Structural Adapters in Pretrained Language Models for AMR-to-Text Generation

Leonardo F. R. Ribeiro[†], Yue Zhang[‡] and Iryna Gurevych[†]

[†]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt
[‡]School of Engineering, Westlake University
`ribeiro@aiphes.tu-darmstadt.de`

Abstract

Pretrained language models (PLM) have recently advanced graph-to-text generation, where the input graph is linearized into a sequence and fed into the PLM to obtain its representation. However, efficiently encoding the graph structure in PLMs is challenging because such models were pretrained on natural language, and modeling structured data may lead to catastrophic forgetting of distributional knowledge. In this paper, we propose STRUCTADAPT, an adapter method to encode graph structure into PLMs. Contrary to prior work, STRUCTADAPT effectively models interactions among the nodes based on the graph connectivity, only training graph structure-aware adapter parameters. In this way, we incorporate task-specific knowledge while maintaining the topological structure of the graph. We empirically show the benefits of explicitly encoding graph structure into PLMs using STRUCTADAPT, outperforming the state of the art on two AMR-to-text datasets, training only 5.1% of the PLM parameters.[1]

1 Introduction

Data-to-text tasks aim to generate meaningful and coherent natural language text that faithfully conveys *structured data*. Some examples of structured information include tables (Parikh et al., 2020), Knowledge Graphs (KGs) (Gardent et al., 2017; Vougiouklis et al., 2018) and Abstract Meaning Representation (AMR) (Banarescu et al., 2013). In this work, we focus on AMR-to-text generation where the goal is to generate a fluent and grammatical sentence that is faithful to a given AMR graph (See Figure 1a). AMR is a semantic formalism that has received much research interest (Song et al., 2018; Guo et al., 2019; Ribeiro et al., 2019; Opitz et al., 2020, 2021; Fu et al., 2021) and has been shown to benefit downstream tasks

---

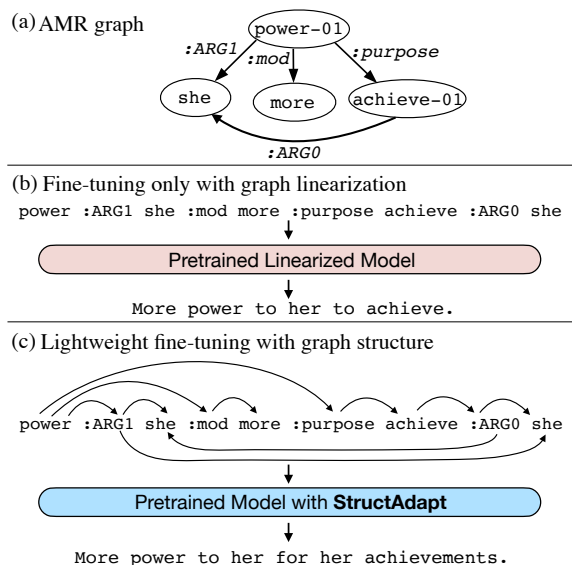[1]Our code and checkpoints are available at https://github.com/UKPLab/StructAdapt.

Figure 1: (a) AMR for the sentence *More power to her for her achievements*. While in (b) the pretrained model gets as input the graph linearization, in (c) it additionally receives the graph connectivity information.

such as text summarization (Liao et al., 2018) and machine translation (Song et al., 2019). Both statistical (Flanigan et al., 2016; Pourdamghani et al., 2016) and neural methods (Bai et al., 2020; Cai and Lam, 2020) have been investigated for AMR-to-text generation, and dominant methods make use of Graph Neural Networks (GNNs) (Kipf and Welling, 2017) or Transformers (Vaswani et al., 2017) for representing the input graph.

Pretrained language models (PLMs) (Devlin et al., 2019; Liu et al., 2020; Radford et al., 2019; Lewis et al., 2020) have been shown useful as a general text representation method, giving much improved results on a wide range of tasks (Wang et al., 2018, 2019). However, they cannot be directly leveraged to benefit AMR-to-text generation, and more generally graph-to-text generation, due to the structural nature of the input. One solution is to transform the structured input into a se-

quence, which can be directly fed into PLMs (See Figure 1b). Recent studies (Mager et al., 2020; Harkous et al., 2020; Ribeiro et al., 2020a, 2021) transform AMRs into sequences by top-down linearization (Konstas et al., 2017). It has been shown that such linearized graph representation can be used to fine-tune a PLM and improve graph-to-text generation performances (Kale, 2020).

The above methods, however, suffer from two salient limitations. First, linearized graph structures are different in nature from natural language. As a result, knowledge from large-scale pretraining intuitively cannot be fully transferred, and fine-tuning a sentence representation using linearized graphs can lead to catastrophic forgetting of such distributional knowledge (Goodfellow et al., 2014; Kirkpatrick et al., 2017). Second, a linearized representation weakens structural information in the original graphs by diluting the explicit connectivity information (i.e., which nodes are connected to each other), and PLMs must infer how edge connections are specified in the sequence. This fact was also observed by Song et al. (2018), Beck et al. (2018) and Ribeiro et al. (2019), who show that GNN encoders outperform sequential encoders for AMR-to-text generation without pretraining.

To mitigate the issues, we aim to explicitly encode the graph data into a PLM without contaminating its original distributional knowledge. To this end, we propose STRUCTADAPT, a novel structure-aware adapter that effectively allows leveraging the input graph structure into PLMs (See Figure 1c). The main idea is to add layer-wise modules, which extract information from the pretrained layers and make use of it in a graph-structure encoding. As shown in Figure 2, STRUCTADAPT employs a *graph convolution* in order to learn representations built upon the graph connectivity over the PLM encoder. Because STRUCTADAPT is added to each encoder layer, deep integration of linguistic knowledge and graph knowledge can be achieved. During fine-tuning, only the adapter parameters are trained, whereas the PLM parameters remain unchanged, in contrast to previous methods based on the graph linearizations that fine-tune all model parameters.

