$'s future structure in $y_{>t}$ with a multi-task scheme. Extensive experiments on two English benchmarks show that our approach achieves comparable performance to the state-of-the-art, with an improvement of $1.4\%$ Smatch score on AMR 2.0 and an improvement of $1.0\%$ Smatch score on AMR 3.0 respectively over a strong baseline. Overall, this paper makes the following contributions.

- We use heuristic rules to transfer structure from AMR graph to AMR token graph, i.e., structure among AMR tokens in the linearized sequence. Then we propose to leverage structure in the AMR token graph: modeling structure in prediction history via a GAT and learning structure in future via a multi-task scheme.

- Experimental results on two benchmarks show that our proposed approach significantly improves the performance of AMR parsing, and achieves comparable performance to the state-of-the-art.

## 2. Related Work

AMR parsing, a task that translates sentences into directed and acyclic graphs (Banarescu et al., 2013), has seen diverse modeling approaches for the structure in AMR graphs. Previous research in AMR parsing can be broadly categorized into tree-based approaches (Wang et al., 2015; Groschwitz et al., 2018), graph-based approaches (Flanigan et al., 2014; Werling et al., 2015; Cai and Lam, 2019), transition-based approaches (Zhou et al., 2016; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Astudillo et al., 2020; Zhou et al., 2021a,b), sequence-to-sequence (seq2seq) approaches (Peng et al., 2017; van Noord and Bos, 2017; Konstas et al., 2017; Ge et al., 2019; Xu et al., 2020; Bevilacqua et al., 2021; Xu et al., 2021), and sequence-to-graph (seq2graph) approaches (Lyu and Titov, 2018; Zhang et al., 2019a,b; Cai and Lam, 2020).

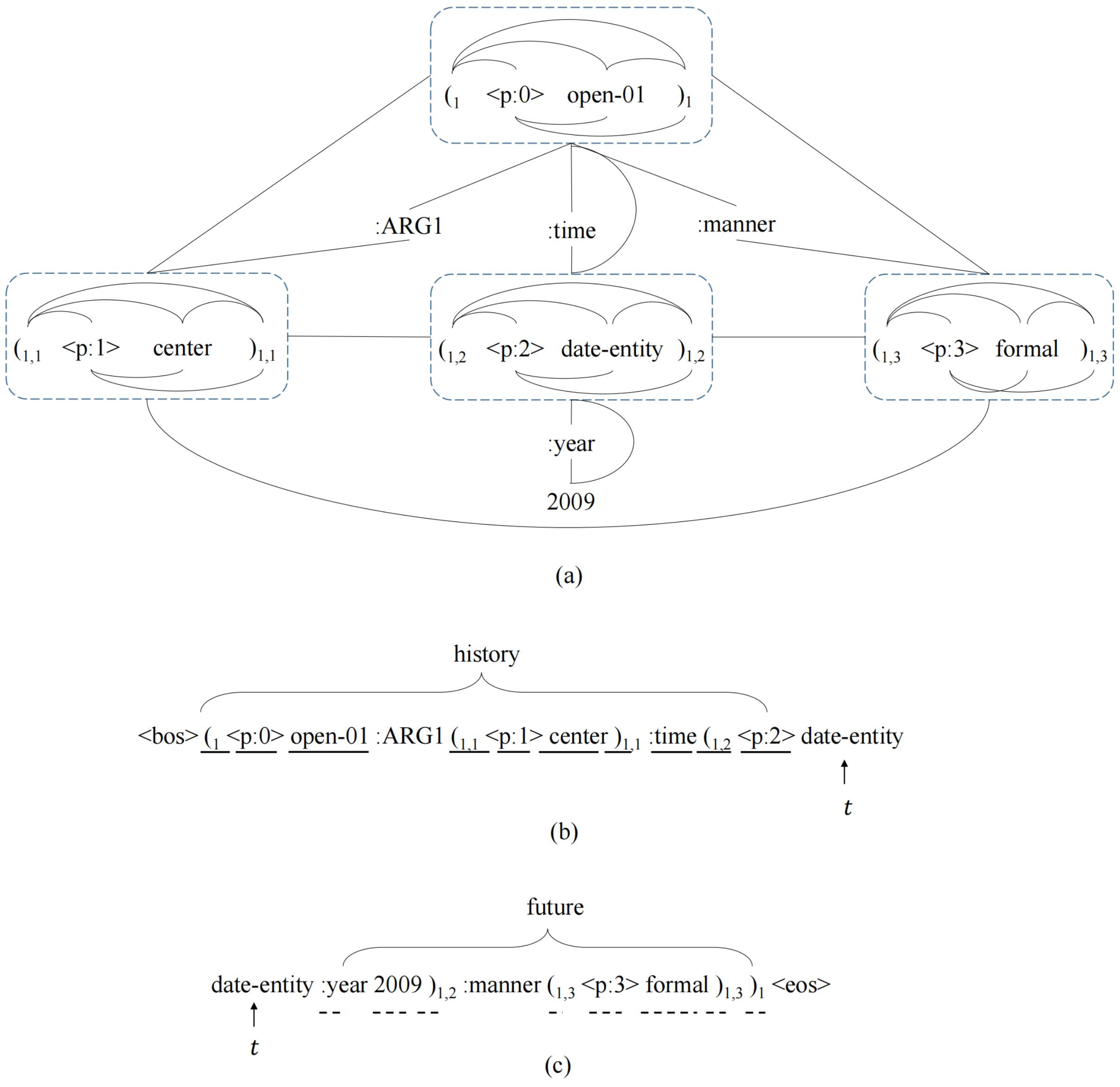
Among the previous studies, tree-based, graph-based, transition-based and seq2graph approaches can naturally take advantage of structural information in the decoding phase. By contrast, linearizing an AMR graph into a sequence unavoidably loses structural information, but seq2seq learning for AMR parsing is prevalent and achieves state-of-the-art performance due to the availability of pre-trained seq2seq models, such as BART (Lewis et al., 2020). Our work also belongs to the paradigm of seq2seq learning, with a target of explicitly incorporating structural information in decoding to compensate for the loss of structural information in AMR linearization. In the direction of seq2seq AMR processing, there are previous works that incorporate syntactic and semantic information in encoding (Ge et al., 2019; Zhu et al., 2019). However, there are limited works that aim to incorporate structural information in decoding. Bai et al. (2022) propose graph self-supervised pre-training to improve the structure awareness, then the pre-trained model is fine-tuned for AMR parsing. The most related work to ours is Yu and Gildea (2022), which incorporates ancestor information in prediction history in the decoding phase. By contrast, we aim to leverage the whole graph structure, including structure in both prediction history and future. Vasylenko et al. (2023) also propose a different approach to incorporate graph information, which shows that partial information of the target sequence can be learned via self-knowledge distillation. Gao et al. (2023) explore the structure loss problem in seq2seq AMR parsing and propose a reverse graph linearization learning framework which uses an extra encoder to encode the right-to-left linearized AMR graph.

