Promoting Graph Awareness in Linearized Graph-to-Text Generation

Alexander Hoyle[♣*] **Ana Marasović**[†◇] **Noah A. Smith**[†◇]

[♣]Department of Computer Science, University of Maryland, College Park
[†]Allen Institute for Artificial Intelligence
[◇]Paul G. Allen School of Computer Science and Engineering, University of Washington
hoyle@umd.edu, {anam,noah}@allenai.org

Abstract

Generating text from structured inputs, such as meaning representations or RDF triples, has often involved the use of specialized graph-encoding neural networks. However, recent applications of pretrained transformers to linearizations of graph inputs have yielded state-of-the-art generation results on graph-to-text tasks. Here, we explore the ability of these linearized models to encode local graph structures, in particular their invariance to the graph linearization strategy and their ability to reconstruct corrupted inputs. Our findings motivate solutions to enrich the quality of models' implicit graph encodings via scaffolding. Namely, we use graph-denoising objectives implemented in a multi-task text-to-text framework. We find that these *denoising scaffolds* lead to substantial improvements in downstream generation in low-resource settings.

1 Introduction

Parameter-rich pretrained transformer language models succeed at generating text that is prima facie fluent, but that closer inspection will often reveal to be semantically transgressive (Bisk et al., 2020). Indeed, there is limited practical use for unconditional text generation: we expect language to relate to some identifiable, extrinsic meaning. When a system communicates information to an individual in natural language, it will typically rely on a structured representation of that information. Consequently, generating text that faithfully conveys structured data is an important goal in NLP, where inputs can take the form of tables (ToTTo, Parikh et al., 2020), RDF triples (e.g., WebNLG, Gardent et al., 2017), or Abstract Meaning Representations (AMR, Flanigan et al., 2016). NLP datasets in this domain consists of pairs of structured data (e.g., <henri_matisse, hasOccupation, artist>) and a representation of that data in text ("Matisse is an artist.").

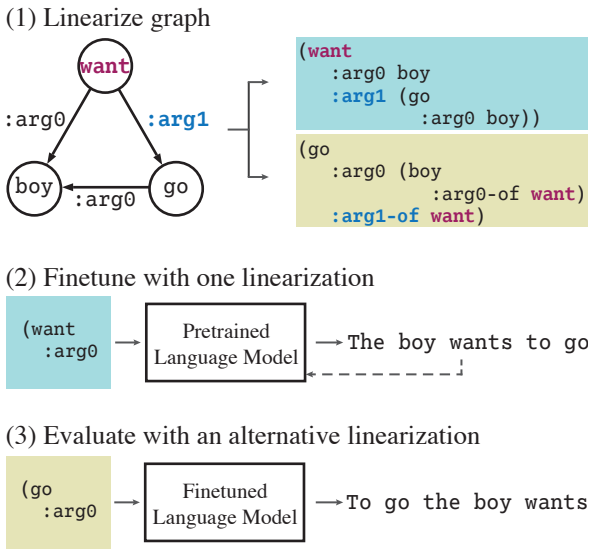

Figure 1: Diagram of our adversarial evaluation procedure for graph-to-text generation using pretrained language models (§3.2). (1) A graph can admit multiple possible linearizations. (2) Following standard practice, we train with a single linearization. (3) At evaluation time, we present the model with a meaning-preserving alternative.

Importantly, these types of inputs can be encoded as *graphs*. Accordingly, advances in neural architectures designed to explicitly encode graphs, such as graph neural networks (GNNs, Kipf and Welling, 2017) and graph transformers, have been used in these graph-to-text settings (Zhu et al., 2019; Zhao et al., 2020; Wang et al., 2020, to name a few). But graphs can also be represented as text (see top portion of Fig. 1). Hence, as an alternative to constraining a model architecture with a graph structure, it is also possible to simply *linearize* a graph into a string and train a sequence-to-sequence model from scratch (Pourdamghani et al., 2016; Konstas et al., 2017; Vinyals et al., 2015). Graph-

[*]Work undertaken during an internship at AI2.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 944–956
August 1–6, 2021.

```
(a / and
  :op1 (d / dream-01
       :ARG1 (f / film
            :ARG0-of (d2 / disturb-01))
       :ARG2-of (r / resemble-01
            :ARG1 a2))
  :op2 (a2 / and
       :op1 (f2 / fascinate-01
            :ARG0 f)
       :op2 d2))
```

(a) Canonical

```
(a / and
  :op1 (d / dream-01
       :ARG2-of (r / resemble-01)
       :ARG1 (f / film
            :ARG0-of (f2 / fascinate-01)
            :ARG0-of d2))
  :op2 (a2 / and
       :op2 (d2 / disturb-01)
       :op1 f2
       :ARG1-of r))
```

(b) Reconfigured

```
(r / resemble-01
  :ARG2 (d / dream-01
       :op1-of (a / and
            :op2 a2)
       :ARG1 (f / film))
  :ARG1 (a2 / and
       :op1 (f2 / fascinate-01
            :ARG0 f)
       :op2 (d2 / disturb-01
            :ARG0 f)))
```

