Graph Pre-training for AMR Parsing and Generation

Xuefeng Bai♠♡ , **Yulong Chen**♠♡ , **Yue Zhang**♡◇
♠ Zhejiang University, China
♡ School of Engineering, Westlake University, China
◇ Institute of Advanced Technology, Westlake Institute for Advanced Study, China

Abstract

Abstract meaning representation (AMR) highlights the core semantic information of text in a graph structure. Recently, pre-trained language models (PLMs) have advanced tasks of AMR parsing and AMR-to-text generation, respectively. However, PLMs are typically pre-trained on textual data, thus are sub-optimal for modeling structural knowledge. To this end, we investigate graph self-supervised training to improve the structure awareness of PLMs over AMR graphs. In particular, we introduce two graph auto-encoding strategies for graph-to-graph pre-training and four tasks to integrate text and graph information during pre-training. We further design a unified framework to bridge the gap between pre-training and fine-tuning tasks. Experiments on both AMR parsing and AMR-to-text generation show the superiority of our model. To our knowledge, we are the first to consider pre-training on semantic graphs.

1   Introduction

Abstract meaning representation (AMR; Banarescu et al. (2013)) is a semantic structure formalism. It represents the meaning of a text in a rooted directed graph, where nodes represent basic semantic units such as entities and predicates, and edges represent their semantic relations, respectively. One example is shown in Figure 1(a), with the corresponding sentence in Figure 1(b). Serving as a structural representation, AMR has been shown useful for NLP tasks such as text summarization (Liu et al., 2015; Liao et al., 2018; Chen et al., 2021), machine translation (Song et al., 2019), information extraction (Huang et al., 2016; Zhang and Ji, 2021) and dialogue systems (Bai et al., 2021).

There are two fundamental NLP tasks concerning AMR, namely AMR parsing (Flanigan et al., 2014; Konstas et al., 2017; Lyu and Titov, 2018; Guo and Lu, 2018; Zhang et al., 2019a; Cai and Lam, 2020; Bevilacqua et al., 2021) and AMR-to-text generation (Konstas et al., 2017; Song et al.,
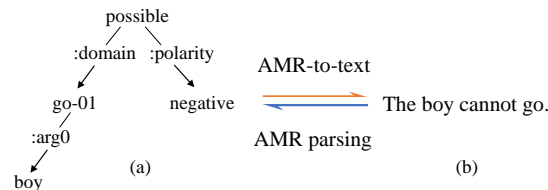

Figure 1: Illustration of AMR tasks: (a) an AMR graph; (b) a corresponding sentence.

2018; Zhu et al., 2019; Zhao et al., 2020; Bai et al., 2020; Ribeiro et al., 2021a). As shown in Figure 1, the former transforms a textual input (*e.g.*, a sentence) into a corresponding AMR structure, and the latter transforms an AMR input into a fluent and grammatical sentence that conveys the same meaning. A common challenge to both tasks is that AMR exists in the form of a graph structure, which is difficult for neural models to learn with limited human-curated data.

Recently, large-scale pre-trained sequence-to-sequence (seq2seq) language models (Lewis et al., 2020; Raffel et al., 2020) have been shown useful for both tasks above. The basic idea is to linearize AMR structures into a sequence form, so that both AMR parsing and AMR-to-text generation can be solved as standard seq2seq tasks, using a pre-trained language model fine-tuned on task-specific data. In this way, semantic knowledge learned in self-supervised text-to-text (`t2t`) pre-training can benefit both text-to-graph (`t2g`) and graph-to-text (`g2t`) transformation.

Intuitively, structural knowledge from AMR can be a useful complement to semantic knowledge from text. A natural question is whether similar self-supervision strategy can be useful for AMR graphs, so that graph-to-graph (`g2g`) denoise auto-encoder training can serve as effective addition to `t2t` pre-training, before a model is fine-tuned on `t2g` and `g2t` tasks. We investigate this problem in this paper. In particular, there are three questions of interest. *First, as mentioned before, is* `g2g`

pre-training complementary to `t2t` *pre-training? Second, what is the most effective way to combine* `t2t` *and* `g2g` *training? Third, is silver data useful for AMR self-supervision training, and what is the most effective way of making use of such data?*

Taking BART (Lewis et al., 2020) as the seq-to-seq model, we introduce two strategies for `g2g` pre-training and propose four tasks to combine `t2t` and `g2g` training. To reduce the gap among different pre-training tasks and between pre-training and fine-tuing, we unify all pre-training tasks and fine-tuning tasks in a general framework. Experimental results on standard benchmarks show that: 1) graph pre-training achieves significant improvements over the state-of-the-art systems; 2) silver data are useful for our pre-training framework; 3) our pre-training framework is a better way than fine-tuning to make use of silver data and; 4) our model is more robust than existing systems in unseen domains. Our final models give the best reported results on both AMR parsing and AMR-to-text generation tasks, with a large margin of improvement over the previous best results. To our knowledge, we are the first to consider graph-to-graph self-supervised training on semantic graphs. We release code at `https://github.com/muyeby/AMRBART`.

2 Related Work

AMR Parsing. Early AMR parsing systems use statistical methods (Flanigan et al., 2014, 2016; Wang et al., 2015a,b). With the advance in deep learning, various neural models are developed for AMR parsing. Those models can be categorized into: 1) neural transition-based parsers (Ballesteros and Al-Onaizan, 2017; Liu et al., 2018; Fernandez Astudillo et al., 2020; Zhou et al., 2021); 2) sequence-to-graph parsers (Zhang et al., 2019a; Lyu et al., 2020; Cai and Lam, 2020) and; 3) sequence-to-sequence parsers (Konstas et al., 2017; Peng et al., 2017, 2018; Zhang et al., 2019b; Xu et al., 2020; Bevilacqua et al., 2021). Recently, pre-training techniques have significantly boosted the performance of AMR parsing. For example, Lyu and Titov (2018), Zhang et al. (2019a,b) and Cai and Lam (2020) use BERT (Devlin et al., 2019) for sentence encoding; Bevilacqua et al. (2021) fine-tune BART for sequence-to-AMR generation. Xu et al. (2020) pre-train a model on relevant seq2seq learning tasks (e.g., machine translation (Bahdanau et al., 2015), syntactic parsing (Zhu et al., 2013)) before fine-tuning on AMR parsing. Similar to

those methods, we consider using pre-trained models to improve the model capacity. However, previous studies focus on fine-tuning language models trained on text data for AMR parsing task, in contract, we focus on integrating structural information into the pre-training. In addition, our method does not require information from auxiliary tasks.