Empirically we show that STRUCTADAPT significantly outperforms linearized fine-tuning baselines and naive sequential adapters (Houlsby et al., 2019). Moreover, STRUCTADAPT is more robust to different graph linearizations, better treats reentrancies (nodes with more than one entering edge) and long-range node dependencies. Our proposed models, based on STRUCTADAPT, surpass the current state of the art on LDC2017T10 and LDC2020T02 datasets by up to 3.1 BLEU points, training only 5.1% of the original PLM parameters.

2 Related Work

Fine-tuning for Graph-to-text Generation. While previous approaches (Song et al., 2018; Ribeiro et al., 2019; Cai and Lam, 2020; Schmitt et al., 2021; Zhang et al., 2020b) have shown that explicitly encoding the graph structure is beneficial, fine-tuning PLMs on linearized structured data has established a new level of performance in data-to-text generation (Nan et al., 2021; Kale, 2020; Ribeiro et al., 2021). Our work can be seen as integrating the advantage of both graph structure encoding and PLMs, using a novel adapter module.

Mager et al. (2020) employ cycle consistency to improve the adequacy of generated texts from AMRs using GPT-2 (Radford et al., 2019), whereas Harkous et al. (2020) train a classifier to rank candidate generations based on the semantic fidelity. Ribeiro et al. (2020a) investigate encoder-decoder PLMs for graph-to-text generation, and show that task-specific pretraining can lead to notable improvements and that PLMs benefit much more from the graph structure of AMRs than of KGs. Hoyle et al. (2021) explore the extent to which PLMs are invariant to graph linearization, finding that models trained on canonical linearizations fail to generalize to meaning-preserving alternatives. Compared to this line of work, which tunes all PLM parameters, our method obtains a further 19x reduction in task-specific parameters, tuning only 5.1% of the parameters while achieving state-of-the-art performance, being more robust to permutations of the graph representation and better encoding larger graphs.

Lightweight Fine-tuning. Recently, different approaches have emerged as an alternative training strategy in order to avoid fine-tuning all parameters of a PLM. Zhang et al. (2019) train a lightweight "side" network that is fused with the pretrained model via summation. Li and Liang (2021) propose to prepend a trainable continuous prefix as an alternative to adapters, maintaining comparable performance in data-to-text tasks using fewer trained parameters. Liu et al. (2021) develop a method to automatically search prompts in the continuous space and evaluate it in few-shot NLU tasks. Ham-

bardzumyan et al. (2021) propose adversarial reprogramming attempts to learn task-specific word embeddings to customize the language model for the downstream task.

Adapter-based approaches (Houlsby et al., 2019; Rebuffi et al., 2017; Lauscher et al., 2020; Pfeiffer et al., 2020a, 2021) introduce a small number of task specific parameters, keeping the underlying pretrained model fixed. Pfeiffer et al. (2020b) propose an adapter method to arbitrary tasks and languages by learning modular language and task representations. The above works are related to STRUCTADAPT as it trains much fewer parameters, but also different because they do not explicitly encode the input structure, whereas STRUCTADAPT directly aims to encode it.

3 Graph-to-Text Model

Let $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0, \mathcal{R}_0)$ denote a rooted and directed AMR graph with a node set $\mathcal{V}_0$ and labeled edges $(u, r, v) \in \mathcal{E}_0$, where $u, v \in \mathcal{V}_0$ and $r \in \mathcal{R}_0$ is a relation type. An example of an AMR graph and its corresponding sentence is shown in Figure 1a.

3.1 Encoder-Decoder Architecture

Consider a conditional generation task where the input is a context x and the output $y = \langle y_1, \ldots, y_{|y|} \rangle$ is a sequence of tokens. In AMR-to-text generation, the context x is the AMR graph and y is the sentence that describes the AMR graph in natural language.

Let $p_\phi(y \mid x)$ denote a PLM parametrized by ϕ, where x is encoded by a bidirectional encoder, and the decoder predicts y autoregressively, conditioned on the encoded x and its left context. We focus on PLMs based on the Transformer encoder-decoder architecture (Vaswani et al., 2017), as they are suitable for conditional text generation. We define $x = \text{LIN}(\mathcal{G}_0)$, where LIN is a function that linearizes $\mathcal{G}_0$ into a sequence of tokens.[2] Following Damonte and Cohen (2019), as shown in Figure 1b, we linearize the AMR into a sequence of nodes and edges using the depth-first traversal of the canonical human-created AMR.[3] In a nutshell, the hidden representation $\mathbf{h}_i^l \in \mathbb{R}^d$, for all $x_i \in x$, is computed by the encoder layer l, where d is the hidden dimension. The decoder hidden representation $\hat{\mathbf{h}}_i^l \in \mathbb{R}^d$ is computed by the layer l of the

---

[2] The variable of a re-entrant node – node with more than one incoming edge – is replaced with its co-referring concept.

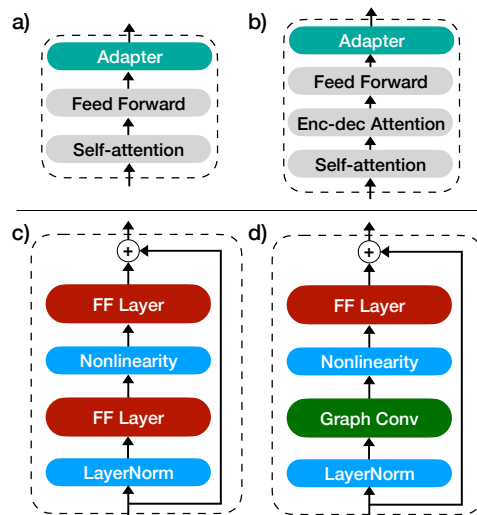[3] Other AMR linearizations are discussed in §6.1.