(c) Randomized

Figure 2: Three PENMAN-based linearizations of AMR graphs corresponding to the sentence, "The film is a dream and, like a dream, is both fascinating and disturbing." Note that the **bolded** relation in the graph, `(resemble-01 :ARG1 and)`, is represented differently depending on the linearization.

based encoders were introduced because they outperformed these sequence-to-sequence models. Recently, however, there has been a stark reversal: graph-encoder generation performance has been *far surpassed* by pretrained transformer language models (LMs) finetuned on pairs of linearized graphs and their corresponding surface realizations (Mager et al., 2020; Kale and Rastogi, 2020; Harkous et al., 2020; Ribeiro et al., 2020, henceforth termed *pretrained linearized models*). Moreover, both automated and human assessments indicate that text generated with LMs retains meaning at least as well as graph-encoding baselines (Mager et al., 2020).

This is not the sole product of pretrained models' general language knowledge: Mager et al. (2020), using a GPT-2-based (Radford et al., 2019) model, report that ablating structural graph information (e.g., edges) in the linearized representation notably degrades generation performance, particularly in AMR-to-text tasks. The remarkable performance of pretrained linearized models is intriguing: explicit representation of the input graph by way of the model architecture appears to be well-substituted by simply writing the graph as a linear sequence.

In this work, we further investigate the extent to which pretrained models can leverage linearized graph inputs. Focusing on AMR graphs and sets of RDF triples in English-language datasets, we structure our investigation by first testing whether models' encodings are invariant to the linearization strategy—the way in which a graph is traversed and encoded when producing the linearized representation (see Figure 1). We discover that generation suffers under random permutations of the linearization, and embrace a simple-but-effective training strategy to mitigate this problem: adversarial training (Goodfellow et al., 2015). Motivated by this finding, we encourage more faithful encodings of graph structure via denoising objectives in the more complex AMR setting. This multi-task *scaffolding* (Swayamdipta et al., 2018) reveals that straightforward masking of the graph input is sufficient to improve generation quality in low resource settings.[1]

Moreover, when treating this denoising performance as a proxy for the quality of models' implicit graph encoding, we find that it correlates with the semantic fidelity of the resulting generation better than reasonable alternatives, suggesting possibilities for future evaluation metrics.

We organize our investigation around two research questions:

RQ1 To what extent are pretrained linearized models invariant to graph linearization strategy? (§3)

RQ2 Does encouraging pretrained linearized models' implicit graph representation lead to better generation? (§4)

2 Background: Graph-to-Text Generation

In a graph-to-text setting, we transduce graph inputs g to their corresponding surface realization $y = \langle y_1, \ldots, y_N \rangle$ via a parameterized probabilistic model $p_\theta(\cdot)$. In linearized models specifically, the graph g is first mapped to text by way of a (usually deterministic) linearization function $x = l(g)$, where $p_\theta(\cdot)$ is an off-the-shelf sequence-to-sequence model. This leads to the likelihood objective: $p_\theta(y \mid g) = \prod_{i=1}^{N} p_\theta(y_i \mid x, y_{1:i-1})$. When $p_\theta(\cdot)$ is an autoregressive pretrained transformer, generation quality far exceeds architectures with encoders specifically engineered to encode graphs (Mager et al., 2020; Kale and Rastogi, 2020; Harkous et al., 2020; Ribeiro et al., 2020).

[1]Implementation available at `github.com/ahoho/transformers/tree/graph-promotion`

	N	Dev. ppl.	Avg. edges
LDC2017T10	36k	21.1	11.4
WebNLG	18k	9.2	3.0

Table 1: Dataset statistics. Perplexity estimated on the development set with GPT-2 (Radford et al., 2019) fine-tuned on the training data using default hyperparameters in the `transformers` library (`gpt-2` model, Wolf et al., 2020).

Graph-to-Text Generation Datasets We explore two datasets for generation from a graph structure to English text.

Abstract Meaning Representation (AMR, Banarescu et al., 2013) is a formalism intended to represent the propositional meaning of utterances—"who is doing what to whom"—using graphs that have minimal dependence on the surface form. AMR graphs are directed and acyclic with a single "top" node (Goodman, 2020). They can be represented as either a graph, a tree, or sets of triples (van Noord and Bos, 2017). For our data, we use the AMR 2.0 release (LDC2017T10),[2] both because it spans a varied set of domains and styles, and because of its extensive use in prior work.

A simpler graph-to-text problem involves converting a set of RDF triples to natural text realizations of the information contained in the set, exemplified by the WebNLG dataset (Gardent et al., 2017). WebNLG pulls information from an existing knowledge base (DBPedia, Mendes et al., 2012) for a specific subset of 15 categories (e.g., "astronaut"). To generate the paired sentences, crowdworkers verbalize individual triples. Then, for examples consisting of multiple triples, they merge already-annotated sentences and apply minimal changes (leading to reduced sentence complexity relative to AMR; see perplexity scores in Table 1). There can be multiple surface realizations per input.