AMR-to-Text Generation. On a coarse-grained level, we categorize existing AMR-to-text generation approaches into two main classes: Graph-to-sequence models that adopt a graph encoder to process an AMR graph and use a sequence decoder for generation (Beck et al., 2018; Damonte and Cohen, 2019; Zhu et al., 2019), and sequence-to-sequence models that linearize an AMR graph into a sequence and solve it as a seq2seq problem using randomly initialized (Konstas et al., 2017) or pre-trained models (Mager et al., 2020; Ribeiro et al., 2021a; Bevilacqua et al., 2021). This work follows a seq2seq manner, but we use an encoder that integrates AMR and text information. The closest to our work, Ribeiro et al. (2021b) integrate AMR structures into pre-trained T5 (Raffel et al., 2020) using adapters (Houlsby et al., 2019) for AMR-to-text generation. However, they do not pre-train on AMR graphs, and their method cannot solve both AMR parsing and AMR-to-text generation tasks as they require the full AMR structure as the input.

Graph Self-supervised Learning. Kipf and Welling (2016) introduce a variational graph auto-encoder to allow self-supervised learning on graph data. Hu et al. (2020a,b) propose local and global learning strategies to pre-train a graph neural network on large-scale protein ego-networks, academic graphs and recommendation data. Lu et al. (2021) enhance the graph learning strategies of Hu et al. (2020b) with dual adaptations. While existing work considers graph neural networks, we pre-train a seq2seq model on AMR graphs. In addition, we jointly pre-train on graphs and text for graph-text correlation modeling. In contrast, existing work pre-trains models on graphs and in isolation with text pre-training. To our knowledge, we are the first to consider AMR as a graph pre-training target.

3 Method

We take BART (Lewis et al., 2020) as the basic seq2seq model (Section 3.1), and introduce graph pre-training strategies (Section 3.2) and an unified pre-training framework (Section 3.3) for both AMR parsing and AMR-to-text generation.
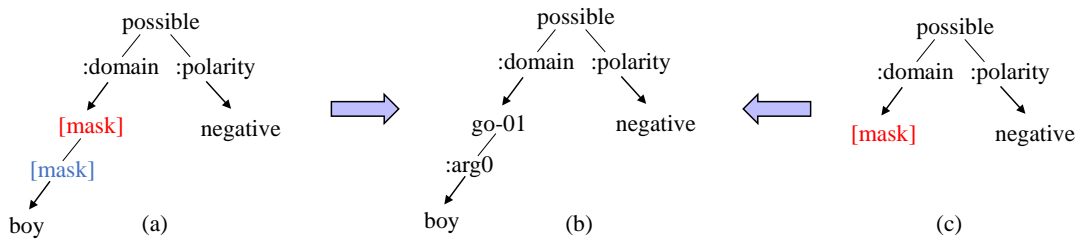
Figure 2: Illustration of two graph pre-training strategies: 1) node/edge level denoising (a→ b); 2) sub-graph level denoising (c→ b). Two transformations can be composed.

3.1 BART

BART (Lewis et al., 2020) is a pre-trained denoising auto-encoder, which is implemented as a seq2seq model based on standard Transformer (Vaswani et al., 2017) architecture. Typically, BART is trained to reconstruct original text based on a corrupted text generated by 5 noising functions: 1) Token Masking. Tokens are randomly replaced by [mask] elements; 2) Token Deletion. Tokens are randomly deleted from the input; 3) Text Infilling. Text spans are randomly replaced by a single [mask] token; 4) Sentence Permutation. Text is divided into segments and then shuffled; 5) Document Rotation. A document is rotated to start with a random token. In the fine-tuning, BART takes a complete text as input and maps it into a task-specific output sequence.

We linearize an AMR graph into a sequence, so that both AMR parsing and AMR-to-text generation can be performed using a seq2seq model. In addition, it allows pre-training on AMR structures using BART. Following Konstas et al. (2017), we adopt the depth-first search (DFS) algorithm, which is closely related to the linearized natural language syntactic trees (Bevilacqua et al., 2021). For instance, the AMR graph in Figure 1 is linearized into: (<Z0> possible :domain (<Z1> go :arg0 (<Z2> boy)) :polarity (<Z3> negative)), where <Z0>, <Z1> and <Z2> are special tokens to handle co-referring nodes. To deal with such AMR symbols, we follow previous work (Bevilacqua et al., 2021) and expand the vocabulary by adding all relations and frames. In addition, to distinguish between texts and AMR graphs, we add two special tokens, <g> and </g>, to mark the beginning and end of AMR graphs, respectively.

3.2 Pre-training on AMR graphs

We introduce two self-supervised training strategies to further pre-train a BART model on AMR

graphs. As shown in Figure 2(a), the node/edge level denoising strategy encourages the model to capture local knowledge about nodes and edges. The graph level denoising strategy (Figure 2(c)) enforces the model to predict a sub-graph, thus facilitating the graph-level learning.

1) Node/edge level denoising. We apply a noise function on AMR nodes/edges to construct a noisy input graph. In particular, the noise function is implemented by masking 15% nodes and 15% edges in each graph. As shown in Figure 2(a), the node [go-01] and edge [:arg0] are replaced with two [mask] tokens.

2) Sub-graph level denoising. This task aims to recover the complete graph when given part of the graph. We randomly remove a sub-graph[1] from the graph and replace it with a [mask] token (*cf.* Figure 2(c)). The masking probability is 0.35.

3.3 Unified Pre-training Framework

The above standard pre-training and fine-tuning strategies are shown in Table 1(a), by using <s> and <g> for differentiating text and graphic information, respectively. However, the model does not fully learn the interaction between textual and AMR information during pre-training. To further address this issue, we consider a unified pre-training framework, which combines text and AMR sequences as input to the denoise auto-encoder. In this way, dynamic masking can be carried out on the text, AMR or both ends, so that the model can learn to make use of one source of information for inferring the other. This can benefit both a parser and a generation model by enforcing the learning of correspondence between text and AMR structures.