Figure 2: Integration of the adapter modules with the (a) encoder and (b) decoder layers of the Transformer; layer normalization and residual connections are omitted for clarification. (c) ADAPT with two feed-forwards layers. (d) STRUCTADAPT encodes the graph structure using a *graph convolutional layer*.

autoregressive decoder at time step i.

3.2 Fine-tuning

The model is initialized with pretrained parameters ϕ (e.g. using T5, Raffel et al., 2019) and fine-tuned to optimize the following log-likelihood objective over each gold instance (x, y):

$$\max_\phi \ \log p_\phi(y \mid x) = \sum_{i=1}^{|y|} \log p_\phi(y_i \mid y_{1:i-1}, x).$$

(1)

3.3 Baseline Adapter

We employ an adapter module after the feed-forward sub-layer of each layer on both encoder (Figure 2a) and decoder (Figure 2b) of the PLM. We modify the adapter architecture from Houlsby et al. (2019), computing the adapter representation at each layer l, given the encoder layer representation $\mathbf{h}_i^l$ (or $\hat{\mathbf{h}}_i^l$ in the decoder), as follows:

$$\hat{\mathbf{z}}_i = \mathbf{W}_o^l(\sigma(\mathbf{W}_p^l \, \text{LN}(\mathbf{h}_i^l))) + \mathbf{h}_i^l,$$

(2)

where σ is the activation function and $\text{LN}(\cdot)$ denotes layer normalization. $\mathbf{W}_o^l \in \mathbb{R}^{d \times m}$ and $\mathbf{W}_p^l \in \mathbb{R}^{m \times d}$ are adapter parameters, and m is the hidden dimension of the adapter. Figure 2c illustrates the baseline adapter module, which we call ADAPT.

Training. Let the set of adapters' parameters for the encoder and decoder layers be parametrized by θ. The training objective is the same as Equation (1), but the set of trainable parameters changes: the PLM parameters ϕ are frozen and the adapter parameters θ are the only trainable parameters. In contrast to fine-tuning, adapters substantially reduce the number of trainable parameters that are used to adapt the PLM to the downstream task.

3.4 Limitation

Intuitively, the connection between nodes in the input graph can influence the encoding of x by guiding what to extract from x in order to generate y. Note that in both fine-tuning and ADAPT approaches, the self-attention mechanisms of the encoder layers treat the sequence of nodes and edges x essentially as a fully connected graph, greatly diluting the original graph structure. In this way, the model has to retrieve the original connectivity of the graph from x. For example, the AMR linearization in Figure 1b has two mentions of the node *she*, and the model should capture that both mentions belong to the same node in the original graph.

4 Structural Adapter

We propose STRUCTADAPT, a lightweight alternative to injecting *structural inductive bias*[4] into PLMs.

We first describe the intuition in §4.1 and define our method formally in §4.3.

4.1 Intuition

Injecting *graph structural bias* into graph-to-text models trained from scratch improves the performance compared to linearized approaches (Damonte and Cohen, 2019; Ribeiro et al., 2019). However, it is not straightforward how to effectively model the input graph structure when fine-tuning PLMs, which usually are pretrained using natural language and not structured data.

Our key idea is modeling the graph connectivity in the encoder utilizing an adapter module, using information flows between adjacent nodes in a message-passing update, employing a *graph convolution* (see Figure 2d). In this way, the graph structure substantially impacts the node representations, better encoding the input graph without impacting the knowledge learned during pretraining. This can

---

[4]The model architecture explicitly encodes the graph structure, i.e., which nodes are connected to each other.
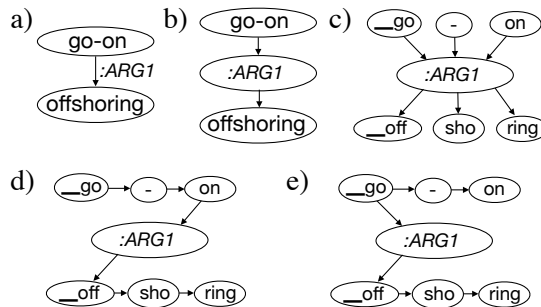

Figure 3: An example of (a) an AMR graph structure, (b) its unlabeled version and three different subword representations: (c) *rep1*, (d) *rep2* and (e) *rep3*.

lead to more efficient and better AMR-to-text generation as we will show in §5 and §6. Moreover, different adapters for distinct graph domains can be used with the same PLM, yielding a high degree of parameter sharing for graph-to-text tasks.

4.2 Graph Representation

We convert each $\mathcal{G}_0$ into a bipartite graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$, replacing each labeled edge $(u, r, v) \in \mathcal{E}_0$ with two unlabeled edges $e_1 = (u, r)$ and $e_2 = (r, v)$. Similar to Beck et al. (2018), this process converts the graph into its unlabeled version. Figure 3 shows an (a) AMR subgraph and (b) its unlabeled representation.

Note that PLMs typically use a vocabulary with subword units (Sennrich et al., 2016). This presents a challenge in how to represent such a graph using subword tokens. Inspired by Ribeiro et al. (2020b), we transform each $\mathcal{G}_1$ into a new token graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each token of a node in $\mathcal{V}_1$ becomes a node $v \in \mathcal{V}$. We convert each edge $(u_1, v_1) \in \mathcal{E}_1$ into a set of edges and connect every token of u_1 to every token of v_1. That is, an edge (u, v) will belong to $\mathcal{E}$ if and only if there exists an edge $(u_1, v_1) \in \mathcal{E}_1$ such that $u \in u_1$ and $v \in v_1$, where u_1 and v_1 are seen as sets of tokens. Figure 3c shows an example of the token graph.

