Two Heads Are Better Than One: Exploiting Both Sequence and Graph Models in AMR-To-Text Generation

Anonymous ACL submission

Abstract

Abstract meaning representation (AMR) is a special semantic representation language that captures sentences' meaning with syntax-irrelevant graphs. AMR-to-text generation aims to generate text according to a given AMR graph and is helpful in various downstream NLP tasks. Existing AMR-to-text generation methods roughly fall into two categories, each with pros and cons. The sequence-to-sequence models, especially pretrained language models (PLMs), have good text generation ability but cannot cope well with the structural information of AMR graphs. The graph-to-sequence models utilize graph neural networks (GNNs), showcasing complementary strengths and limitations. Combining both methods could harness their strengths; yet, merging a GNN with a PLM is non-trivial. In this paper, we propose DualGen, a dual encoder-decoder model that integrates a specially designed GNN into a sequence-to-sequence PLM. We conduct extensive experiments, human evaluation, and a case study, finding that DualGen achieves the desired effect and yields state-of-the-art performance in the AMR-to-text generation task. We also show it outperforms the most potent general-purpose PLMs, LLaMA and GPT-4.

1 Introduction

Abstract meaning representation (AMR) is a semantic representation language representing sentences' meaning as rooted, directed, and labeled graphs, free from syntactic idiosyncrasies (Banarescu et al., 2013). In AMR graphs, nodes depict entities, events, and properties, while edges denote node relationships. Figure 1 exemplifies an AMR graph with two formats.

AMR-to-text generation aims to generate text with the same meaning as an AMR graph. It is a well-established task that is useful in various downstream applications, including text summarization (Liu et al., 2015; Takase et al., 2016), machine translation (Jones et al., 2012; Song et al., 2019), and information extraction (Zhang and Ji, 2021). Figure 1 illustrates AMR-to-text generation.

Previous studies of AMR-to-text generation employ two kinds of architectures. The first one is the sequence-to-sequence (s2s) model, which uses a sequence encoder to process the linearized AMR graphs and a sequence decoder to generate text (Konstas et al., 2017; Cao and Clark, 2019). Benefiting from the strong language ability of pretrained language models (PLMs) (Lewis et al., 2020; Raffel et al., 2020), recent s2s AMR-to-text models have achieved leading results (Ribeiro et al., 2021a; Bevilacqua et al., 2021; Bai et al., 2022). However, linearized AMR graphs that s2s models take as inputs suffer from information loss, resulting in reduced performance (Ribeiro et al., 2021b; Song et al., 2018; Beck et al., 2018).

The second one is the graph-to-sequence (g2s) model (Song et al., 2018, 2020; Beck et al., 2018; Guo et al., 2019), which consists of a graph neural network (GNN) encoder and a sequence decoder. Different from s2s models, g2s models can capture the complete structural information of AMR graphs with GNN encoders. They usually outperform un-pretrained s2s models (Song et al., 2020), particularly for complex graphs. However, because g2s models cannot be pretrained on corpora, they exhibit weaker overall performance than PLMs.

In this paper, to combine the strengths of both s2s and g2s models, we introduce DualGen, a dual encoder-decoder model, using BART (Lewis et al., 2020) as the foundation model.[1] Based on the s2s architecture of BART, we add a GNN encoder. In this way, DualGen is expected to take complete information of AMR graphs while benefiting from the strong language capabilities of PLMs.

Integrating a GNN encoder into a pretrained Transformer-based PLM is non-trivial. First, all

[1] DualGen is applicable to other Transformer-based PLMs.

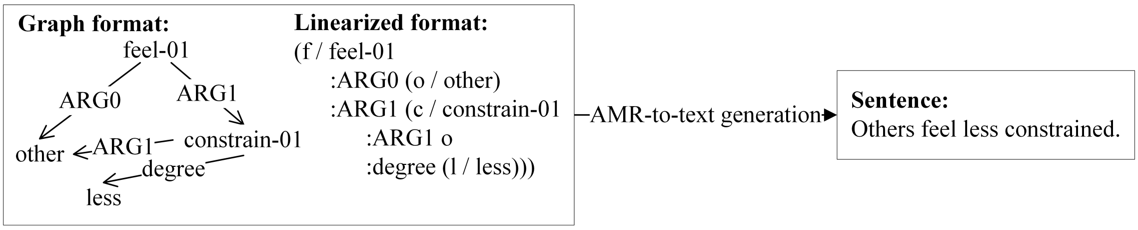

Figure 1: Illustration of two equivalent formats of an AMR graph and the AMR-to-text generation task. “ARG0”, “ARG1”, and “degree” are edge labels. In linearized format, nodes are denoted by abbreviations, e.g., “f” denotes “feel-01”. The linearized format is indented for better readability.