In addition, as shown in Table 1, there is a gap between standard pre-training and fine-tuning for AMR from/to text transduction. Specifically, the input and output formats are same in the pre-training (*i.e.*, $\hat{t}2t$ and $\hat{g}2g$) but different in the fine-tuning

---

[1] We define a sub-graph has at least one edge and one node.

Phase		Task	Input	Output
(a)	**Std. P.T.**	$\hat{t}2t$	`<s>` $x_1,..[\text{mask}]..,x_n$ `</s>`	`<s>` $x_1,x_2,...,x_n$ `</s>`
		$\hat{g}2g$	`<g>` $g_1,..[\text{mask}]..,g_m$ `</g>`	`<g>` $g_1,g_2,...,g_m$ `</g>`
	Std. F.T.	$g2t$	`<g>` $g_1,g_2,...,g_m$ `</g>`	`<s>` $x_1,x_2,...,x_n$ `</s>`
		$t2g$	`<s>` $x_1,x_2,...,x_n$ `</s>`	`<g>` $g_1,g_2,...,g_m$ `</g>`
(b)	**Unified P.T.**	$\hat{t}\overline{g}2t$	`<s>` $x_1,..[\text{mask}]..,x_n$ `</s>` `<g>` $[\text{mask}]$ `</g>`	`<s>` $x_1,x_2,...,x_n$ `</s>`
		$\overline{t}\hat{g}2g$	`<s>` $[\text{mask}]$ `</s><g>` $g_1,..[\text{mask}]..,g_m$ `</g>`	`<g>` $g_1,g_2,...,g_m$ `</g>`
		$\hat{t}g2t$	`<s>` $x_1,..[\text{mask}]..,x_n$ `</s>` `<g>` $g_1,g_2,...,g_m$ `</g>`	`<s>` $x_1,x_2,...,x_n$ `</s>`
		$t\hat{g}2g$	`<s>` $x_1,x_2,...,x_n$ `</s>` `<g>` $g_1,..[\text{mask}]..,g_m$ `</g>`	`<g>` $g_1,g_2,...,g_m$ `</g>`
		$\hat{t}\hat{g}2t$	`<s>` $x_1,..[\text{mask}]..,x_n$ `</s>` `<g>` $g_1,..[\text{mask}]..,g_m$ `</g>`	`<s>` $x_1,x_2,...,x_n$ `</s>`
		$\hat{t}\hat{g}2g$	`<s>` $x_1,..[\text{mask}]..,x_n$ `</s>` `<g>` $g_1,..[\text{mask}]..,g_m$ `</g>`	`<g>` $g_1,g_2,...,g_m$ `</g>`
	Unified F.T.	$\overline{t}g2t$	`<s>` $[\text{mask}]$ `</s>` `<g>` $g_1,g_2,...,g_m$ `</g>`	`<s>` $x_1,x_2,...,x_n$ `</s>`
		$t\overline{g}2g$	`<s>` $x_1,x_2,...,x_n$ `</s>` `<g>` $[\text{mask}]$ `</g>`	`<g>` $g_1,g_2,...,g_m$ `</g>`

Table 1: Different pre-training and fine-tuning strategies. P.T. = pre-training, F.T. = fine-tuning. t/g denotes the *original* text/graph. $\hat{t}/\hat{g}$ represents a *noisy* text/graph. $\overline{t}/\overline{g}$ means an *empty* text/graph.

(*i.e.*, t2g and g2t). This gap restrains models to make the best use of *pre-trained knowledge* in the fine-tuning phase. The unified pre-training framework can also benefit task-specific fine-tuning by eliminating the difference of input and output formats between pre-training and fine-tuning.

Formally, denoting the text and linearized graph sequence as t and g, where $t = \{x_1, x_2, ..., x_n\}$ and $g = \{g_1, g_2, ..., g_n\}$. $\hat{t}$ and $\hat{g}$ represent the *noisy* text and graph, respectively, and $\overline{t}$ and $\overline{g}$ refer to the *empty* text and graph, respectively. As shown in Table 1(b), we unify the input format for both pre-training and fine-tuning to tg. For consistency, all input sequences start with a text sequence and end with a graph sequence.

Joint Text and Graph Pre-training. We introduce 4 auxiliary pre-training tasks to encourage information exchanging between graphs and text. As shown in Table 1(b), the auxiliary tasks are:

1) Graph augmented text denoising ($\hat{t}g2t$), where an AMR graph is taken as additional input to help masked text reconstruction;

2) Text augmented graph denoising ($t\hat{g}2g$), where text helps masked graph reconstruction;

3) Noisy graph augmented text denoising ($\hat{t}\hat{g}2t$), where the target text is generated based on a pair of masked text and masked graph;

4) Noisy text augmented graph denoising ($\hat{t}\hat{g}2g$), where a target graph is generated based on a pair of masked text and masked graph.

Dynamic masking rate. Different from standard masking (Devlin et al., 2019) that uses a static masking rate, we adopt a dynamic masking rate p for task $\hat{t}g2t$ and $t\hat{g}2g$. Formally, at step t, we calculate the masking probability p as:

$$p = 0.1 + 0.75 * t/T, \tag{1}$$

where 0.1 is the initial masking rate, T denotes the total training step. p increases as t grows, as t approaches to T, the pre-training tasks $\hat{t}g2t$ and $t\hat{g}2g$ are closer to fine-tuning tasks.

Unified Pre-training and Fine-tuning. In our unified framework, fine-tuning tasks can be viewed as having an *empty* text/graph in the original input, resulting in an input format of $\overline{t}g2t$ for AMR-to-text generation and $t\overline{g}2g$ for AMR parsing. In this way, pre-training and fine-tuning tasks share the same input format, thus facilitating knowledge transfer from pre-training to fine-tuning.