## 3. Approach

For easy narration, we refer to items in AMR graph as **AMR nodes** and **semantic edges** while we refer to items in the linearization as **AMR tokens**. Given an AMR graph, we first project its structure in the graph onto an **AMR token graph**, representing the relationships among AMR tokens in the corresponding linearized sequence (Section 3.1). During the decoding phase, where AMR tokens are predicted sequentially in an autoregressive manner, the structure associated with an AMR token $y_t$ at the $t$-th time step can be divided into two parts: the structure in the prediction history $y_{<t}$ and the structure in the future $y_{>t}$. To capture the structure in the history, we introduce a graph attention network (GAT) (Section 3.2), while the structure in the future is learned through a multi-task scheme (Section 3.3).

### 3.1. Building AMR Token Graph from AMR Graph

As shown in Figure 1, an AMR graph is made up of AMR nodes which are connected by semantic edges. In the linearization (Bevilacqua et al., 2021), an AMR node (e.g., *open-01*) is usually linearized into four AMR tokens (e.g., $(_1$, *<p:0>*, *open-01*, and



(1 <p:0> open-01 )1
:ARG1
:time
:manner
(1,1 <p:1> center )1,1
(1,2 <p:2> date-entity )1,2
(1,3 <p:3> formal )1,3
:year
2009
(a)
history
<bos> (1 <p:0> open-01 :ARG1 (1,1 <p:1> center )1,1 :time (1,2 <p:2> date-entity
t
(b)
future
date-entity :year 2009 )1,2 :manner (1,3 <p:3> formal )1,3 )1 <eos>
t
(c)

Figure 2: (a) The AMR token graph projected from the AMR graph shown in Figure 1 (b). (b) underline: Adjacent tokens in prediction history ($\delta = 1$) when the input is *date-entity*. (c) dashed-underline: Adjacent tokens in future ($\eta = 1$) when the input is *date-entity*.

$)_1$) while a semantic edge (e.g., *:ARG1*) maps a single token (e.g., *:ARG1* itself).[1] We denote AMR tokens derived from either an AMR node or a semantic edge as $\mathcal{T}(\cdot)$. The following heuristic rules are employed to construct the AMR token graph from the AMR graph:

- For an AMR node $a$, AMR tokens $\mathcal{T}(a)$ are fully connected, as shown by the dashed boxes in Figure 2.
- If AMR nodes $a_1$ and $a_2$ have a parent-child or sibling relationship, an edge exists for a pair of AMR tokens $(t_i, t_j)$, where $t_i \in \mathcal{T}(a_1)$ and $t_j \in \mathcal{T}(a_2)$.
- Given an AMR token $a$ and a semantic edge $e$ that starts from or ends at $a$, an edge exists for a pair of AMR tokens $(t_i, t_j)$, where $t_i \in \mathcal{T}(a)$ and $t_j \in \mathcal{T}(e)$.

Figure 2 illustrates the **AMR token graph** derived from the AMR graph in Figure 1 (b). In addition to the structures in the AMR graph, edges between AMR tokens of sibling AMR nodes are included. Furthermore, edges in the AMR token graph are undirected. Importantly, when constructing the AMR token graph from the AMR graph, reentrancy nodes do not require special attention.

[1]For AMR nodes representing constants (e.g., *2009*), they map into a single AMR token in linearization.