existing AMR datasets are inadequate to train a GNN encoder of a similar size as the sequence encoder from scratch. Second, no pretrained GNNs tailored for language tasks are available; prior studies employing dual-encoders for NLP tasks initiate GNN training from the ground up. To address these challenges, we design a specialized GNN encoder that can be initialized with PLM parameters and seamlessly integrated with the PLM.

Experiment results on datasets AMR2.0 and AMR3.0 demonstrate that DualGen outperforms the state-of-the-art method (Bai et al., 2022) and the most potent PLMs, LLaMA and GPT-4 across multiple metrics. We conduct quantitative and qualitative analyses, demonstrating that DualGen excels in processing graph structures while maintaining text generation quality on par with PLMs. We find that DualGen particularly excels in handling complex graphs compared with `s2s` models, showing that DualGen combines the strengths of both `g2s` and `s2s` models. We conduct a human evaluation and a case study that further validate these findings.

2 Related Work

AMR-to-text generation. AMR-to-text generation involves transforming AMR graphs into the corresponding text. One approach for AMR-to-text generation employs a sequence-to-sequence (`s2s`) model that consists of a sequence encoder and a sequence decoder. The first neural model for this task (Konstas et al., 2017) uses stacked bidirectional LSTM, while recent studies adopt the Transformer architecture (Vaswani et al., 2017) and employ pretrained language models (PLMs). Ribeiro et al. (2021a) proposes adaptive pretraining, while Bevilacqua et al. (2021) explores linearization methods. Mager et al. (2020) introduces an additional rescoring stage and explores joint probability. Bai et al. (2022) employs graph pretraining. The sequence encoder can only take linearized AMR graphs as input. However, linearization causes a loss of graph structure information.

Another approach employs a graph-to-sequence (`g2s`) model, which consists of a graph neural network (GNN) encoder and a sequence decoder. Various GNN encoders have been explored, including gated GNN (Beck et al., 2018), graph LSTM (Song et al., 2018), graph convolutional network (Guo et al., 2019), and graph attention network (Song et al., 2020; Koncel-Kedziorski et al., 2019; Cai and Lam, 2020). While the `g2s` model can effectively handle graph structures, it cannot process text. Consequently, it cannot be pretrained by textual data, which limits its language generation ability.

To combine the strengths of `s2s` and `g2s` models, Ribeiro et al. (2021b) employs a PLM-based approach, incorporating a graph convolutional network (GCN) adapter following the sequence encoder for better graph handling. Unlike DualGen, which uses a dual encoder architecture, Ribeiro et al. (2021b) employs an un-pretrained GCN and only fine-tunes the GCN while keeping others frozen. Later experimental results show the superiority of our method over this model.

Dual encoder architecture. Dual encoder architecture is widely used in NLP. In generative models, prior work mainly employs un-pretrained models. For instance, Junczys-Dowmunt et al. (2018) utilized two un-pretrained encoders and a decoder to recover translation errors. Zhang et al. (2021) applied two un-pretrained encoders and two un-pretrained decoders for dialogue summarization. For pretrained models, Dou et al. (2021) employs two Transformer encoders and a Transformer decoder for text summarization. However, to our knowledge, there has been no prior dual encoder-decoder model that simultaneously uses distinct architectures for the two encoders while utilizing pretrained models for both encoders. Also, no prior research has employed the dual encoder architec-