3.4 Training

To pre-train our model, we optimize the total loss ($\mathcal{L}_{total}$) which is calculated as:

$$
\begin{aligned}
\mathcal{L}_{\hat{t}2t} &= -\log P(t|\hat{t}, \overline{g}), \\
\mathcal{L}_{\hat{g}2g} &= -\log P(g|\overline{t}, \hat{g}), \\
\mathcal{L}_{\hat{t}g2t} &= -\log P(t|\hat{t}, g), \\
\mathcal{L}_{t\hat{g}2g} &= -\log P(g|t, \hat{g}), \\
\mathcal{L}_{\hat{t}\hat{g}2t} &= -\log P(t|\hat{t}, \hat{g}), \\
\mathcal{L}_{\hat{t}\hat{g}2g} &= -\log P(g|\hat{t}, \hat{g}), \\
\mathcal{L}_{total} &= \mathcal{L}_{\hat{t}2t} + \mathcal{L}_{\hat{g}2g} + \mathcal{L}_{\hat{t}g2t} \\
&\quad + \mathcal{L}_{t\hat{g}2g} + \mathcal{L}_{\hat{t}\hat{g}2t} + \mathcal{L}_{\hat{t}\hat{g}2g},
\end{aligned}
\tag{2}
$$

where $\mathcal{L}_{\hat{t}2t}$ and $\mathcal{L}_{\hat{g}2g}$ are standard pre-training loss on text (Section 3.1) and graph (Section 3.2), respectively. $\mathcal{L}_{\hat{t}g2t}, \mathcal{L}_{t\hat{g}2g}, \mathcal{L}_{\hat{t}\hat{g}2t}$ and $\mathcal{L}_{\hat{t}\hat{g}2g}$ denote joint pre-training losses (Section 3.3), respectively.

For fine-tuning, the training objectives are:

$$
\begin{aligned}
\mathcal{L}_{\text{amr2text}} &= -\log P(t|\overline{t}, g), \\
\mathcal{L}_{\text{text2amr}} &= -\log P(g|t, \overline{g}),
\end{aligned}
\tag{3}
$$

where $\mathcal{L}_{\text{amr2text}}$ and $\mathcal{L}_{\text{text2amr}}$ are training loss of AMR-to-text generation and AMR parsing, respectively.

Datasets	AMR2.0	AMR3.0	New3	TLP	Bio
Train	36521	55635	-	-	-
Valid	1368	1722	-	-	-
Test	1371	1898	527	1562	500

Table 2: Benchmark AMR datasets.

4 Experiments

We evaluate the effectiveness of our model on five benchmarks and compare the results with state-of-the-art models on AMR parsing and AMR-to-text generation, respectively. In addition to standard supervised training settings, we evaluate the robustness of our model in a zero-shot domain adaptation setting.

4.1 Datasets

Table 2 shows the statistics of datasets. Following Bevilacqua et al. (2021), we use the **AMR2.0** (LDC2017T10) and **AMR3.0** (LDC2020T02). We also evaluate the model performance on **New3**, *The Little Prince* (**TLP**) and *Bio AMR* (**Bio**) corpora. For pre-training, we additionally use 200k silver data parsed by SPRING (Bevilacqua et al., 2021). These data are randomly selected from Gigaword (LDC2011T07) corpus, which shares the same textual source with AMR data.[2]

4.2 Settings

We follow Bevilacqua et al. (2021) in pre-processing and post-processing AMR graphs, except for omitting the recategorization step which does not consistently improve model performance in our preliminary experiments. Our model is built based on a vanilla BART[3]. The best model and hyper-parameters are selected by performance on the validation set. The detailed hyper-parameters are given in Appendix A.

Metrics. Following Bevilacqua et al. (2021), we evaluate on the AMR parsing benchmarks by using Smatch (Cai and Knight, 2013) and other fine-grained metrics.[4] Regarding AMR-to-text, we use three common Natural Language Generation measures, including BLEU (Papineni et al., 2002), CHRF++ (Popović, 2017) and METEOR (Banerjee and Lavie, 2005), tokenizing with the script provided with JAMR (Flanigan et al., 2014).

Setting	Smatch	BLEU	Avg
BART-base	82.7	42.5	62.6
+ $\hat{t}\bar{g}2t$	82.9	42.9	62.9
+ $\bar{t}\hat{g}2g$	83.1	42.6	62.9
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$	83.1	42.8	63.0
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$, $t\hat{g}2g$	83.4	42.8	63.1
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$, $\hat{t}g2t$	83.1	45.3	63.2
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$, $t\hat{g}2g$, $\hat{t}g2t$	83.3	45.0	63.2
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$, $\hat{t}\hat{g}2g$	83.2	43.0	63.1
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$, $\hat{t}\hat{g}2t$	83.1	44.2	63.7
+ $\hat{t}\bar{g}2t$, $\bar{t}\hat{g}2g$, $\hat{t}\hat{g}2g$, $\hat{t}\hat{g}2t$	83.2	44.0	63.6
+ ALL	83.6	45.6	64.1

Table 3: AMP parsing (Smatch) and AMR-to-text generation (BLEU) performance on valid set of AMR2.0.

4.3 Compared Models

For **AMR parsing**, we consider following systems for comparison: 1) Lyu and Titov (2018; LyuT), a neural parser trained by jointly modeling alignments, concepts and relations; 2) Zhang et al. (2019b; Zhang+), a seq2seq approach that incrementally builds up an AMR via predicting semantic relations; 3) Zhou et al. (2020; Zhou+), an aligner-free parser enhanced by explicit dependency and latent structures; 4) Cai and Lam (2020a; CaiL), a graph-based parser that enhances incremental sequence-to-graph model with a graph-sequence iterative inference mechanism; 5) Bevilacqua et al. (2021; Bevilacqua+), a fine-tuned BART model that predicts a linearized AMR graph.

For **AMR-to-text generation**, the compared models are: 1) Zhu et al. (2019; Zhu+), a Transformer-based model that enhances self-attention with graph relations; 2) Zhang et al. (2020; Zhang+), a graph-to-sequence model which uses a dynamic graph convolutional networks for better graph modeling. 3) Bai et al. (2020; Bai+), a graph encoder (Zhu et al., 2019) with a structural decoder that jointly predicts the target text and the input structure; 4) Mager et al. (2020; Mager+), a fine-tuned GPT that predicts text based on a PENMAN linearized AMR graph; 5) Bevilacqua et al. (2021; Bevilacqua+), a fine-tuned BART that predicts text based on a DFS linearized AMR graph; 6) Ribeiro et al. (2021; Ribeiro+), a fine-tuned BART based on a PENMAN linearized AMR graph. For a fair comparison, we leave out models based on T5 (Ribeiro et al., 2021a,b), which has about two times more parameters than BART.