4.3 Method

STRUCTADAPT employs a two-layer architecture in order to re-purpose the PLM for the graph-to-text task using a small number of new parameters. Formally, for each node $v \in \mathcal{V}$, given the hidden representation $\mathbf{h}_v^l$ from the encoder layer l, STRUCTADAPT computes:

$$\mathbf{g}_v^l = \mathsf{GraphConv}_l(\mathsf{LN}(\mathbf{h}_v^l), \{\mathsf{LN}(\mathbf{h}_u^l) : u \in \mathcal{N}(v)\})$$
$$\mathbf{z}_v^l = \mathbf{W}_e^l \sigma(\mathbf{g}_v^l) + \mathbf{h}_v^l , \tag{3}$$

where $\mathcal{N}(v)$ is the immediate neighborhood of v in $\mathcal{G}$. $\mathsf{GraphConv}_l(\cdot)$ is the graph convolution that computes the node representation based on the *local neighborhood* of v, and $\mathbf{W}_e^l \in \mathbb{R}^{d \times m}$ is a parameter. Figure 2d illustrates STRUCTADAPT.[5]

Graph Convolution. The graph convolutional layer allows exploration of distinct strategies for neighborhood aggregation in order to model structural information of the input graph. Different GNN architectures (Velickovic et al., 2018; Xu et al., 2019) can be employed as the graph convolution. Moreover, in this way, we avoid changing the self-attention mechanism of the current pretrained encoder, allowing to also capture *global information* based on the pretrained knowledge.

Our graph convolution is based on the Graph Convolutional Network (GCN) proposed by Kipf and Welling (2017). At each layer l, we compute the representation of a node $v \in \mathcal{V}$ as follows:

$$\mathbf{g}_v^l = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} \mathbf{W}_g^l \mathbf{h}_u^l \; , \qquad (4)$$

where $\mathcal{N}(v)$ is a set of nodes with incoming edges to v and v itself, d_v is the degree of v, and $\mathbf{W}_g^l \in \mathbb{R}^{m \times d}$ is a parameter.

We also consider the variant relational GCN (RGCN) (Schlichtkrull et al., 2018) as graph convolution. RGCN allows capturing the reverse edge direction so that we can consider the differences in the incoming and outgoing relations, which has shown to be beneficial (Beck et al., 2018). In particular, the node representation is computed as:

$$\mathbf{g}_v^l = \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} \mathbf{W}_r^l \mathbf{h}_u^l \; , \qquad (5)$$

where $\mathcal{R}$ denotes the set of relations, i.e., the edge types *default* and *reverse*, $\mathcal{N}_r(v)$ denotes the set of neighbors under relation $r \in \mathcal{R}$, and $\mathbf{W}_r^l \in \mathbb{R}^{m \times d}$ encodes the edge type between the nodes u and v.

Note that STRUCTADAPT computes the refined structural node representation $\mathbf{z}_v^l$ based on the local node context, using as input the global representation $\mathbf{h}_v^l$ generated by the current PLM encoder layer. In this way, the model is able to capture both the global context based on the PLM linguistic knowledge and the local context based on the graph knowledge. Finally, we employ ADAPT into the decoder in order to adapt the language model to the graph-to-text task.

---

[5]Preliminary experiments with other architecture configurations led to worse or similar performance.

	BLEU	chrF++	$\mathcal{M}$	BERT
Mager et al. (2020)	33.0	63.9	-	-
Zhang et al. (2020b)	33.6	63.2	-	-
Harkous et al. (2020)	37.7	-	-	-
Hoyle et al. (2021)	44.9	-	76.54	-
Ribeiro et al. (2020a)	45.8	72.5	-	-
T5base				
FINE-TUNE	38.3±0.3	68.6±0.1	77.8±0.3	95.5±0.1
FT-TOP2(14.8%)	29.9±0.1	63.0±0.1	74.1±0.2	94.4±0.2
FT-BOTTOM2(14.8%)	35.9±0.3	67.0±0.2	76.9±0.1	95.3±0.1
ADAPT(8.5%)	38.7±0.4	69.2±0.2	78.3±0.1	95.6±0.1
STRUCTADAPT-GCN(2.1%)	39.0±0.3	69.1±0.2	78.4±0.2	95.7±0.2
STRUCTADAPT-GCN(8.5%)	41.0±0.5	70.0±0.2	78.4±0.1	95.7±0.1
STRUCTADAPT-RGCN(6.3%)	**44.0**±0.3	**71.2**±0.2	**79.4**±0.1	**95.9**±0.2
T5large				
FINE-TUNE	41.2±0.5	70.2±0.2	78.0±0.1	95.8±0.2
FT-TOP2(7.9%)	28.8±0.4	61.8±0.5	73.9±0.2	94.1±0.2
FT-BOTTOM2(7.9%)	37.6±0.3	68.0±0.2	77.2±0.2	95.5±0.1
ADAPT(6.8%)	42.9±0.3	71.6±0.2	78.9±0.1	96.1±0.1
STRUCTADAPT-GCN(1.7%)	44.1±0.4	71.8±0.3	79.1±0.1	96.1±0.2
STRUCTADAPT-GCN(6.8%)	45.8±0.2	72.5±0.1	79.3±0.1	96.2±0.1
STRUCTADAPT-RGCN(5.1%)	**46.6**±0.3	**72.9**±0.2	**79.6**±0.1	**96.3**±0.1

Table 1: Results on the LDC2017T10 test set. Mean ($\pm$s.d.) over 4 seeds.