Models To study pretrained linearized models' invariance to graph linearization, we use T5 (Raffel et al., 2020), an *encoder-decoder* transformer (Vaswani et al., 2017) that has led to state-of-the-art generation on AMR (specifically, LDC2017T10) and WebNLG (Kale and Rastogi, 2020; Ribeiro et al., 2020).

We modify the T5 implementation from the `transformers` library (Wolf et al., 2020).[3] We use the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 0.0001, selected from the set $\{0.001, 0.0001, 3 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-6}\}$ after tuning on 1000 training examples across five random seeds.[4] We train until development set BLEU has not improved for 10 epochs. See Appendix A.1 for further details.

Evaluation Measures As a primary metric, we evaluate generated text using BLEU (Papineni et al., 2002), calculated with `SacreBLEU` (Post, 2018). Despite its limitations in generation settings, BLEU still generally accords with rankings of models, either by human evaluations or by alternate metrics (Manning et al., 2020). We also evaluate our scaffolding models (§4) using BertScore (Zhang et al., 2020), which measures token similarity with contextual embeddings, permitting a more nuanced measure of semantic similarity. Lastly, we use the $\mathcal{M}$ portion of the $\mathcal{MF}$-score (Opitz and Frank, 2020), which measures how well the source AMR graph can be reconstructed from the generated target sentence using an off-the-shelf parser. Unlike BLEU, which applies corpus-wide, this metric provides a best-guess at sentence-level accuracy.

3 RQ1: Robustness to Permutation of Graph Linearization

In this section, we explore the extent to which pretrained linearized models are invariant to the particular method used to linearize the input graph. Motivated by the strong graph-to-text performance of these models, we ask: do they implicitly develop a robust internal encoding of the input graph? Whereas a GNN-based model has an architecture designed for graph representation (e.g., information flows between adjacent nodes in a message-passing update), a linearized model must infer how connections are specified in a sequence during training.

If linearized models do form a representation, then their estimates of the target sentence should be invariant to an alternative linearization of the same graph, so long as the original linearization is in principle recoverable from this alternative. If a model meets this criterion, we call it **linearization-invariant**.

[2] `catalog.ldc.upenn.edu/LDC2017T10`

[3] We use T5-Base for WebNLG and T5-Large for AMR, finding that the larger model did not benefit the WebNLG task.

[4] Less extensive experiments with the full dataset indicated the same optimal setting, although in general it is relatively robust to learning rate.

3.1 Experimental Setup

To better understand models' graph-encoding behavior, we experiment with adversarial linearization strategies in two graph-to-text settings.[5]

Permutations of AMR-Graph Linearizations Standard AMR corpora are linearized as spanning trees over the graphs in PENMAN notation (Matthiessen and Bateman 1991, see Fig. 2a). In the present work, we also linearize graphs using PENMAN, doing so for several reasons: (1) it is sufficiently flexible to accommodate significant changes to the linearization, discussed below; (2) it is more concise than sets of directed triples, both reducing training time and ensuring that inputs fit in the transformer context window; (3) the format leads to superior generation over reasonable alternatives, e.g., DFS traversal paths (Mager et al., 2020).

We will refer to the human-created linearizations in AMR corpora as CANONICAL, since annotators follow a standardized process. There is evidence that this format, in particular the relative ordering of edge types, leaks information about the associated sentence order (Konstas et al., 2017). We speculate that overparametrized models may overfit to such correlations rather than develop robust implicit graph encodings, since it has been repeatedly reported that large models use dataset shortcuts (Jia and Liang, 2017; Gururangan et al., 2018; Geva et al., 2019, among others).

As an alternative linearization, Goodman (2020) defines the RECONFIGURE operation as creating a tree from an AMR graph, where order information from the canonical linearization is ignored, except for the top node (e.g., `and` in Figs. 2a and 2b). Although it is not a labeled element in the graph, the top node conveys structural information about the sentence—for instance, it is often the main verb. Reconfiguration can include reversals of edge labels (e.g., `ARG0` to `ARG0-of`), therefore constituting a substantive change to the linearization.

We also experiment with a more drastic restructuring of the graph, where we construct a tree from a RANDOMIZED triple set alone, disregarding all order information from the canonical format (Fig. 2c). Since it remains a valid traversal of the graph, in principle a model should be able to use this information to construct the surface sentence.

We parse, reconfigure, and randomize graphs using the `Penman` library (Goodman, 2020),[6] then replace variable names with their references and remove word sense information, following Ribeiro et al. (2019).

Permutations of RDF-Triple Linearizations We follow the procedure of Ribeiro et al. (2020) to form our standard linearization: we prepend a special token to each element of the triple, and separate triples with another dedicated token. For the output sentence "Ned is the father of Rod and Todd," we would have:

In: (Ned *fatherOf* Rod), (Ned *fatherOf* Todd)
Out: `<rel> <S> Ned <V> father of <O> Rod`
`<rel> <S> Ned <V> father of <O> Todd`

For our adversarial permutation, we RANDOMIZE the ordering of the triples.