Setting	AMR parsing	AMR-to-text
Full Model	83.6	45.6
- Node/edge masking	83.4	45.1
- Sub-graph masking	83.1	44.7

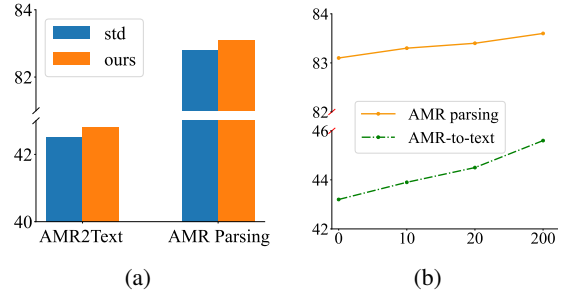Table 4: Impact of two masking strategies.

Figure 3: Development results: (a) comparison of standard pre-training and fine-tuning phase (std) and our unified frameworks; (b) impact of silver data.

4.4 Development Experiments

Table 3 shows results on the validation set of AMR2.0 under different model settings, where we take a fine-tuned BART-based model (Bevilacqua et al., 2021) as our baseline.

We first study the effectiveness of pre-training only on text and graphs. As shown in Table 3, both pre-training on the text ($\bar{t}\bar{g}2t$) and graph ($\bar{t}\hat{g}2g$) leads to better results, and combining them can give better results on both tasks. Also, adding joint pre-training tasks improves the performance. In particular, $t\hat{g}2g$ gives a Smatch improvement of 0.7 for AMR paring, and $\hat{t}g2t$ reaches a BLEU of 45.3 for AMR-to-text generation, which is 2.8 points higher than baseline. Adding $\hat{t}\hat{g}2g$ gives a Smatch of 83.2 for AMR parsing, and $\hat{t}\hat{g}2t$ improves the baseline by 1.7 BLEU points for generation. By combining $t\hat{g}2g$ and $\hat{t}g2t$, the performance increase by 0.6 and 2.5 points on AMR parsing and AMR-to-text generation, respectively. Similar trend can be observed by combining $\hat{t}\hat{g}2g$ and $\hat{t}\hat{g}2t$. Finally, using all 6 pre-training tasks, our model reach a result of 83.6 Smatch and 45.6 BLEU, respectively.

We also study the impact of two graph self-supervised training strategies. In particular, we evaluate the performance after removing the node/edge or the sub-graph masking task independently. As shown in Table 4, the performance decreases on both AMR parsing and AMR-to-text generation tasks without the node/edge level masking strategy. The performance drop is larger when removing the sub-graph masking task, with a margin of 0.5 Smatch and 0.9 BLEU, respectively.

Figure 3(a) compares the performance of standard pre-training ($\hat{t}2t$, $\hat{g}2g$) and fine-tuning ($t2g$, $g2t$) with our unified framework. The unified framework gives better results than standard versions on both tasks. This confirms our assumption that our unified framework is helpful for reducing the gap between pre-training and fine-tuning. Besides, we find that by unifying pre-training and fine-tuning formats, our model converges faster than the baseline during fine-tuning (*cf.* Appendix C.1).

Figure 3(b) shows the model performance regarding different scales of silver data. Even without silver data, the performance of our model is better than the baseline, indicating that graph pre-training is beneficial for downstream tasks when using various auxiliary tasks. When silver data are available, the performance of both AMR parsing and AMR-to-text generation tasks increases as the scale of silver data increases, with a margin of 2 BLEU score. We also *fine-tune* a BART model on silver data under our unified framework (i.e., $\bar{t}g2t$ and $t\bar{g}2g$), and find that our dual graph and text denoising tasks are more useful (*cf.* Appendix C.2 for more analysis and discussion).

4.5 Main Results

AMR parsing. Table 5 lists the result of different models on AMR2.0 and AMR3.0. Among previous works, Bevilacqua+ (2021, large) achieves the best results, consistently outperforming other systems. Compared with the system of Bevilacqua et al. (2021), our model obtains significantly (p<0.01) better Smatch scores in both *base* and *large* settings on both datasets. In particular, our *base* model outperforms the Bevilacqua+ (2021, base) by 0.9 Smatch point on AMR2.0, and our *large* model obtains a Smatch of 85.4 and 84.2 on AMR2.0 and AMR3.0, respectively. To our knowledge, these are the best-reported results, showing the effectiveness of our method.

Besides, Bevilacqua+ (2021, large)[s] uses silver data for fine-tuning, yet does not lead to consistent improvement over Bevilacqua+ (2021, large). In contrast, our large model gives 1.1 and 1.2 higher Smatch than Bevilacqua+ (2021, large)[s] on AMR2.0 and AMR3.0, respectively. This indicates that our pre-training framework is a better way than fine-tuning to make use of silver data. The main

Model	Smatch	Unlab.	NoWSD	Con.	Wiki.	NER	Reent.	Neg.	SRL
AMR2.0									
LyuT (2018)	74.4	77.1	75.5	85.9	75.7	86.0	52.3	58.4	69.8
Zhang+ (2019b)[†]	77.0	80.0	78.0	86.0	86.0	79.0	61.0	77.0	71.0
Zhou+ (2020)[†]	77.5	80.4	78.2	85.9	86.5	78.8	61.1	76.1	71.0
CaiL (2020a)[†]	80.2	82.8	80.0	88.1	86.3	81.1	64.6	78.9	74.2
Xu+ (2020)[†]	80.2	83.7	80.8	87.4	75.1	85.4	66.5	71.5	78.9
Bevilacqua+ (2021, base)[†]	82.7	85.1	83.3	89.7	82.2	90.0	70.8	72.0	79.1
Bevilacqua+ (2021, large)[†]	84.5	86.7	84.9	89.6	**87.3**	83.7	72.3	**79.9**	79.7
Bevilacqua+ (2021, large)[†s]	84.3	86.7	84.8	90.8	83.1	90.5	72.4	73.6	80.5
Ours (base)[†]	83.6	86.7	84.0	90.2	78.6	90.0	71.3	73.7	79.5
Ours (large)[†]	**85.4**	**88.3**	**85.8**	**91.2**	81.4	**91.5**	**73.5**	74.0	**81.5**
AMR3.0									
Bevilacqua+ (2021, large)[†]	83.0	85.4	83.5	89.8	**82.7**	87.2	70.4	**73.0**	78.9
Bevilacqua+ (2021, large)[†s]	83.0	85.4	83.5	89.5	81.2	87.1	71.3	71.7	79.1
Ours (base)[†]	82.5	85.7	82.9	89.4	76.1	86.8	69.9	70.3	78.2
Ours (large)[†]	**84.2**	**87.1**	**84.6**	**90.2**	78.9	**88.5**	**72.4**	72.1	**80.3**

Table 5: AMR parsing results on AMR2.0 and AMR3.0. [*] means the model uses 200k silver data for fine-tuning. [†] means the model is based on pre-trained models. The best result within each row block is shown in bold.

reason is that our models are pre-trained using a denoising auto-encoding manner, which is less sensitive to silver (or noisy) data than fine-tuning. We also find that further fine-tuning our models on silver data (same with pre-training) cannot bring improvement (*cf.* Appendix C.3).