5 Experiments

Our models are initialized with pre-trained T5 (Raffel et al., 2019), but our approach can be combined with other PLMs such as BART (Lewis et al., 2020). Our implementation is based on Hugging Face Transformer models (Wolf et al., 2019). We use T5base for all experiments and report results with T5large for the test sets.[6] We use the Adam optimizer (Kingma and Ba, 2015) and employ a linearly decreasing learning rate schedule without warm-up. BLEU is used for the stopping criterion. Following recent work (Mager et al., 2020; Zhang et al., 2020b), we evaluate our proposed models on LDC2017T10 and LDC2020T02 corpora.

Evaluation. We evaluate the results with BLEU (Papineni et al., 2002) and chrF++ (Popović, 2015) metrics. We also report the meaning ($\mathcal{M}$) component of the $\mathcal{MF}$-score (Opitz and Frank, 2021), which measures how well the source AMR graph can be reconstructed from the generated sentence. We use BERTScore (Zhang et al., 2020a) allowing a semantic evaluation that depends less on the surface forms. Finally, we also perform a human evaluation (§5.2).

5.1 Main Results

We compare STRUCTADAPT with four methods: fine-tuning (FINE-TUNE), fine-tuning only the top or bottom 2 layers (FT-TOP2, FT-BOTTOM2) and ADAPT. All

---

[6]Hyperparameter details are in the appendix A.

	BLEU	chrF++	$\mathcal{M}$	BERT
Zhang et al. (2020b)	34.3	63.7	-	-
Bevilacqua et al. (2021)	44.9	72.9	-	-
T5$_{large}$				
FINE-TUNE	41.6±0.6	70.4±0.5	78.5±0.2	96.0±0.1
FT-TOP2(7.9%)	33.4±0.5	63.5±0.3	73.4±0.4	94.3±0.1
FT-BOTTOM2(7.9%)	38.2±0.2	68.3±0.1	78.1±0.2	95.6±0.1
ADAPT(6.8%)	43.0±0.2	71.3±0.2	79.3±0.1	96.2±0.1
STRUCTADAPT-GCN(1.7%)	46.2±0.2	71.8±0.2	79.4±0.3	96.0±0.2
STRUCTADAPT-GCN(6.8%)	47.1±0.4	72.5±0.1	79.7±0.2	96.2±0.1
STRUCTADAPT-RGCN(5.1%)	**48.0**±0.2	**73.2**±0.1	**80.1**±0.3	**96.3**±0.1

Table 2: Results on the LDC2020T02 test set.

Graph Size	ADAPT	STRUCTADAPT-RGCN
All	5.6^A	6.1^B
01-30	6.1^A	6.2^A
31-60	5.4^A	5.4^A
>60	5.2^A	6.2^B

Table 3: Meaning similarity obtained in the human evaluation. The ranking was determined by Mann-Whitney tests with $p<0.05$. Difference between systems which have a letter in common is not statistically significant.

models use the same graph linearization generated by the depth-first traversal. We also report recent state-of-the-art results on both datasets. Tables 1 and 2 show the results.

We find that training only 5.1% task-specific parameters, STRUCTADAPT-RGCN achieves a BLEU score of 46.6 in LDC2017T10, substantially improving over FINE-TUNE and other lightweight baselines (ADAPT, FT-TOP2, FT-BOTTOM2), and outperforming Ribeiro et al. (2020a) and Hoyle et al. (2021) which fine-tune T5 updating significantly more parameters. STRUCTADAPT also achieves state-of-the-art performance on LDC2020T02, considerably improving over Bevilacqua et al. (2021), which implicitly models the graph structure information using linearization techniques.

In general, STRUCTADAPT is better than ADAPT when training the same number of parameters, and slightly better even when training only 1.7% of the parameters for both datasets. This highlights that the gains not only come from using an adapter architecture, but from considering the graph connectivity. STRUCTADAPT-RGCN is more effective than STRUCTADAPT-GCN using fewer parameters, demonstrating that considering reverse relations is advantageous. ADAPT is consistently better than FINE-TUNE, agreeing with our intuition of catastrophic forgetting when fine-tuning. Interestingly, in contrast to popular strategies that focus on upper layers in fine-tuning (Howard and Ruder, 2018; Houlsby et al., 2019; Li and Liang, 2021), FT-BOTTOM2's performance is better than FT-TOP2's, suggesting that lower layers have a significant impact in adapting the PLM to structured data.

Different from our work, both Mager et al. (2020) and Ribeiro et al. (2020a) use the PENMAN notation which makes the input much longer (containing more tokens), and demonstrate that this representation is able to achieve strong results – this is orthogonal to our STRUCTADAPT representation and

can be incorporated in future work.

Overall, the results indicate that explicitly considering the graph structure using an adapter mechanism is effective for AMR-to-text generation, significantly reducing the number of trained parameters while improving generation quality.

5.2 Human Evaluation

To further assess the quality of the generated texts by the adapter-based models in LDC2020T02, we conduct a human evaluation via crowdsourcing using Amazon Mechanical Turk. We follow previous work (Ribeiro et al., 2019; Castro Ferreira et al., 2019) and evaluate the *meaning similarity*, i.e., how close in meaning is the generated text to the reference sentence.[7] We divide the datapoints into 3 different sets by by the graph size, i.e., the number of nodes, after converting edges into nodes (cf. §4.2). This setting allows us to evaluate the performance of the models based on the complexity of the AMR graph.

We randomly select 100 generated texts for each set and each model (total of 600), which annotators then rate on a 1-7 Likert scale. For each text we collect scores from 3 annotators and use MACE (Hovy et al., 2013), a Bayesian model that incorporates the reliability of individual workers, to merge sentence-level labels.[8] Table 3 shows that STRUCTADAPT improves the meaning similarity over ADAPT with statistically significant margins ($p<0.05$). Note that the gains mainly come from datapoints with >60 nodes, indicating that STRUCTADAPT is better when encoding larger graphs.