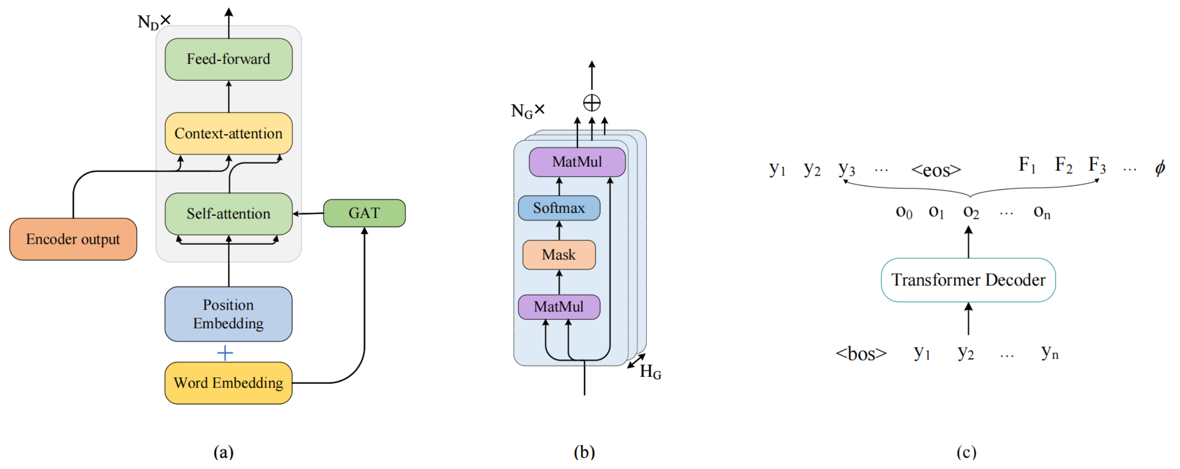


Figure 3: Illustration of our approach. (a) Modeling history structure with a GAT; (b) GAT architecture; (c) Learning future structural relationship via multi-tasking scheme.

## 3.2. Modeling Graph Structure in Prediction History via GAT

As in decoding phase, the AMR tokens are predicted one-by-one in an autoregressive way, the prediction history can be transformed into a subgraph, which preserves structural information between AMR tokens. For example in Figure 2 (b), the sequence itself fails to tell that for the $t$-th time step input *date-entity*, AMR tokens $(_1$, <*p:0*>, and *open-01* are from its parent AMR node while AMR tokens $(_{1,1}$, <*p:1*>, *center*, and $)_{1,1}$ are from its sibling AMR node. Therefore, being aware of such structured information can enhance the representation learning for the input token, and thus can help the model to make better predictions of the next token.

Inspired by Ahmad et al. (2021) which uses graph attention network (GAT) (Veličković et al., 2018) to encode syntax, here we adapt it to properly model structural information between AMR tokens. Figure 3 (a) shows the AMR parsing decoder which is additionally equipped with a GAT in self-attention module. The GAT in Figure 3 (b) consists of $N_G$ identical GAT encoding layers.

**Self-attention with GAT output.** In vanilla decoder-side self-attention, it uses multi-head attention to transform the input $A \in \mathbb{R}^{n\times d}$ (i.e., the output from the previous decoder layer) to $B \in \mathbb{R}^{n\times d}$, where $n$ is the input length and $d$ is the embedding and model size. $B$ is computed as:

$$B = \text{MultiHead}\left(A, A, A, M_S, H_S\right), \quad (1)$$

where $M_S \in \{0, -\infty\}^{n\times n}$ is an upper triangular matrix so the decoder cannot peak to future tokens, $H_S$ is the number of heads in self-attention. Specifically,

$$\text{MultiHead}\left(Q, K, V, M, H\right) = [\text{head}_1, \cdots, \text{head}_H]W^O, \quad (2)$$

$$\text{head}_i = \text{Att}\left(QW_i^Q, KW_i^K, VW_i^V, M\right), \quad (3)$$

$$\text{Att}\left(Q, K, V, M\right) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V, \quad (4)$$

where $W^O \in \mathbb{R}^{d\times d}$, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d_k}$ are parameter matrices, $H$ is the number of heads and $d_k = d/H$.

To enable the self-attention be aware of the structural information for input tokens, we simply add the representation from GAT output to the hidden states of query-key pairs in attention. Given GAT output $\mathcal{G} \in \mathbb{R}^{n\times d}$, we replace Eq. 3 with the following updated head:

$$head_i = \text{Att}\left(QW_i^Q + \mathcal{G}W_G^Q, KW_i^K + \mathcal{G}W_G^K, VW_i^V, M\right), \quad (5)$$

where $W_G^Q, W_G^K \in \mathbb{R}^{d\times d_m}$ are additional parameter matrices. Note that in each decoder layer, we use not all but partial heads to integrate GAT output (See Section 5.1 for discussion).

**Modeling history structure via GAT.** Given a target-side sequence $y = \{y_1, \cdots, y_n\}$ with $n$ AMR tokens, the input sequence is then encoded into an input matrix $\mathcal{G}^{(0)} = [g_1^{(0)}, \cdots, g_n^{(0)}]$, where $g_i^{(0)}$ is word embedding of $y_i$. The $i$-th GAT encoding layer takes $\mathcal{G}^{(i-1)} \in \mathbb{R}^{n\times d}$ as input and computes $\mathcal{G}^{(i)} \in \mathbb{R}^{n\times d}$ as output. GAT adapts the multi-head attention mechanism and models structural information by attending their structurally adjacent tokens in prediction history. The computation of output $\mathcal{G}^{(i)}$ is:

$$\mathcal{G}^{(i)} = \text{MultiHead}\left(\mathcal{G}^{(i-1)}, \mathcal{G}^{(i-1)}, \mathcal{G}^{(i-1)}, M_G, H_G\right), \quad (6)$$

where $H_G$ is the number of heads in GAT attention. The mask matrix $M_G \in \{0, -\infty\}^{n\times n}$ is set as:

$$M_G[i,j] = \begin{cases} 0, & D_{ij} \le \delta \text{ and } i \le j; \\ -\infty, & \text{otherwise.} \end{cases} \quad (7)$$

where $D_{ij}$ is the shortest path distance between token $i$ and $j$ in the AMR token graph. Following (Ahmad et al., 2021), we set $\delta$ to 1 by allowing attention between adjacent tokens only. Meanwhile, we set $M_G[i,j] = -\infty$ when $i > j$ to ensure that position $j$ cannot depend on the positions greater than $j$.