AMR-to-text generation. We report the results of different systems on AMR2.0 and AMR3.0 in Table 6, respectively. With the help of BART, Ribeiro+ (2021) and Bevilacqua+ (2021, large) obtain significantly better results than previous graph-to-sequence and GPT-based models. Compared with Bevilacqua+ (2021), our models (*base* and *large*) give significantly ($p < 0.001$) better results in terms of all evaluation metrics. In particular, our *base* model achieves comparable or better performance than Bevilacqua+ (2021, large). Compared with Bevilacqua+ (2021, large)[s], our large model improves the performance by 3.9 and 2.7 points on AMR2.0 and AMR3.0, respectively. Similar with AMR parsing, we observe that when fine-tuning our models on silver data cannot bring improvement for AMR-to-text generation task (Table 6 and Appendix C.3).

Zero-shot Domain Adaption. We use the model trained on AMR2.0 to get predictions on out-of-domain test sets. Table 7 shows the results on AMR parsing and AMR-to-text generation tasks. Similar to in-domain experiments, our models achieve better results than existing methods. In particular, our *base* model can give comparable performance than Bevilacqua+ (2021, large), and our *large* model obtains the best-reported results. This indicates that

Model	BLEU	CH.	MET.
AMR2.0			
Zhu+ (2019)	31.8	64.1	36.4
Zhang+ (2020)	33.6	63.2	37.5
Bai+ (2020)	34.2	65.7	38.2
Mager+ (2020)[†]	33.0	63.9	37.7
Ribeiro+ (2021)[†‡]	45.9	-	41.2
Bevilacqua+ (2021, base)[†]	42.7	72.2	40.7
Bevilacqua+ (2021, large)[†]	45.3	73.5	41.0
Bevilacqua+ (2021, large)[s†]	45.9	74.2	41.8
Ours (base)[†]	46.6	74.6	41.4
Ours (large)[†]	**49.8**	**76.2**	**42.6**
AMR3.0			
Zhang+ (2020)	34.3	63.7	38.2
Bevilacqua+ (2021, large)[†]	44.9	72.9	40.6
Bevilacqua+ (2021, large)[s†]	46.5	73.9	41.7
Ours (base)[†]	45.9	73.8	40.8
Ours (large)[†]	**49.2**	**76.1**	**42.3**

Table 6: AMR-to-text results on AMR2.0 and AMR3.0. CH.=CHRF++. MET.=METEOR. [*] means the model uses 200k silver data for fine-tuning. Models marked with [†] are based on PLMs. The best result within each row block is shown in bold. [‡]For fair comparison, we report results of tokenized output of Ribeiro+ (2021).

our model is more robust to new domains, thanks to joint graph and text pre-training. Regarding different domains, our method achieves bigger improvements on New3 than the other two domains. This is intuitive, as pre-training strengthens the model representation power on the domain of graph pre-training data, and New3 is closer to it than other two datasets.

In addition, Bevilacqua+ (2021, large)[s] gives lower results than Bevilacqua+ (2021, large) in

Model	New3	TLP	Bio
AMR Parsing			
Bevilacqua+ (2021, large)	73.7	77.3	59.7
Bevilacqua+ (2021, large)[s]	71.8	77.5	59.5
Ours (base)	74.4	77.8	58.8
Ours (large)	**76.9**	**79.8**	**63.2**
AMR-to-Text			
Bevilacqua+ (2021, large)	38.8	25.4	18.7
Bevilacqua+ (2021, large)[s]	38.2	25.1	19.4
Ours (base)	41.0	26.4	16.9
Ours (large)	**44.8**	**29.1**	**20.7**

Table 7: Out of distribution performance on AMR parsing (Smatch) and AMR-to-text (BLEU).

Graph Size	1-10 (522)	11-20 (556)	>20 (293)
AMR parsing	+0.3	+1.0	+0.8
AMR-to-text	+0.9	+3.2	+2.1
Graph Depth	1-3 (422)	4-6 (667)	>6 (282)
AMR parsing	+0.8	+0.9	0.0
AMR-to-text	+1.2	+2.3	+2.8
Reentrancies	0 (622)	1-3 (712)	>4 (37)
AMR parsing	+1.1	+0.6	0.0
AMR-to-text	+2.0	+2.7	+0.4

Table 8: Performance improvements on AMR parsing (Smatch) and AMR-to-text (BLEU).

New3 (both tasks) and TLP (only AMR-to-text generation). In contrast, our model gives consistent improvements on all 3 domains. This can be because fine-tuning leads to catastrophic forgetting of distributional knowledge (Kirkpatrick et al., 2017).

4.6 Impact of Graph

Table 8 shows the effects of the graph size, graph diameter and reentrancies on the performance. We split the test set of AMR2.0 into different groups and report the performance improvement over the baseline model (Bevilacqua et al., 2021). All models are trained on AMR2.0. We first consider graph size, which records the number of nodes in an AMR graph. Our model consistently outperforms the baseline model on both tasks, with the performance gap growing on larger graphs. This indicates that our system is more powerful in dealing with larger graphs. The main reason is that our joint text and graph pre-training mechanism enhances the model with the ability to capture word or span level correlation between text and graph, which is helpful for dealing with long sequence and large graphs.

The graph depth is defined as the longest distance between the AMR node and root node. A graph with deeper depth has more long-range dependencies. For AMR parsing, our model gives a better Smatch than the baseline model on the first two groups of graphs, and a comparable score on graphs with a depth bigger than 6. For AMR-to-text generation, our model consistently improves over the baseline model on all graphs, and the improvements are bigger on deeper graphs. This shows that our model is better for learning more complex graphs. It can be that our graph masking strategies train the model to learn the relationships between a sub-graph and the remaining graph context, making it easier to understand deep graphs.