5.3 Detailed Discussion

Parameter/Performance Trade-off. We investigate how the number of parameters affects the models. A higher hidden dimensionality means more

---

[7] We also assessed the *fluency* of the texts and the differences between the models were not statistically significant.

[8] Refer to Appendix B for a detailed description of the human evaluation.
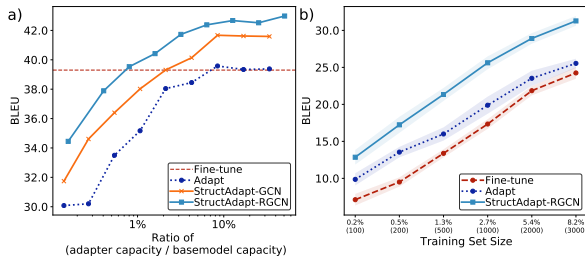
Figure 4: (a) Impact (measure with BLEU) of the number of parameters in the LDC2017T10 dev set. (b) Performance in the LDC2017T10 test set when experimenting with different amounts of training data.

trainable parameters, and smaller adapters introduce fewer parameters at a possible cost to performance. That is, the adapter size controls the parameter efficiency. Figure 4a shows the effect of the number of trained parameters in the performance measured using BLEU. Each point in the ADAPT and STRUCTADAPT curves represents a hidden dimension in the range $[8, 16, \ldots, 2048]$. STRUCTADAPT-GCN is consistently better than ADAPT over all model capacities, even though both approaches train the same number of parameters. STRUCTADAPT-RGCN achieves similar performance than FINE-TUNE when training only 0.8% of the parameters whereas ADAPT achieves similar performance to 8.5%, demonstrating the effectiveness of injecting the graph structure into the PLM.

Low-data Setting. Previous work (Li and Liang, 2021) has shown that lightweight fine-tuning has an advantage in some generation tasks when the training size is smaller. Therefore, we investigate how STRUCTADAPT behaves in a low-data setting. We subsample the LDC2017T10 training set to analyze different smaller training sets. For each size, we sample 5 different datasets and average over 2 training random seeds. Thus, we average over 10 models to get an estimate for each low-data setting.[9] Figure 4b shows the results. First note that both adapter-based approaches improve over FINE-TUNE. When training with only 1000 datapoints, STRUCTADAPT outperforms FINE-TUNE by 8.2 BLEU points. Also note that the gap between ADAPT and FINE-TUNE decreases when the size of the training set increases. In general, STRUCTADAPT outperforms FINE-TUNE and ADAPT in low-resource scenarios by 7.3 and 4.8 BLEU points on average, respectively, whereas requiring much fewer trained parameters

[9]We use the LDC2017T10 dev set to choose hyperparameters and do early stopping.

```
(b / break-up-08
    :ARG1 (i / i)
    :ARG3 (p / person
        :ARG0-of (h / have-rel-role-91
            :ARG1 (p2 / person
                :ARG0-of (h2 / have-rel-role-91
                    :ARG1 i
                    :ARG2 (s3 / son)))
            :ARG2 (f / father)))
    :time (s2 / since
        :op1 (d / date-entity :month 8)))
```

REFERENCE: Me and my son's father have been broken up since August.

FINE-TUNE-2000: I've broken up with my son and father since August.

FINE-TUNE: I've been with my son's father since August.

STRUCTADAPT-2000: Since August 8 I have broken up with my son's father.

STRUCTADAPT: I've been breaking up with my son's father since August.

Table 4: An example of an AMR graph and generated sentences by different models trained on full data and on a low-data setting with 2000 datapoints.

than FINE-TUNE and fewer number of parameters than ADAPT.

Case Study. We perform a case study to provide a better understanding of the STRUCTADAPT's performance. Table 4 shows an AMR graph in PENMAN notation containing reentrancies (marked in bold) and sentences generated by FINE-TUNE and STRUCTADAPT trained on the LDC2017T10 full training set and in a low-data setting where the models are trained with 2000 data points. FINE-TUNE fails in generating a sentence with the correct concept *break-up* whereas STRUCTADAPT correctly generates a sentence that describes the input graph. The incorrect verb tense is due to lack of tense information in AMR. FINE-TUNE-2000 mixes the semantic relation between *I* and *son* (i.e., mistranslation of the edges in the graph) whereas STRUCTADAPT-2000 generates a correct sentence (except by generating the number 8). Overall, STRUCTADAPT produces a more accurate text output than FINE-TUNE by generating correct pronouns and mentions when control verbs and reentrancies are involved, in both full and low-data scenarios.

Model Variations. In Table 5, we report an ablation study on the impact of distinct adapter components, using adapters only in the encoder or decoder. We evaluate different architecture configurations keeping the same number of parameters for a fair comparison. We find that only training adapters in

	BLEU	BERT
FINE-TUNE	38.5	95.6
ADAPT ONLY ENC	38.5	95.7
ADAPT ONLY DEC	11.6	90.3
ADAPT ENC + DEC	38.6	95.6
STRUCTADAPT-GCN ONLY ENC	40.3	95.9
STRUCTADAPT-GCN ENC + DEC	41.7	96.0

Table 5: Impact of the adapter modules in the encoder or decoder in the LDC2017T10 dev set. All adapter-based models have the same number of parameters.