We use the output of the final GAT layer as the output, i.e., $\mathcal{G}^{(N_G)}$. Note that GAT encoder layer only consists of multi-head attention and there is no feed-forward operation and residual connection. Moreover, GAT does not use positional encoding.

## 3.3. Learning Graph Structure in Future via Multi-task Scheme

In decoding, the decoder only predicts an AMR token at each decoding time step, which makes the prediction independent from relevant AMR tokens in future. Motivated by related studies in machine translation that future-aware decoding is helpful (Weng et al., 2017; Lyu et al., 2023), we propose to learn graph structure in future via a multi-task scheme. For target-side AMR token $y_i$, we use $F_i$ to denote the future structural token set having structural relationship with $y_i$, i.e.,

$$F_i = \{y_j \mid j > i \text{ and } D_{ij} \leq \eta\}\,, \tag{8}$$

where $\eta$ is set to 1 by including adjacent tokens only. As shown in Figure 2 (c), for example, the future structural token set for *date-entity* is {*:year, 2009, formal, (, )*}.[2] Therefore, simultaneously learning future structure when predicting an AMR token can enhance the capability of foreseeing the future, and thus can help the model to make better predictions.

Following Weng et al. (2017), we learn to predict the future structural token set $F_t$ under a multi-task scheme when predicting $y_t$ at the $(t-1)$-th time step. As shown in Figure 3 (c), we assume that the output of the last decoder layer at the $(t-1)$-th time step is $o_{t-1}$. Then we predict the next token $y_t$ as:

$$P\left(y_t|y_{<t}, x; \Theta\right) = \text{Softmax}\left(o_{t-1}W^T\right), \tag{9}$$

where $x$ is the source-side input, $\Theta$ is model parameters, $W \in \mathbb{R}^{|V|\times d}$ is target-side word embedding, and $|V|$ is the size of its vocabulary $V$.

Similarly, we predict $y_t$'s future structural token set $F_t$ as:

$$\begin{aligned} P\left(F_t|y_{<t}, x; \Theta\right) &= \text{Softmax}\left(\mathsf{f}\left(o_{t-1}\right)W^T\right), \\ \mathsf{f}\left(o_{t-1}\right) &= o_{t-1}W^S, \end{aligned} \tag{10}$$

where $W^S \in \mathbb{R}^{d\times d}$ is a parameter matrix.

Overall, given the source-side and target-side input pair $(x, y)$, and future structural token sets $\mathcal{F} = (F_1, \cdots, F_n)$, where $F_i = \{f_{i,j}\}|_{j=1}^{|F_i|}$, the model is jointly trained by maximum likelihood estimate, i.e., minimizing the negative log likelihood loss of current token and its future structural token set:[3]

$$L_{joint}\left(y, \mathcal{F}|x; \Theta\right) = L_c\left(y|x;\Theta\right) + \lambda L_f\left(\mathcal{F}|x;\Theta\right), \tag{11}$$

$$L_c\left(y|x;\Theta\right) = -\sum_{i=1}^{n} \log P\left(y_t|y_{<t}, x; \Theta\right), \tag{12}$$

$$\begin{aligned} L_f\left(\mathcal{F}|x;\Theta\right) &= -\sum_{i=1}^{n} \frac{1}{|F_i|} \log P\left(F_i|y_{<t}, x; \Theta\right) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{|F_i|} \frac{1}{|F_i|} \log P\left(f_{i,j}|y_{<t}, x; \Theta\right), \end{aligned} \tag{13}$$

where $\lambda$ is a hyper-parameter used in controlling the weight of predicting future structural token sets.

## 4. Experimentation

### 4.1. Experimental Settings

**Datasets.** We use AMR 2.0 (LDC2017T10) and AMR 3.0 (LDC2020T02) datasets. AMR 2.0 includes 36,521, 1,371 and 1,368 AMRs in the training, development and test datasets, respectively while AMR 3.0 includes 55,635, 1,722, and 1,898 AMRs. Specifically, AMR 2.0 is a subset of AMR 3.0. In pre-processing stage, we use the same depth-first search linearization as Bevilacqua et al. (2021) to linearize AMR graphs while in post-processing step, we also use Penman toolkit (Goodman, 2020) to recover AMR graphs from generated sequences. We follow Bevilacqua et al. (2021) to expand the tokenization vocabulary of the pre-trained model (i.e., BART). Finally, we follow the heuristic rules defined in Section 3.1 to build AMR token graphs.

**Model Settings and Training.** Following Bevilacqua et al. (2021), we use BART-large (Lewis et al., 2020) as the pre-trained model, which has 12 layers and 16 heads for the encoder and the decoder. We randomly initialize all other parameters, including word embeddings of those new added tokens in vocabulary, parameters in the GAT and $W_G^Q$, $W_G^K$ in Eq. 5 and $W^S$ in Eq. 10. When modeling history structure, we set the number GAT encoder layer as 4 and the number of heads in multi-head attention in GAT as 12. When learning future structure, we set $\lambda$ in Eq. 11 to 0.1. We use RAdam optimizer (Liu et al., 2020) to train all models with $\beta_1$ of 0.9 and $\beta_2$ of 0.999. We set the token batch size to 512 and dropout to 0.25, and upgrade our parameters every

[2]We remove repeated tokens in the token set.