Reentrancy is the number of nodes that has multiple parents. Reentrancies pose difficulties to both AMR parsing and AMR-to-text tasks (Damonte and Cohen, 2019; Szubert et al., 2020). The more reentrancies, the harder the graph is to be understood. Our method gives significantly ($p<0.01$) better results on both tasks when the input graphs have less than 4 reentrancies. For graphs with more than 4 reentrancies, the proposed model is 0.4 better on AMR-to-text generation task and comparable than the baseline model on AMR parsing task. This means that our system has an overall better ability on learning reentrancies.

4.7 Case study

Table 9 presents two cases of AMR parsing, with the model outputs generated by our model and the baseline model, and the gold output given the same input sentence. As shown in the first case, the baseline model omits the semantic unit "*hard*", thus generates an incomplete AMR graph of a different meaning compared with the input sentence. In contrast, our system preserves the concept "*hard*" and transfers the semantic relations correctly, thanks to the modeling of correspondence between text and graph during pre-training. In the second case, the baseline output includes a **cyclic** sub-graph (*i.e.*, (z1 harm-01 :ARG1 z1)), which is contrary to the grammar that *AMRs should be acyclic*. Our system gives a valid AMR graph which is semantically similar with gold graph.

Table 10 lists two AMR graphs and model outputs of our AMR-to-text model and the baseline model. In the first case, although the baseline generates a fluent sentence, it ignores the concept "*have-purpose-91*", resulting in that the generated sentence is of a different meaning compared with the input graph. In the second AMR graph, "*before*" modifies the phrase "*won many championships*". However, in the baseline output, "*before*" is used to

Text#1: It's getting hard to keep strong and keep carrying on with life.

Gold:
```
(g / get-03
  :ARG1 (a / and
    :op1 (k / keep-02
      :ARG1 (s / strong-02))
    :op2 (k2 / keep-02
      :ARG1 (c / carry-on-02
        :ARG1 (l / live-01))))
  :ARG2 (h / hard-02))
```

Baseline:
```
(z0 / get-03
  :ARG1 (z1 / and
    :op1 (z2 / keep-02
      :ARG1 (z3 / strong-02))
    :op2 (z4 / carry-on-02
      :ARG1 (z5 / life))))
```

Ours:
```
(z0 / get-03
  :ARG1 (z1 / and
    :op1 (z2 / keep-02
      :ARG1 (z3 / strong-02))
    :op2 (z4 / keep-02
      :ARG1 (z5 / carry-on-02
        :ARG1 (z6 / life))))
  :ARG2 (z7 / hard-02
    :ARG1 z1))
```

Text#2: Self harming is addictive, but you can overcome it.

Gold:
```
(c / contrast-01
  :ARG1 (a / addictive-02
    :ARG0 (h / harm-01
      :ARG1 (s / self)))
  :ARG2 (p / possible-01
    :ARG1 (o / overcome-01
      :ARG0 (y / you)
      :ARG1 h)))
```

Baseline:
```
(z0 / addictive-02
  :ARG0 (z1 / harm-01
    :ARG1 z1)
  :concession-of (z2 / possible-01
    :ARG1 (z3 / overcome-01
      :ARG0 (z4 / you)
      :ARG1 z1)))
```

Ours:
```
(z0 / contrast-01
  :ARG1 (z1 / addictive-02
    :ARG0 (z2 / harm-01
      :ARG1 (z3 / self)))
  :ARG2 (z4 / possible-01
    :ARG1 (z5 / overcome-01
      :ARG0 (z6 / you)
      :ARG1 z1)))
```

Table 9: Two AMR parsing cases. Given a text input, we present the gold AMR graph and two model outputs, parsed by the baseline and our model, respectively.

modify the phrase "*participating in international competitions*". Compared with the baseline, our

AMR#1:
```
(h / have-purpose-91
  :ARG1 (t / thing
    :ARG1-of (e / expend-01
      :ARG2 (t2 / transport-01)))
  :ARG2 (a / amr-unknown))
```

Gold: What is the purpose of transportation-related expenditures?
Baseline: What are the transportation expenses?
Ours: What is the purpose of transportation expenses?

AMR#2:
```
(w / win-01
  :ARG0 (p2 / person
    :wiki -
    :name (n / name
      :op1 "Fengzhu"
      :op2 "Xu"))
  :ARG1 (c / championship-02
    :ARG0 p2
    :quant (m / many))
  :time (b / before)
  :part-of (c2 / compete-01
    :mod (i / international)))
```

Gold: Fengzhu Xu has won many championships in international competitions before.

Baseline: Fengzhu Xu won many championships before participating in international competitions.

Ours: Fengzhu Xu has won many championships in international competitions before.

Table 10: Two AMR-to-text generation cases. Given an AMR graph, we present the gold text and two generated outputs, given by baseline and our model, respectively.

system recovers all concepts and maps the modification relationship from the AMR graph to text correctly. This indicates that our model generates more faithful sentences than the baseline.

5 Conclusion

We investigated graph pre-training as a complement to text pre-training for AMR parsing and AMR-to-text generation tasks, using a novel unified framework with dual graph and text denoising. We find that graph pre-training is highly effective for both AMR parsing and AMR -to-text generation, and is a more effective way of making use of silver data compared with fine-tuning. Our methods give the best results on multiple benchmarks for both tasks.

Acknowledgments

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Xuefeng Bai, Yulong Chen, Linfeng Song, and Yue Zhang. 2021. Semantic representation for dialogue modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4430–4445, Online. Association for Computational Linguistics.

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2020. Online back-parsing for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1206–1219, Online. 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, Sofia, Bulgaria. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. 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 3649–3658, Minneapolis, Minnesota. 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.

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.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

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 the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. 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.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020a. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020b. GPT-GNN: generative pre-training of graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1857–1867. ACM.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268, Berlin, Germany. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *CoRR*, abs/1611.07308.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

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.

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

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado. 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.

Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to pre-train graph neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4276–4284. AAAI Press.

Chunchuan Lyu, Shay B. Cohen, and Ivan Titov. 2020. A differentiable relaxation of graph segmentation and alignment for AMR parsing. *CoRR*, abs/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.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

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

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021a. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021b. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Ida Szubert, Marco Damonte, Shay B. Cohen, and Mark Steedman. 2020. The role of reentrancies in Abstract Meaning Representation parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2198–2207, Online. Association for Computational Linguistics.