Linearization	Graph Representation	BLEU	BERT
(i) only nodes	rep1	39.1	95.8
	rep2	38.5	95.6
	rep3	38.9	95.7
(ii) nodes and edges	rep1	41.7	96.0
	rep2	40.4	95.8
	rep3	40.8	95.9
	complete graph	39.4	95.8

Table 6: Performance on the LDC2017T10 dev set when using different graph representation strategies.

the decoder is not sufficient for a good performance, even having the same number of parameters. This suggests that adapting the PLM encoder to handle graph structures is key in AMR-to-text tasks. Interestingly, the model that only employs STRUCTADAPT in the encoder (i.e., no ADAPT is used in the decoder) has a better performance (+1.7 BLEU) than using ADAPT in both encoder and decoder, highlighting STRUCTADAPT's strong graph encoding abilities. Finally, the best performance is achieved when we employ STRUCTADAPT in the encoder and ADAPT in the decoder, reaching 41.7 BLEU points.

6 Graph Representation Evaluation

In this section, we explore how different graph properties impact the models' abilities to encode the input graph structure.

6.1 Impact of the Graph Representation

Inspired by Damonte and Cohen (2019), we investigate two different approaches when linearizing the AMR: (i) only nodes have explicit representations, whereas edge relations are represented by the adapter parameters using the RGCN;[10] and (ii) the sequence of nodes and edges using depth-first traversal of the graph.

We also propose and evaluate three different graph structures based on subwords (cf. §4.2): *rep1*: for each edge, we connect every token from the source node to every token of the target node; *rep2*: we connect the last token of the source node to the first token of the target node and connect the tokens of a node sequentially; *rep3*: we connect the first token of the source node to the first token of the target node and connect the token of a node sequentially. Figure 3 shows an example of the three representations for an AMR graph structure.

Additionally, we also investigate a fully connected graph structure (*complete graph*), that is, similarly to the self-attention mechanism in Transformers, all nodes and edges are connected.

As shown in Table 6, explicitly considering nodes and edges in the graph linearization is beneficial. This approach has the advantage of allowing the model to handle new edge relations during inference, as they are not encoded as model parameters. Note that the *complete graph* representation has relatively inferior performance, again demonstrating the advantage of explicitly encoding the input graph connectivity.

Finally, we observe that the best configuration is using nodes and edges with *rep1* (see an example in Figure 3c). We believe that this is because *rep1* allows direct interactions between all source and target tokens, making all token representations of an AMR node directly influenced by the neighbouring tokens.

6.2 Robustness to Graph Linearization

A critical advantage of modeling the graph structure is to be less dependent on linearization strategies because the graph connectivity is invariant to the graph linearization. We thus are interested in measuring the impact of the graph linearization in the models.

Following Hoyle et al. (2021), we investigate three different graph linearizations: (i) CANON: the original order of the canonical human-created linearizations in AMR corpora; (ii) RECONF: the order from the canonical graph linearization is ignored, except for the top node;[11] and (iii) RANDOM: constructs a linearization from a random node in the graph, disregarding all order information from the canonical format, but it remains a valid traversal of the graph. All linearizations are converted to a

---

[10]We use regularization based on the basis decomposition for relation weights (Schlichtkrull et al., 2018) since AMR can contain around 150 different edge types.

[11]RECONF can significantly modify the linearization, including shifting edge labels (e.g., poss to poss-of).

	CANON	RECONF	RANDOM
FINE-TUNE	38.0	35.6	31.3
ADAPT	+0.9	+0.8	+0.9
STRUCTADAPT-RGCN	**+4.1**	**+3.6**	**+5.9**

Table 7: Differences, with respect to FINE-TUNE, in the BLEU score of the LDC2017T10 test set as a function of different graph linearizations.

sequence of node and edge labels using depth-first traversal and used for both training and evaluation. Examples of such graph linearizations are shown in Appendix C.

Table 7 presents the results. Note that while RECONF has a negative impact on all models, STRUCTADAPT has the best performance. ADAPT has similar performance gains over FINE-TUNE in all graph linearizations. Finally, note that for RANDOM, there is a drastic performance drop in FINE-TUNE and the gap between STRUCTADAPT and FINE-TUNE is widest (+5.9 BLEU), demonstrating that explicitly encoding the graph structure is beneficial and that STRUCTADAPT is much less impacted by different graph linearizations.

6.3 Graph Properties

Table 8 shows the effects of the graph size, graph diameter and reentrancies in the performance. First, note that the BLEU scores decrease as the graph size increases since larger graphs often are more complex. The performance gap between STRUCTADAPT and FINE-TUNE becomes larger for relatively larger graphs, showing that STRUCTADAPT is able to better encode complex graphs. As ADAPT is not aware of the graph connectivity, it has much worse scores compared to STRUCTADAPT, especially for larger graphs.

It is expected that the benefit of the STRUCTADAPT will be more evident for AMR graphs containing larger diameter as the encoder is aware of the input graph structure. As seen in Table 8, similarly to the graph size, the scores decrease as the graph diameter increases. STRUCTADAPT achieves a clear improvement when handling graphs with ≥20 diameter, with a improvement of +4.2 BLEU points over FINE-TUNE.

Previous work (Damonte and Cohen, 2019; Szubert et al., 2020) showed that reentrancies (nodes with multiple parents) pose difficulties in encoding AMRs correctly. Because STRUCTADAPT is the only approach to model reentrancies explicitly, we expect it to deal better with these structures. The

graph size	1-30	31-60	>60
# datapoints	548	537	286
FINE-TUNE	40.6	37.3	38.1
ADAPT	+0.5	+1.4	+1.1
STRUCTADAPT-RGCN	**+2.3**	**+4.0**	**+4.6**
graph diameter	1-10	11-20	>20
# datapoints	384	769	218
FINE-TUNE	43.3	37.6	38.5
ADAPT	-0.1	+1.7	+0.3
STRUCTADAPT-RGCN	**+0.5**	**+4.3**	**+4.2**
# reentrancies	0	1-3	4-20
# datapoints	619	664	88
FINE-TUNE	42.9	38.0	31.3
ADAPT	+0.2	+1.7	+0.8
STRUCTADAPT-RGCN	**+3.4**	**+4.4**	**+4.4**

Table 8: Differences, with respect to FINE-TUNE, in the BLEU score of the LDC2017T10 test set as a function of the graph size, graph diameter and number of reentrancies.

gap between STRUCTADAPT and the other models is widest for examples with more reentrancies, confirming our hypothesis. In particular, when graphs contain ≥4 reentrancies, STRUCTADAPT has an improvement of +3.6 BLEU points compared to ADAPT.