Encouraging Robustness to Linearization We train additional models with the goal of encouraging an agnosticism to graph linearization strategy. We adopt an adversarial training approach (Goodfellow et al., 2015), and alter the graph linearization presented to the model at each epoch. We argue that this scheme ought to reduce any model dependence on the human-derived annotation.

3.2 Robustness Results

For both tasks, we train the model on the canonical linearization, then evaluate on the various linearizations described in Section 3.1.

Impact of Adversarial Linearizations The CANONICAL columns of Table 2 show results for models trained on that linearization, then evaluated on permuted graph linearizations. We note a strong negative impact in models' generation capacity for both tasks, with a starker decrease for the AMR data. These results suggest that pretrained linearized models are not linearization-invariant, failing to learn robust implicit graph representations, even in the case of the much simpler WebNLG data.

The remaining columns of Table 2 show that our straightforward adversarial training technique improves robustness, with only minor cost to generation performance. This is the case even with the more drastic RANDOMIZED AMR linearization. Moreover, it only incurs a minor training time cost—for AMR, the CANONICAL, RECONFIGURE,

[5] Although "adversarial" can imply inputs specifically designed to break a model, here we use it to mean that inputs are merely *likely* to cause issues by diverging from the training order. In addition, we intend to draw parallels to adversarial training (Goodfellow et al., 2015).

[6] `github.com/goodmami/penman`

	AMR (LDC2017T10)			WebNLG			
				Seen		Unseen	
Train linearization→	CANON.	RECON.	RANDOM.	CANON.	RANDOM.	CANON.	RANDOM.
Eval. linearization↓							
CANONICAL	43.52	43.08	40.90	62.56	62.55	44.73	45.09
RECONFIGURED	33.27 / 76%	41.13 / 95%	40.33 / 99%	–	–	–	–
RANDOMIZED	22.89 / 53%	31.00 / 72%	39.80 / 97%	54.00 / 86%	59.40 / 95%	39.23 / 88%	42.35 / 94%
GNNs	–	–	–	–	–	–	–
Wang et al. (2020)	28.80	–	–	–	–	–	–
Zhao et al. (2020)	–	–	–	64.42	38.23	–	–

Table 2: BLEU under different linearizations, using T5-LARGE (AMR, development set) and T5-BASE (WebNLG, for both "seen" and "unseen" test sets). Percentages represent the decrease from the CANONICAL representation following the adversarial evaluation (i.e., they should be read columnwise).

and RANDOMIZE variants attain 40 BLEU at 2, 3, and 5 epochs, respectively.

Given that elements of canonical annotations are known to correlate with the target sentence order (Konstas et al., 2017), we do not find it surprising that the models trained *and* evaluated on the permuted linearizations show decreased performance. However, it is meaningful that the canonical linearization at evaluation time still leads to the best results, even for models trained with the randomized inputs—these models did not learn to associate the canonical ordering signal with the input graph. One possible explanation is that the earlier pretraining induces a sensitivity to input token order that persists despite the adversarial fine-tuning, but the behavior merits further exploration.

In addition, note that the canonical order does not have an explicit formal definition, and may require heuristic reverse engineering; in a real-world system, recreating a canonical graph linearization (e.g., from a knowledge-base query) for a subsequent graph-to-text model may not be straightforward. These results show it is possible to inoculate models to accommodate this sort of use case.

4 RQ2: Better Implicit Graph Encodings with Text-to-Text Scaffolding

The positive results of our adversarial training procedure (§3.2) suggest that pretrained linearized models can form a robust internal graph representation, even though they rely on linearized inputs. Under substantively different linearizations, models retain the ability to generate accurately (even the RANDOMIZE model outperforms best-in-class graph transformers; Wang et al. 2020).

Prior work, involving both GNNs and pretrained linearized models, has explored various ways of improving models' sensitivity to the structure of the input graph. To better maintain fidelity to the graph, previous graph-to-text methods incorporate additional loss terms, specialized architectures, or generation-time ranking to influence the semantic accuracy of generation: ranking outputs by the correctness of the AMR parse (Mager et al., 2020; Harkous et al., 2020), jointly "back-parsing" graphs when decoding (Bai et al., 2020), or using distinct components to model different graph traversals (Ribeiro et al., 2019).

These efforts suggest that explicitly accounting for graph structure can assist generation. Can we expand on this idea, and improve generation quality by inducing more robust internal graph representations? To answer this question, we propose secondary objectives designed to promote graph "awareness." In addition to the above graph-to-text approaches, we also draw inspiration from *denoising* methods used in language model pretraining (Raffel et al., 2020; Lewis et al., 2020), as well as syntactic *scaffolds* that support semantic tasks with an auxiliary syntax-dependent loss (Swayamdipta et al., 2018). Intermediate auxiliary pretraining has been repeatedly shown to be successful in other contexts (Phang et al., 2018; Li et al., 2019; Gururangan et al., 2020).