[3]In practice, we ignore $F_i$ and do not compute $P\left(F_i|y_{<t}, x; \Theta\right)$ if $F_i$ is an empty set.

| Model | Smatch | Unlab. | NoWSD | Con. | NER | Neg. | Wiki. | Reent. | SRL |
|---|---|---|---|---|---|---|---|---|---|
| Results on AMR 2.0 | | | | | | | | | |
| Bevilacqua et al. (2021) | 83.8 | 86.1 | 84.4 | 90.2 | 90.6 | 74.4 | **84.3** | 70.8 | 79.6 |
| Zhou et al. (2021b) | 84.3 | 87.9 | 84.7 | 90.6 | **92.1** | 72.5 | 80.8 | 74.3 | 83.4 |
| Yu and Gildea (2022) | 84.8 | 88.1 | 85.3 | 90.5 | 91.8 | 74.0 | 84.1 | 75.1 | 83.4 |
| Vasylenko et al. (2023) | **85.7** | **88.6** | **86.2** | **91.0** | 83.9 | **91.1** | 74.2 | **76.8** | 81.8 |
| Baseline (B) | 84.1 $\pm 0.1$ | 86.8 | 84.5 | 90.0 | 91.0 | 74.8 | 81.1 | 72.9 | 83.1 |
| B + History | 85.2 $\pm 0.1$ | 87.7 | 85.4 | **91.1** | 91.9 | 77.5 | 82.2 | 74.3 | 83.7 |
| B + Future | 84.8 $\pm 0.0$ | 87.4 | 85.2 | 91.0 | 91.6 | 75.6 | 82.7 | 74.1 | 83.0 |
| B + History + Future | 85.5 $\pm 0.1$ | 88.2 | 85.6 | 90.7 | 91.5 | 77.2 | 83.0 | 74.8 | **83.8** |
| Results on AMR 3.0 | | | | | | | | | |
| Bevilacqua et al. (2021) | 83.0 | 85.4 | 83.5 | 89.8 | 87.2 | 73.0 | **82.7** | 70.4 | 78.9 |
| Zhou et al. (2021b) | 82.0 | - | - | - | - | - | - | - | - |
| Yu and Gildea (2022) | 83.5 | 86.6 | 84.0 | 89.5 | 88.9 | 72.6 | 81.5 | 74.2 | 82.2 |
| Vasylenko et al. (2023) | **84.5** | **87.5** | **84.9** | **90.5** | 80.7 | **88.5** | 73.1 | 73.7 | 80.7 |
| Baseline (B) | 83.2 $\pm 0.1$ | 86.1 | 83.5 | 89.0 | 87.9 | 71.4 | 79.9 | 73.2 | 82.0 |
| B + History | 83.8 $\pm 0.1$ | 87.2 | 84.4 | 90.0 | 88.9 | 73.1 | 81.5 | 74.8 | **83.2** |
| B + Future | 83.6 $\pm 0.1$ | 86.8 | 83.9 | 89.0 | 88.1 | 70.9 | 80.5 | 74.3 | 82.2 |
| B + History + Future | 84.2 $\pm 0.1$ | 87.3 | 84.3 | 90.3 | **89.0** | 73.3 | 81.2 | **74.9** | 83.0 |

Table 1: Experimental results on AMR 2.0 and 3.0. All above models are fine-tuned on top of BART-large, and do not use extra silver data and graph re-categorization. Our results are average of 3 runs with different random seeds. Scores with **bold**/underline indicate the top/second best performance. To save space, we only show standard deviations of Smatch and omit others.

20 steps. The learning rate and weight decay are 3e-5 and 4e-3, respectively. We train the models on one 3090Ti GPU for 40 epochs and choose the model with the best performance on the development dataset. The training takes about 15 hours on AMR 2.0 and 20 hours on AMR 3.0. In inference, we set the beam size to 5. Meanwhile, it is possible that the predicted history has an invalid structure. When invalid structure occurs, we default view all previous tokens as structured tokens.[4]

**Evaluation.** For evaluation, we report performance in Smatch (Cai and Knight, 2013) and other fine-grained metrics proposed in Damonte et al. (2017). We report results of single models that are tuned on the development set.

## 4.2. Experimental Results

Table 1 shows the performance on AMR 2.0 and AMR 3.0, respectively. We re-implement Bevilacqua et al. (2021) as our baseline, and then leverage AMR structure upon it. We also compare the performance with three recent advanced models (Bevilacqua et al., 2021; Zhou et al., 2021b; Yu and Gildea, 2022; Vasylenko et al., 2023) that are also fine-tuned on BART-large. From the results on AMR 2.0 shown in the upper part of Table 1, we have the following observations.

[4]Thanks to the strong baseline (trained on BART), the final system has very limited invalid structures. For example, only 0.88% sentences (12 out of 1,368) in the DEV set have invalid structure in the generation.

- By simply fine-tuning BART-large on AMR dataset, our strong baseline achieves 0.3 Smatch gain over that of Bevilacqua et al. (2021) (i.e., 84.1 $\pm 0.1$ vs. 83.8).

- Compared to the baseline, modeling history structure improves the performance in Smatch and all other fine-grained metrics. For example, it achieves 1.1 Smatch gain (i.e., 85.2 $\pm 0.1$ vs. 84.1 $\pm 0.1$), indicating that explicitly modeling the structure in prediction history is beneficial to AMR parsing. Moreover, our approach of modeling history structure achieves better performance than that of Yu and Gildea (2022), which models structural relationship of current AMR token to its ancestors.

- Similarly, learning future structure also improves the performance in all metrics, with 0.7 Smatch gain over the baseline (i.e., 84.8 $\pm 0.1$ vs. 84.1$\pm 0.1$). This indicates that it is helpful to simultaneously predict current output token and its future structural tokens.