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

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862, Beijing, China. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. 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.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B. Cohen, Zuozhu Liu, and Lidong Bing. 2020. Lightweight, dynamic graph convolutional networks for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2162–2172, Online. Association for Computational Linguistics.

Zixuan Zhang and Heng Ji. 2021. Abstract Meaning Representation guided graph encoding and decoding for joint information extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–49, Online. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020. Line graph enhanced AMR-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics, pages 732–741, Online. Association for Computational Linguistics.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online. 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.

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 the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria. Association for Computational Linguistics.

Param. Name	Value
Pre-training	
Batch Size	32
Optimizer	AdamW
Learning Rate (lr)	5e-5
Lr Scheduler	inverse_sqrt
Warmup Step	2,500
Total Step	100,000
Extended Vocabulary Size	53,843
Max Sequence Length	512
Mix Precision	fp16 (O1)
Number of Parameters	142M (base), 409M (large)
Training Time	13h (base), 70h (large)
Fine-tuning (Parsing)	
Batch Size	8
Optimizer	AdamW
Learning Rate (lr)	3e-5 (base), 8e-6 (large)
Lr Scheduler	constant
Warmup Step	0
Total Epoch	20
Early Stop	5
Max Sequence Length	512
Beam Size	5
Length Penalty	1.0
Label Smoothing	0
Mix Precision	fp16 (O1)
Training Time	6h (base), 12h (large)
Fine-tuning (Generation)	
Batch Size	8
Optimizer	AdamW
Learning Rate (lr)	1e-5 (base), 2e-6 (large)
Lr scheduler	constant
Warmup Step	0
Total Epoch	20
Early Stop	5
Max Sequence Length	512
Beam Size	5
Length Penalty	1.0
Label Smoothing	0
Mix Precision	fp16 (O1)
Training Time	3h (base), 6h (large)

Table 11: Hyper-parameters of our models on Pre-training and Fine-tuning.

A Model Hyper-Parameters

Table 11 lists all model hyper-parameters used for our experiments. We implement our model based on *Pytorch* and *Huggingface Transformers*. The pre-processed data, source code and pre-trained models are released at `https://github.com/muyeby/AMRBART`.

B Fine-grained Evaluation Metric for AMR Parsing

The Smatch score (Cai and Knight, 2013) measures the degree of overlap between the gold and the prediction AMR graphs. It can be further broken into different sub-metrics, including:

- Unlabeled (Unlab.): Smatch score after re-

Setting	AMR parsing	AMR-to-text
BART	82.7	42.5
+ silver (fine-tuning)	82.6	44.9
+ silver (denoising)	83.6	45.6

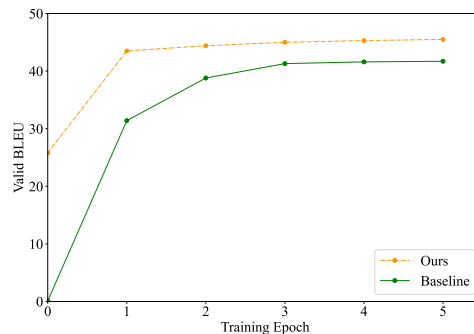Table 12: Ablation study on silver data and denoising tasks.

Figure 4: The learning curve of baseline and our system on AMR-to-text generation task.

moving edge-labels

- NoWSD: Smatch score after ignoring Propbank senses (*e.g.*, go-01 vs go-02)

- Concepts (Con.): F-score on the concept identification task

- Wikification (Wiki.): F-score on the wikification (`:wiki` roles)

- Named Entity Recognition (NER): F-score on the named entities (`:name` roles).

- Reentrancy (Reen.): Smatch score on reentrant edges.

- Negation (Neg.): F-score on the negation detection (`:polarity` roles).

- Semantic Role Labeling (SRL): Smatch score computed on `:ARG-i` roles.

C More Experimental Results

C.1 Effect of Unified Framework

Figure 4 compares the learning curve between our system (fine-tuning from our pre-trained model) and baseline (fine-tuning from vanilla BART, i.e., Bevilacqua+) on AMR2.0 validation set[5]. It can be observed that our system has a initial BLEU score of 26.0, which is significantly ($p < 0.001$) better than the baseline. This confirm that our unified

---

[5]We use the same learning rate and optimizer.

Setting	AMR parsing	AMR-to-text
AMR2.0		
Ours (large)	85.4	49.8
+ silver	85.1	49.6
AMR3.0		
Ours (large)	84.2	49.2
+ silver	83.8	48.9

Table 13: Model performance on AMR2.0 and 3.0 datasets for AMR parsing and AMR-to-text. For AMR parsing, we report Smatch score here, and for AMR-to-text, we report BLEU-4 score here. +silver denotes to that further fine-tuning the model on silver data.

framework can reduce the gap between pre-training and fine-tuning. In addition, the training curve of the proposed model converges faster while the BLEU score is better than the baseline. This indicates that our model has a larger capacity than baseline.

C.2 Impact of denoising Tasks

To distinguish the contribution of de-nosing tasks and silver data, an ablation study is present where we 1) "fine-tune" a vanilla BART on silver data following our unified framework (i.e., $\overline{tg}2t$ and $t\overline{g}2g$); 2) continue pre-train a BART on silver data according to proposed de-nosing tasks (in Table 1). As shown in Table 12, we observe that using sliver data for fine-tuning leads to a 0.1 Smatch decrease in AMR parsing and 2.4 BLEU increase in AMR-to-text. This observation is consistent with previous works (Konstas et al., 2017; Song et al., 2018; Bevilacqua et al., 2021). In addition, using silver data for pre-training gives further improvements on both tasks, with 1.0 Smatch for AMR pasring and 0.7 BLEU for AMR-to-text generation. This indicates that our de-nosing tasks can help model to better understand silver data.

C.3 Are Silver Data Still Helpful for Fine-tuning after Being Used for Pre-training?

As discussed in Section 4.5, we find that graph pre-training is a better way to make use of silver data compared with fine-tuning. We further investigate whether fine-tuning our model on silver data can still bring improvement. As shown in Table 13, our models achieve the best performance on all tasks and datasets, indicating that further fine-tuning our models on silver data decreases the performance. This can be that silver data are already presented in the pre-training phase and thus further fine-tuning can bring no improvement. In addition, fine-tuning can be more sensitive to data quality than pre-training. When training data contain noise (silver data), fine-tuning on such data can in turn damage the model performance.