4.1 Experimental Setup

In particular, we propose unsupervised graph-denoising tasks that we train alongside AMR-to-text generation, following the multi-task setup of Raffel et al. (2020). For each batch, we either optimize the likelihood in Section 2 or one of the objectives described below.[7]

[7]Per-task batches proved marginally better than mixing within a batch. The scaffolding task probability is a hyperparameter, which we set to 0.5.

Masked Graph Modeling When training transformers to have wide-ranging natural language capabilities, unsupervised denoising objectives like masked language modeling have proven extremely successful (Devlin et al., 2019; Raffel et al., 2020). We argue that a similar principle ought to apply to graph understanding, and therefore apply masking directly to linearized graphs.

In masked language modeling, each word token is masked with probability 15%. Here, we mask different sets of tokens, depending on the experimental condition, always setting the probability such that 15% of *all* tokens will be masked. Specifically, we mask: all tokens in the linearized graph, the graph components alone (edge labels and parentheses), and the semantic nodes. We also experiment with standard masking of the surface sentence, which mirrors the unsupervised domain-adapted pretraining employed by Ribeiro et al. (2020).[8] For example, when masking components alone:

orig `( stupefy :ARG1 ( we ) )`
in `( stupefy <M> ( we <M> )`
out *original text*

Graph masking can also be performed on any of the linearization variants defined in Section 3.1.[9]

Graph Reordering Building on our findings from Section 3.2, we introduce a *reordering* objective. Specifically, we provide the model with a RECONFIGURED or RANDOMIZED linearization, then task the model with reconstructing the canonical version. We suspect that learning this mapping requires that the model captures the graph structure better, leading to superior graph-to-text generation. Unlike the joint re-generation approach of Mager et al. (2020), where the input graph is copied alongside the target text, our method both requires a nontrivial encoding of the graph and has the effect of augmenting the data (due to the nondeterministic reconfiguration).[10]

4.2 Scaffolding Results

We find that, overall, denoising objectives drive substantial improvements over the baseline when training on the reduced $n = 1000$ dataset (Table 3).

[8]We use MASS-style masking (Song et al., 2019) for the tokens, rather than the span-replacing of T5, as it performed somewhat better.

[9]We restrict ourselves to the RECONFIGURE setting given that early results showed little difference from RANDOMIZE.

[10]Simultaneously generating the surface text and reordering to the canonical linearization did not improve results.

	BLEU
Baseline	24.33 (0.94)
Sentence masking (MLM)	27.73 (1.29)
Graph Masking	
All tokens	28.48 (0.90)
Components	28.49 (0.48)
Nodes	**29.56** (1.05)
w/ RECONFIGURED *input*	
All tokens	**29.41** (0.90)
Components	28.34 (0.58)
Nodes	28.77 (0.80)
Reordering to canonical	
From RECONFIGURED	28.27 (0.90)
From RANDOMIZED	28.29 (0.91)

Table 3: Development set BLEU across scaffolding objectives and baselines, trained on 1000-example subsets of the AMR dataset (LDC2017T10). Mean (s.d.) over 5 seeds.

	BLEU	BS	$\mathcal{M}$
Bai et al. (2020)	34.19	-	-
Ribeiro et al. (2020)	45.80	-	-
Baseline	44.51 (0.48)	77.40 (0.36)	76.53 (0.19)
Scaffolding			
Mask nodes	45.14 (0.23)	77.75 (0.13)	76.52 (0.14)
RECON., mask all	44.89 (0.39)	77.56 (0.26)	76.54 (0.20)
Reorder from RECON.	44.86 (0.19)	77.62 (0.16)	76.34 (0.17)

Table 4: Test-set results of scaffolding objectives and baselines trained on the *full* AMR dataset (LDC2017T10). Bai et al. (2020) is a state-of-the-art graph transformer. Ribeiro et al. (2020) finetunes T5-LARGE, which we re-implement as our baseline model. BS is BertScore (Zhang et al., 2020), and $\mathcal{M}$ is the meaning component of the $\mathcal{MF}$-score (Opitz and Frank, 2020). Mean (s.d.) over 5 seeds.

In fact, using less than 3% of the full data produces results that exceed that of state-of-the-art GNN models from less than two years prior to this writing (BLEU 27.37, Ribeiro et al., 2019). Moreover, the results suggest that focusing on the graph representation itself is most important: standard sentence masking (i.e., MLM-style) is less beneficial than graph masking, although it still outperforms the baseline. Surprisingly, the various graph-masking objectives perform similarly to one another—there is little benefit to more complex strategies that specifically account for the graph structure.