- Leveraging both history and future structures achieves the best performance. For instance, it results in a 1.4 Smatch gain compared to the baseline (i.e., 85.5$\pm 0.1$ vs. 84.1$\pm 0.1$). This suggests that while historical and future structures exhibit some overlap, such as that the structure relationship between $(y_i, y_j)$ could be captured by both modeling history structure for $y_j$ and learning future structure for $y_i$, these two types of structures are complementary to each other. It is noteworthy that the *Reentrancies* metric benefits significantly from

| #Heads | Smatch | #Heads | Smatch |
|---|---|---|---|
| 0 (Baseline) | 84.5 | 10 | 85.2 |
| 2 | 85.3 | 12 | 84.5 |
| 4 | **85.5** | 14 | 84.4 |
| 6 | 85.3 | 16 | 84.4 |
| 8 | 85.2 | - | - |

Table 2: Performance comparison of B + History model on AMR 2.0 development set when using different number of heads to integrate GAT output.

| Model | Type | Smatch |
|---|---|---|
| Baseline (B) | - | 84.1 $\pm0.1$ |
| B + History | Adjacent ($\delta = 1$) | 85.2 $\pm0.1$ |
| B + History | Random | 84.5 |
| B + History | All | 84.6 |
| B + History | Ancestor | 84.9 |
| B + History | Adjacent + Ancestor | **85.3** |

Table 3: Performance comparison on AMR 2.0 test set when modeling history structure or not.

structural information. Given that reentrancies represent concept nodes with multiple parents in the AMR graph, this is reasonable as modeling/learning parent-child structure can effectively reduce the distance between parent and child nodes (see more details in Section 5.5).

From the results on AMR 3.0 as shown in the lower part of Table 1, we can observe similar performance trends as on AMR 2.0. Overall, our approach (B + History + Future) achieves 1.0 Smatch gain over the baseline (i.e., 84.2 $\pm0.1$ vs. 83.2$\pm0.1$). Compared to related studies, it achieves comparable performance to the state-of-the-art (Vasylenko et al., 2023), which uses structural adapters in Transformer to explicitly incorporate graph information into the learned representations.

## 5. Analysis

Next we use AMR 2.0 as a representative to demonstrate the effect of our approach. Note that we run the analysis experiments with one seed.

### 5.1. Effect of Different Number of Heads to Integrate GAT Output

There are 16 heads in total in the masked self-attention of each decoder layer. We set the number of heads integrating GAT output from 0 to 16 in increments of 2.

As shown in Table 2, optimal performance on the development set is achieved when utilizing 4 heads to integrate GAT output. Using more heads than 4 starts to hurt performance. This is consistent with the findings in Yu and Gildea (2022) where their model achieves the best performance with 4 to 6 heads attending exclusively to ancestors. Therefore, we set the number of heads in each decoder layer integrating GAT output to 4.

### 5.2. Effect of History Structure with GAT

In leveraging structure within prediction history, we follow Ahmad et al. (2021) and set $\delta$ to 1 by allowing attention between adjacent tokens only. To gain a deeper insight into how modeling historical structure contributes to AMR parsing, we carry out the following four contrasting experiments:

- Attending random tokens in prediction history: for token $y_t$, we randomly select a few tokens in prediction history and attend $y_t$ to them. For a fair comparison, the number of randomly selected tokens is the same as that of tokens with $\delta = 1$ in Eq. 7.
- Attending all tokens in prediction history: the attention module attends to all history tokens by setting $\delta$ to the maximum target-side length.
- Attending ancestor tokens in prediction history: inspired by Yu and Gildea (2022), the attention module attends to ancestor tokens in prediction history. For more details of defining ancestor tokens, we refer the reader to Yu and Gildea (2022).
- Attending both adjacent tokens and ancestor tokens in prediction history: as both adjacent tokens and ancestor tokens are important, we test whether attending to both of them will achieve better performance.

As shown in Table 3, compared to baseline, attending to either random tokens or all history tokens in GAT improves parsing Smatch score from 84.1 $\pm0.1$ to 84.5 or 84.6. This suggests that using GAT to implicitly capture the structural relationship between AMR tokens benefits AMR parsing. Moreover, differentiating adjacent tokens from others further improves Smatch score to 85.2 $\pm0.1$, indicating that explicitly modeling history structure via adjacent tokens is effective. It also shows that attending to adjacent tokens slightly outperforms ancestor tokens (i.e., 85.2 $\pm0.1$ VS. 84.9). Finally, attending to both adjacent and ancestor tokens achieves similar performance as attending to adjacent tokens only. We conjecture that it is due to the fact that there exists overlap between ancestor nodes and adjacent tokens. In AMR graph, parent node and its edge are considered as both ancestor nodes and adjacent tokens. Our statistics on the DEV set show that 33% ancestor nodes (including both parent node and its edge) are among adjacent tokens.

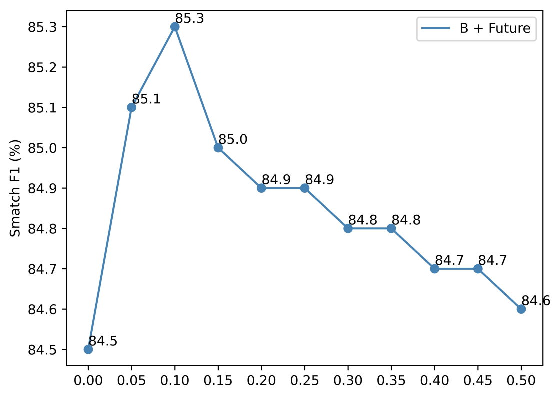


Figure 4: Performance (in Smatch) of B + Future model on AMR 2.0 development set with different values of $\lambda$.

| Model | Type | Smatch |
|---|---|---|
| Baseline (B) | - | 84.1 $\pm 0.1$ |
| B + Future | Adjacent ($\eta = 1$) | **84.8** $\pm 0.0$ |
| B + Future | Random | 84.4 |
| B + Future | All | 84.6 |

Table 4: Performance comparison of B + Future model on AMR 2.0 test set when learning structural or random future AMR tokens.