7 Conclusion

We presented STRUCTADAPT, a novel adapter architecture to explicitly model graph structures into pretrained language models, providing an extensive evaluation of our approach and showing that it achieves state-of-the-art results on two AMR-to-text benchmarks, training much fewer parameters. We also found that STRUCTADAPT is more effective when encoding complex graphs, when trained on fewer datapoints, and is more robust to different graph linearizations and reentrancies. In future work, we plan to consider other graph-to-text tasks, such as those based on Knowledge Graphs.

Acknowledgments

References

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.

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.

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. Graph transformer for graph-to-sequence learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7464–7471.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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 552–562, Hong Kong, China. 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.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. 2021. End-to-end AMR corefencence resolution. 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 4204–4214, Online. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

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

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. 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 4921–4933, Online. Association for Computational Linguistics.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424, Barcelona, Spain (Online). International Committee on 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*, volume 97 of *Proceedings*

of Machine Learning Research, pages 2790–2799. PMLR.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv e-prints*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

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.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. 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.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, 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.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *CoRR*, abs/2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

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.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Juri Opitz, Angel Daza, and Anette Frank. 2021. Weisfeiler-leman in the bamboo: Novel amr graph metrics and a benchmark for amr graph similarity. *Transactions of the Association for Computational Linguistics*.

Juri Opitz and Anette Frank. 2021. Towards a decomposable metric for explainable evaluation of text generation from AMR. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1504–1518, Online. Association for Computational Linguistics.

Juri Opitz, Letitia Parcalabescu, and Anette Frank. 2020. AMR similarity metrics from principles. *Transactions of the Association for Computational Linguistics*, 8:522–538.

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.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *Technical report, OpenAI*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30, pages 506–516. Curran Associates, Inc.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Jonas Pfeiffer, Yue Zhang, and Iryna Gurevych. 2021. Smelting gold and silver for improved multilingual amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020a. Investigating pretrained language models for graph-to-text generation. *arXiv e-prints*.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020b. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 593–607.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2021. Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 10–21, Mexico City, Mexico. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. 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, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. Journal of Web Semantics, 52-53:1 – 15.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.

Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas J. Guibas, and Jitendra Malik. 2019. Side-tuning: Network adaptation via additive side networks. arXiv e-prints.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with BERT. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B. Cohen, Zuozhu Liu, and Lidong Bing. 2020b. 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.

Appendices

In this supplementary material, we detail experiments' settings and additional information about the human evaluation and graph representations.

A   Details of Models and Hyperparameters

The experiments were executed using the version 3.3.1 of the *transformers* library released by Hugging Face (Wolf et al., 2019). In Table 9, we report the hyperparameters used to train the models presented in this paper. We train until the development set BLEU has not improved for 5 epochs.

	learning rate	batch size	beam search size
FINE-TUNE	3e-05	4	5
FT-TOP2	1e-04	4	5
FT-BOTTOM2	1e-04	4	5
ADAPT	1e-04	4	5
STRUCTADAPT	1e-04	4	5

Table 9: Hyperparameter settings for our methods.

B   Details on the Human Evaluation

The human evaluation was conducted via Amazon Mechanical Turk. We randomly select 100 generated texts for each of the 3 sets and each adapter model (ADAPT, STRUCTADAPT-GCN), with a total of 600 texts to be evaluated. The annotators then rate the meaning similarity on a 1-7 Likert scale. For each text, we collect scores from 3 annotators. We use MACE (Hovy et al., 2013) to further improve upon these raw answers by unsupervised estimation of worker trustworthiness and subsequent recovery of the most likely score. Models are ranked according to the mean of sentence-level scores. We defined a filter for all our evaluations, allowing to participate only workers who have more than 5000 HITs approved and with an acceptance rate of 95% or higher. The task took workers a median time of 1.6 minutes per pair of sentences. We apply a quality control step filtering workers who do not score some faked and known sentences properly or did the experiment in a very short time.

C   Example of Graph Linearizations

In Table 10, we present three different linearizations for the same AMR graph and its corresponding reference sentence. Figure 5 shows the two possible graphs that are represented by the linearizations. In particular, Figure 5a shows a graph that is represented by CANON and RECONF linearizations

and Figure 5b shows a graph that is represented by RANDOM. Note that whereas the linearizations can greatly differ from each other, the graph structure for all linearizations remains very similar.
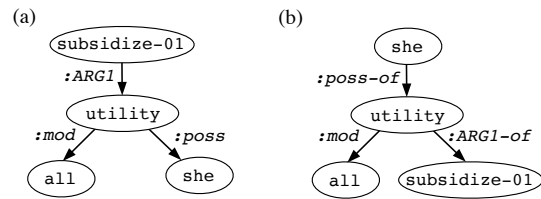

Figure 5: Two AMR graphs with the same meaning.

CANON

(s / subsidize-01
    :ARG1 (u / utility
        :poss (s2 / she)
        :mod (a / all)))

RECONF

(s / subsidize-01
    :ARG1 (u / utility
        :mod (a / all)
        :poss (s2 / she)))

RANDOM

(s2 / she
    :poss-of (u / utility
        :ARG1-of (s / subsidize-01)
        :mod (a / all)))

SENTENCE: Her utilities are all subsidized.

Table 10: Different linearizations for an AMR graph.