While the increased generation quality from the graph-denoising methods is not drastic relative to the MLM case, we contextualize our gains by noting that other ways of promoting greater graph awareness yield similar improvements in absolute terms—and come at the cost of greater model

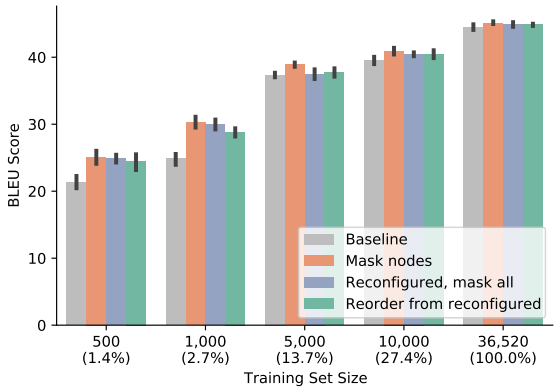

Figure 3: Test set BLEU on the AMR dataset (LDC2017T10) under different amounts of training data for selected scaffolding objectives (over 5 seeds). In low-data regimes, the scaffolding objectives substantially improve BLEU score over the baseline.

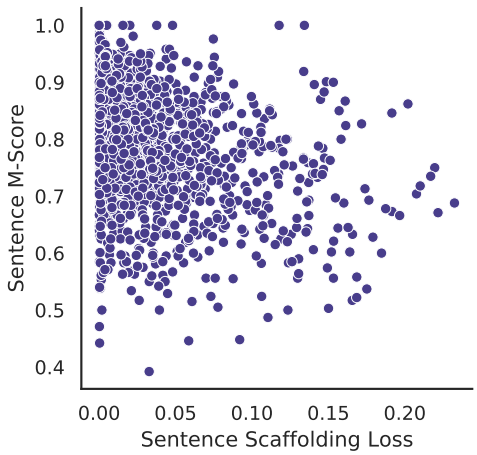

Figure 4: Sentence-level scaffolding loss and $\mathcal{M}$-score on the validation set, using a model trained with the *reordering-from-reconfigured* scaffold. $\mathcal{M}$-score is a measure of the generated sentence's semantic fidelity, and the scaffolding loss is a proxy for the graph encoding accuracy.

complexity or generation time. For instance, the use of two graph representations in Ribeiro et al. (2019) achieve a roughly 1-BLEU increase over the use of one alone.

Based on the findings from the $n = 1000$ setting (Table 3), we select three of the best-performing scaffolding objectives—*mask nodes*, *reconfigure & mask all tokens*, and *reorder from reconfigured*—and train them at $n \in \{500, 1000, 5000, 10000, N\}$. Results are shown in Fig. 3. At $n = 5000$, representing 14% of the data, the impact of scaffolding is no longer strong across all objectives. When evaluating on the full dataset, the difference is minor (Table 4). For both BLEU and BertScore, we observe slight improvement over the baseline on average for the *mask nodes* case, but it is within a standard deviation of the baseline (estimated over 5 seeds). $\mathcal{M}$-score does not vary between models, but it is also not yet established for fine-grained model selection. It appears that the increased size of the data supplants the need for scaffolding losses: the diversity of the source graphs encourages a graph-reasoning ability sufficient to generate accurate sentences. Of course, in a realistic application, hundreds or thousands of training examples are more attainable than tens of thousands. That such straightforward methods can yield strong gains is promising for future work in low-resource graph-to-text generation.

Manual Annotations In a manual analysis of 100 random model predictions, we generally observe broad agreement between the model trained with the *reordering-from-reconfigured* scaffold and the baseline (73% agreement in fidelity), both trained with the full dataset. However, in three cases, the baseline model fails to capture the order of arguments (e.g., "from y to x" when "from x to y" is correct), whereas the scaffolded model remains true to the graph (see Table 5; we did not observe instances of the reverse case). While we fail to observe "hallucinations"—material information that is not contained in the graph input—both models occasionally drop modifiers (e.g., adjectives or adverbs). Both models exhibit word-sense confusion (see the third row in Tab. 5, where "long [in length]" is substituted with "long [in duration]"). We presume this is due to the removal of word-sense suffixes during preprocessing to avoid sparsity issues (`long-03` → `long`).

4.3 Encoding Graphs and Generation Performance

The results of Section 4.2 show that the denoising scaffolds impact generation performance. If we consider the sentence-level scaffolding loss as a proxy for the quality of its implicit graph encoding, can it help explain generation fidelity? In order to determine this relationship, we quantify generation accuracy using the $\mathcal{M}$ component of the $\mathcal{MF}$-score (Opitz and Frank, 2020). It is calculated by first using an off-the-shelf parser to create an AMR graph from the generated target sentence, then by measuring the overlap with the gold source AMR (from 0 to 1)—this is equivalent to the "smatch"

Target	*Both Norway and Sweden have been spared violent terror acts but authorities in both countries have voiced concern about terrorists or terror financiers operating out of Scandinavia.*
Baseline	**Norwegian and Swedish authorities have spared Norway and Sweden** from violent acts of terror but have voiced concern about terrorists or financiers of terror operating out of Scandinavia.
Ours	Norway and Sweden have been spared terror acts of violence but Norwegian and Swedish authorities have voiced concern about terrorists or financiers of terror operating out of Scandinavia.
Target	*The 30-day simple yield fell to an average 8.19% from 8.22%; the 30-day compound yield slid to an average 8.53% from 8.56%.*
Baseline	The simple 30 day yield **fell to 8.22 percent from 8.19 percent** on average and the compound 30 day yield **slid to 8.56 percent from 8.53** percent on average.
Ours	Simple 30 day yields fell from 8.22 to an average 8.19% and compound 30 day yields slid from 8.56 to an average 8.53%.
Target	*Many young Saudi radicals have crossed the long and porous border between the Kingdom and Iraq and joined up with Sunni Muslim insurgents there.*
Baseline	Many young Saudi radicals have crossed the porous border **from Iraq to the Kingdom** and joined up with Sunni Islamic insurgents there.
Ours	Many young Saudi radicals have crossed the porous **long-term border** with Iraq and joined up with Sunni Islamic insurgents there.