## 5.3. Effect of Hyper-parameter $\lambda$

In Eq. 11 we use hyper-parameter $\lambda$ to control the contribution of the loss of predicting future structural token sets. We carry out a grid-based hyperparameter sweep for $\lambda$ between 0 and 0.5 with step 0.05 on the development set.

Figure 4 shows the learning curve with different values of $\lambda$. It reaches the best performance 85.3 in Smatch when $\lambda$ is 0.1 while the performance decreases along with the increase of $\lambda$ after 0.1. Therefore, we set $\lambda$ to 0.1.

## 5.4. Effect of Future Structure via Multi-task Scheme

To better understand that learning future structure via multi-tasking scheme helps AMR parsing, similarly we carry out two contrastive experiments:

- Learning random tokens in future: for token $y_t$, we randomly construct a future token set $F_i' \subset \{y_{i+1}, \cdots, y_n\}$. For a fair comparison, the size of $F_i'$ is the same as $F_i$.
- Learning all tokens in future: we set $\eta$ in Eq. 10 to the maximum target-side length such that $F_i' = \{y_{i+1}, \cdots, y_n\}$.

As shown in Table 4, predicting random future tokens has limited effect by slightly improving Smatch score from 84.1 $\pm 0.1$ to 84.4 while predicting all future tokens increases from 84.1 $\pm 0.1$ to 84.6. By contrast, predicting future structural tokens achieves better performance of 84.9 $\pm 0.1$. This reveals that in learning future tokens, it is useful to differentiate structural tokens from the others.

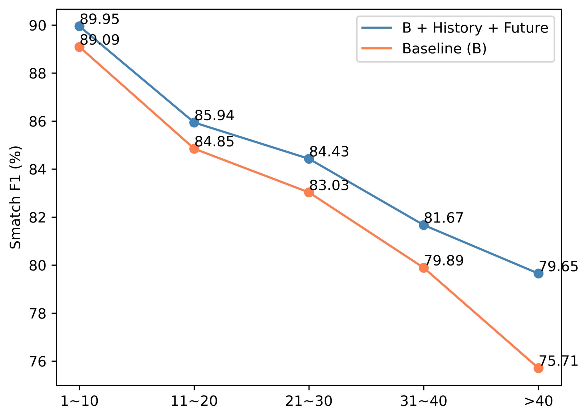


Figure 5: Performance (in Smatch) on AMR 2.0 test set with respect to the number of concept nodes of gold AMR graphs.

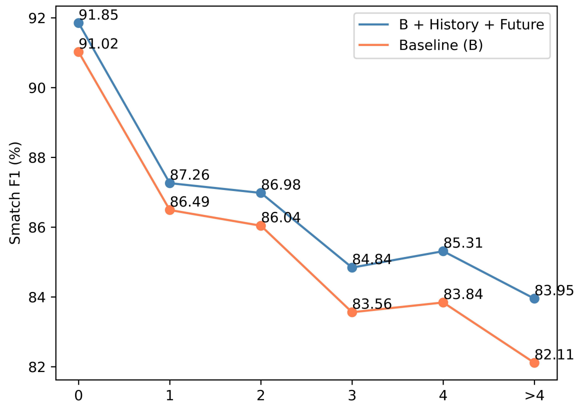


Figure 6: Performance (in Smatch) on AMR 2.0 test set with respect to the reentrancy number of gold AMR graphs.

## 5.5. Effect on AMR Graphs with Different Sizes of Concept Nodes and Reentrancies

Our approach aims to address the loss of structural information incurred during the linearization of an AMR graph into a sequence. Therefore, it is expected that the benefits of our approach are expected to be more obvious for those AMR graphs with complicated structure. While it is challenging to precisely measure the complexity of AMR graphs, here we simply partition AMR graphs to different groups from two aspects, i.e., by their numbers of concept nodes and reentrancies. Then we evaluate the performance of each group independently.

| Type | #His. | #Fut. |
|---|---|---|
| Concept token | 7.46 | 7.53 |
| Sem. label token | 3.24 | 3.84 |
| All | 6.86 | 6.70 |

Table 5: Averaged number of adjacent tokens in prediction history ($\delta = 1$) and future ($\eta = 1$) on AMR 2.0 development dataset.

As shown in Figure 5, leveraging both history and future structures significantly outperforms the baseline across all AMR sizes. Furthermore, the performance gap between our approach and the baseline widens as the number of concept nodes in AMR graphs increases, revealing the importance of incorporating target-side structures, particularly for larger AMR graphs. Finally, it shows that the parsing performance of larger AMR graphs is much lower than that of smaller counterparts.

As shown in Figure 6, we can observe similar performance trends with respect to the reentrancy number. For example, the performance gap between our approach and the baseline becomes largest for AMR graphs with more than 4 reentrancies, where our approach outperforms the baseline by 1.84 Smatch points.

## 5.6. Number of Adjacent Tokens

Taking AMR 2.0 development dataset as a representative, Table 5 shows that, on average, a concept token has 7.46 and 7.53 adjacent tokens in its prediction history and future, respectively, whereas a semantic label token has 3.24 and 3.84 tokens. In our experiment, we set the maximum number of adjacent tokens in both prediction history and future as the maximum target-side length. Therefore, all adjacent tokens are taken into account.