Table 5: Selected predictions from the baseline and a model using the *reordering-from-reconfigured* scaffold (trained on the full data). **Colored text** denotes a semantically incorrect generation.

rescoring metric (Cai and Knight, 2013). As seen in Fig. 4, there is a substantial negative relationship (Pearson's $\rho = -0.35^*$) between these two variables, measured using outputs from the model trained with the *reordering-from-reconfigured* scaffold on the full data.

To fully operationalize the above question, we estimate a linear regression on the $\mathcal{M}$ score of predicted sentences from the validation set. In this scenario, the linear regression can quantify how much variation in predicted sentences' semantic fidelity (measured by the $\mathcal{M}$-score) can be explained by model components and target sentence characteristics. If the coefficient for the sentence-wise scaffolding loss is significant (and negative), then this would suggest that there is a relationship between the scaffolding objective and the predicted sentence's semantic fidelity.

As covariates, we include the above (logged) scaffolding loss, in addition to other metrics that have a significant independent correlation with generation quality. In particular, we use sentence-BLEU, the number of edges in the graph, graph re-entrancies, words in the target sentence, and the (also logged) sentence generation loss.[11]

We use the Bayesian information criterion (BIC) to select the model from all possible combinations of the above covariates. We find that the preferred model with p covariates, $p \in \{1, \ldots, 6\}$, includes the reordering loss in all but one case ($p = 2$), suggesting its validity as an indicator of graph fidelity above and beyond other alternatives. As seen in Table 6, it has a significant negative relationship with the $\mathcal{M}$ score, larger than that of the comparably-scaled generation loss. These results indicate that the reordering loss captures important information about the quality of the graph encoding.

[11] We eliminate outliers consisting of the bottom 0.5% of target lengths and $\mathcal{M}$-scores and the top 0.5% of the losses.

X	β
Intercept	0.7590*
Scaffolding loss (log)	-0.0094*
Generation loss (log)	-0.0088*
BLEU/100	0.0628*
Words in target	-0.0021*
BIC	-2378
Adj. R^2	0.267

Table 6: OLS regression results on validation sentence $\mathcal{M}$-score, a measure of semantic fidelity that relies on the gold AMR graph. These results indicate that the scaffolding loss explains a significant amount of the variation in the semantic fidelity of the generated sentence to the gold target. Model trained with the *reordering-from-reconfigured* scaffold. *Significance at $p < 0.001$.

5 Related Work

Pretrained Transformers for Graph-to-Text Generation Mager et al. (2020) condition GPT-2 (Radford et al., 2019) on a linearized AMR graph, then fine-tune on the corresponding surface representation text. Later work using transformers has also found success on both AMR-to-text and data-to-text tasks (Kale and Rastogi, 2020; Harkous et al., 2020; Ribeiro et al., 2020). To our knowl-

edge, across a diverse set of tasks and automated[12] metrics, a pretrained transformer of sufficient capacity will always outperform a specialized GNN, often by a large margin. Ribeiro et al. (2020), following Gururangan et al. (2020), further pretrain on additional in-domain data, using both supervised (silver AMR parses to text) and unsupervised (denoising target text) objectives.

Graph-Dependent Losses Mager et al. (2020) use various heuristics to improve fidelity. During training, they regenerate the input graph, and in inference, they parse generations and rank their consistency with the original graph. Harkous et al. (2020) instead rank with a trained classifier, and introduce additional "state embeddings" to help indicate the ordering of graph components. The encoder-decoder methods cited in the previous paragraph eschew these approaches and nonetheless perform better. In preliminary replications of the Mager et al. experiments with T5, we find that joint re-generation leads to no improvement (moreover, the longer output sequences increase training time). Experimenting with other graph-sensitive embeddings is a valuable direction for future work.

Graph Linearization Other work also studies linearizations for AMR-to-text settings. As opposed to our efforts, the focus is not on enriching or measuring models' graph encoding, but instead on determining what elements of linearization (e.g., edge labels) are necessary for generation.

Closest to our work is Konstas et al. (2017), who experiment with alternative graph traversals by randomizing the edge type order (less drastic than either RECONFIGURE or RANDOMIZE) with an LSTM-based model. Rather than randomizing at each epoch, as in our approach, they employ a *consistent* random ordering for each example during training, and do not evaluate models across different linearizations. The results help establish that LSTMs can be made agnostic to ordering, but fail to measure the extent to which models overfit to the training order (Section 3.2).

Ribeiro et al. (2020) report paired training and evaluation shuffling results (as in Table 2), but they ignore parentheses, only reodering node labels. Hence, their results cannot establish models' graph-encoding ability, instead revealing that node order is informative of word order, corroborating findings in Konstas et al. (2017). Both works, along with Mager et al. (2020), run ablations by removing parenthetical markers, finding that graph structure is necessary for strong generation.

Finally, Kedzie and McKeown (2020), appearing contemporaneously to our work, seek to control the output generation by manipulating the input linearization order, using a randomization similar to ours as an "uncontrolled" baseline. Given their focus on task-oriented dialogue planning, which uses simpler meaning representations and sentences than the AMR dataset used here (i.e., shallower graphs and limited domains), we view their work as complementary to our own.

6 Conclusion

In this work, we explore the graph-encoding ability of pretrained transformers through the lens of graph-to-text generation that relies on linearized graph inputs. First, we determine the extent to which these models are invariant to the method by which graphs are linearized, finding that models trained on the fixed, canonical linearizations fail to generalize to meaning-preserving alternatives. We rectify this shortcoming by training models on linearizations corresponding to alternative random traversals of the graph. Following prior work that has used graph-aware losses to improve generation quality, we then explore ways of improving models' sensitivity to the input graphs. Motivated by the success of denoising objectives in other text-to-text settings, we encourage robust internal graph encodings through additional scaffolding losses. Although scaffolding leads to tepid improvements in generation quality when training data is plentiful, it yields substantial gains in low-resource settings. Finally, while pretrained transformers may not learn the entirety of a graph's structure via scaffolding, our text-to-text methods take a step in the direction of increasing graph sensitivity.

7 Acknowledgments

We thank AI2 AllenNLP team members for advice throughout the project, and anonymous reviewers for their insightful comments. We also thank Leonardo Ribeiro for his publicly available implementations and assistance.

[12] Human evaluation has been less thorough, although Mager et al. (2020) report improved human judgments on AMR-to-text generation. We note similar results in our own experiments.

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.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, 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.

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.

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.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. 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 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. 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.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2020. Controllable meaning representation to text generation: Linearization and data augmentation strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185, Online. Association for Computational Linguistics.

Thomas Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of 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.

Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story ending prediction by transferable bert. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1800–1806. International Joint Conferences on Artificial Intelligence Organization.

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.

Emma Manning, Shira Wein, and Nathan Schneider. 2020. A human evaluation of AMR-to-English generation systems. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4773–4786, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Christian M.I.M. Matthiessen and John A. Bateman. 1991. *Text Generation and Systemic-functional Linguistics: Experiences from English and Japanese*. Communication in Artificial Intelligence Series. Pinter Pub Ltd.

Pablo Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1813–1817, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Juri Opitz and Anette Frank. 2020. Towards a decomposable metric for explainable evaluation of text generation from amr. arXiv:2008.08896.

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.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. arXiv:1811.01088.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. 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.

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, 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, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for Graph-to-Text generation. arXiv:2007.08426.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4596–4604, Stockholm, Sweden. PMLR.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5926–5936, Long Beach, California, USA. PMLR.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28, pages 2773–2781.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang, V. Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations (ICLR)*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, 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.

X	β
Intercept	0.2201*
Scaffolding loss (log)	$-0.0080^{\dagger}$
Generation loss (log)	$-0.0055^{\dagger}$
BLEU/100	1.5916*
Words in target	-0.0001
BIC	-2740
Adj. R^2	0.058

Table 7: OLS regression results on validation sentence $\mathcal{M}$-score, a measure of semantic fidelity that relies on the gold AMR graph. These results indicate that the scaffolding loss explains a significant amount of the variation in the semantic fidelity of the generated sentence to the gold target. Model trained with the *reordering-from-reconfigured* scaffold using 1000 training examples. *Significance at $p < 0.001$. † Significance at $p < 0.01$.

A Appendix

A.1 Hyperparameters, Computing Infrastructure, and Data Processing

We set the batch size to 32 for WebNLG, and 6 for AMR. During decoding, we use a beam size of 10 for WebNLG and 5 for AMR, following Ribeiro et al. (2020). All models were trained with a NVIDIA Quadro RTX 8000 GPU.

To maintain a reasonable batch size, long AMR inputs are truncated to 354 tokens (affecting less than 1% of examples); this step is not necessary for WebNLG. Finally, during preprocessing of the LDC2017T10 data, we differ from Ribeiro et al. (2019) and remove the `:wiki` attribute from the graph[13] both because they are not a standard component of all AMR graphs and because they increase the length of the inputs. We suspect that this change is the reason our baseline achieves slightly different results than Ribeiro et al. (2020) (Table 4).

A.2 Regression Results at $n = 1000$

In Table 7, we show OLS regression results analogous to those in Table 6 for one run of the $n = 1000$ case. While the fit is lower (a lower R^2), the scaffolding loss still has an explanatory effect.

[13]For some named entities, the graph will include a corresponding Wikipedia title.