## 6. Conclusion

In addressing the challenge of losing structural information in seq2seq AMR parsing, this paper concentrates on enhancing the parsing process by incorporating structural information during the decoding phase. To achieve this, we delve into two aspects: the historical structure, which we model using a graph network GAT, and the future structure, which we predict through a multi-tasking scheme. Experimental results on benchmarks AMR 2.0 and AMR 3.0 show that our approach achieves significant improvement over a strong baseline. Following the approaches of many related studies, we fine-tune our model based on BART. In future research, we plan to apply this methodology to the latest advanced pre-trained model, AMRBART (Bai et al., 2022).

## Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive feedback. This work was partially supported by the Project 62276178 under the National Natural Science Foundation of China, the Key Project 23KJA520012 under the Natural Science Foundation of Jiangsu Higher Education Institutions.

## Limitations

Modeling graph structure in prediction history requires finding out adjacent tokens in prediction history. Therefore, it increases the decoding time in inference. In training, for each target-side AMR token, it is required to load the positions of its adjacent tokens in prediction history and the adjacent tokens in future. Thus it will increase the memory cost.

## 7. Bibliographical References

Wasi Ahmad, Haoran Li, Kai-Wei Chang, and Yashar Mehdad. 2021. Syntax-augmented multilingual BERT for cross-lingual transfer. In *Proceedings of ACL-IJCNLP*, pages 4538–4554.

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

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of ACL*, pages 6001–6015.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. Amr parsing using stack-lstms. In *Proceedings of EMNLP*, pages 1269–1275.

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.

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*, pages 12564–12573.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. In *Proceedings of EMNLP*, pages 3799–3809.

Deng Cai and Wai Lam. 2020. AMR parsing via graph⇌sequence iterative inference. In *Proceedings of ACL*, pages 1290–1301.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of ACL*, pages 748–752.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of ACL*, pages 2126–2136.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*, pages 536–546.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of ACL*, pages 1426–1436.

Bofei Gao, Liang Chen, Peiyi Wang, Zhifang Sui, and Baobao Chang. 2023. Guiding amr parsing with reverse graph linearization. In *Findings of EMNLP 2023*.

Donglai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for neural AMR parsing. In *Proceedings of IJCAI*, pages 4975–4981.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for AMR graphs. In *Proceedings of ACL: System Demonstrations*, pages 312–319.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of ACL*, pages 1831–1841.

Zhijiang Guo and Wei Lu. 2018. Better transition-based AMR parsing with a refined search space. In *Proceedings of EMNLP*, pages 1712–1722.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using abstract meaning representation. In *Proceedings of EMNLP*, pages 768–773.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of EMNLP*, pages 125–135.

Xiaotong Jiang, Zhongqing Wang, and Guodong Zhou. 2022. Semantic simplification for sentiment classification. In *Proceedings of EMNLP*, pages 11022–11032.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

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 ACL*, pages 146–157.

Young-Suk Lee, Ramón Astudillo, Hoang Thanh Lam, Tahira Naseem, Radu Florian, and Salim Roukos. 2022. Maximum Bayes Smatch ensemble distillation for AMR parsing. In *Proceedings of NAACL*, pages 5379–5392.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*, pages 7871–7880.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of NAACL-HLT*, pages 1077–1086.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *Proceedings of ICLR*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of ACL*, pages 397–407.

Xinglin Lyu, Junhui Li, Min Zhang, Chenchen Ding, Hideki Tanaka, and Masao Utiyama. 2023. Refining history for future-aware neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:500–512.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of AAAI*, page 2779–2785.

Xiaochang Peng, Chuang Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of EACL*, pages 366–375.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*, pages 6000–6010.

Pavlo Vasylenko, Pere Lluís Huguet Cabot, Abelardo Carlos Martínez Lorenzo, and Roberto Navigli. 2023. Incorporating graph information in transformer-based AMR parsing. In *Findings of ACL 2023*, pages 1995–2011.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proceedings of NAACL*, pages 366–375.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xinyu Dai, and Jiajun Chen. 2017. Neural machine translation with word predictions. In *Proceedings of EMNLP*, pages 136–145.

Keenon Werling, Gabor Angeli, and Christoerpher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of ACL*, pages 982–991.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. Improving AMR parsing with sequence-to-sequence pre-training. In *Proceedings of EMNLP*, pages 2501–2511.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2021. Xlpt-amr: Cross-lingual pre-training via multi-task learning for zero-shot amr parsing and text generation. In *Proceedings of ACL-IJCNLP*, pages 896–907.

Chen Yu and Daniel Gildea. 2022. Sequence-to-sequence AMR parsing with ancestor information. In *Proceedings of ACL*, pages 571–577.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In *Proceedings of ACL*, pages 80–94.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. In *Proceedings of EMNLP-IJCNLP*, pages 3786–3798.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021a. AMR parsing with action-pointer transformer. In *Proceedings of NAACL*, pages 5585–5598.

Jiawei Zhou, Tahira Naseem, Ramń, Fernandez Astudillo, Young-Suk Lee, Radu Florian, and Salim Roukos. 2021b. Structure-aware fine-tuning of sequence-to-sequence transformers for transition-based AMR parsing. In *Proceedings of EMNLP*, pages 6279–6290.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of EMNLP*, pages 680–689.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in Transformer for better AMR-to-text generation. In *Proceedings of EMNLP*, pages 5459–5468.