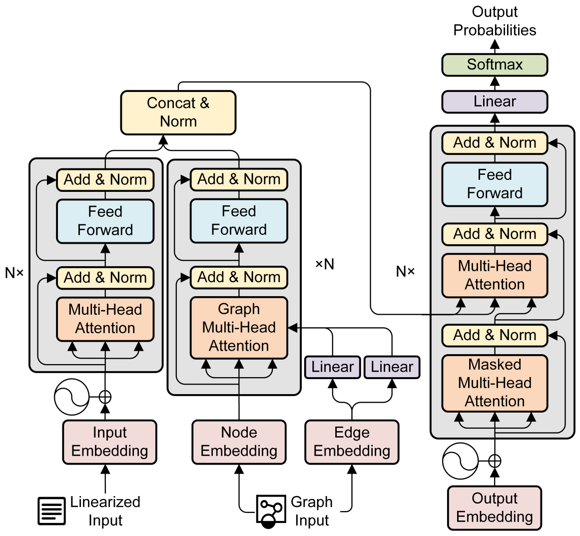

Figure 2: The architecture of the DualGen model.

ture for AMR-to-text generation.

For non-generative tasks, dual encoder architecture is employed in tasks including similarity measurement (Mueller and Thyagarajan, 2016; Yang et al., 2018), context-based candidate selection (Shyam et al., 2017), and information retrieval (Pang et al., 2017).

3 Method

In this section, we provide a detailed description of DualGen. We convert the AMR graph into a linearized and graphical format (Section 3.1), which is then fed into the dual encoder-decoder model (Section 3.2). Following prior research, we employ a two-stage training (Section 3.3).

3.1 Data Processing

We replace the nodes of AMR graphs with their original labels, omitting the PropBank (Palmer et al., 2005) indexes. For example, the node `f/feel-01` in Figure 1 is transformed into `feel`.

We use the DFS-based approach as per Bevilacqua et al. (2021) to linearize. For tokenization, we follow the BART method for both encoders, similarly tokenizing the linearized AMR sequence, nodes, and edges. This allows us to calculate sequence and graph embeddings with shared embedding parameters across the two encoders.

3.2 Model Architecture

DualGen adopts a dual encoder-decoder architecture comprising a Transformer-based sequence encoder, a GNN-based graph encoder, and a Transformer-based sequence decoder, as depicted in Figure 2. The sequence and graph encoder take linearized and graph AMRs as input, respectively.

Sequence encoder: The sequence encoder is a Transformer encoder, initialized with BART parameters, as illustrated in the left part of Figure 2. It accepts the linearized AMR as its input.

Graph embeddings: The graph embeddings comprise node and edge embeddings, which share parameters with the sequence encoder and the sequence decoder embeddings. For a token t in the vocabulary, its word embedding is $\mathbf{t} \in \mathbb{R}^{d_{\text{embed}}}$.

Given an AMR graph $\mathbf{G} = \langle \mathbb{V}, \mathbb{E} \rangle$, where $\mathbb{V}$ is the node set and $\mathbb{E}$ is the edge set. Each node and edge is labeled with one or more words. The words are divided into multiple tokens during tokenization. These tokens are subsequently used to generate embeddings for nodes and edges. A node $v \in \mathbb{V}$ is denoted by l_v tokens $t_1^v, t_2^v, \cdots, t_{l_v}^v$. An edge $e \in \mathbb{E}$ is denoted by m_e tokens $t_1^e, t_2^e, \cdots, t_{m_e}^e$.

As Figure 3 shows, for a node $v \in \mathbb{V}$, its node embedding is the average embedding of all its corresponding tokens $\mathbf{v} = \frac{1}{l_v} \sum_{k=1}^{l_v} \mathbf{t}_k^v$.

To facilitate two-way information exchange along edges, we introduce two linear projections from $\mathbb{R}^{d_{\text{embed}}}$ to $\mathbb{R}^{d_{\text{edge}}}$ for forward and backward edges, defined by matrices W^F, W^B and bias $\mathbf{b}^F, \mathbf{b}^B$. For an edge e from node s_e to t_e, its forward and backward edge embeddings are:

$$\begin{cases} \mathbf{e}^{fwd} = (\frac{1}{m_e} \sum_{k=1}^{m_e} \mathbf{t}_k^e) W^F + \mathbf{b}^F \\ \mathbf{e}^{bwd} = (\frac{1}{m_e} \sum_{k=1}^{m_e} \mathbf{t}_k^e) W^B + \mathbf{b}^B \end{cases} \tag{1}$$

AMR graphs are acyclic, ensuring at most one edge connects any given pair of nodes. Therefore, the edge embedding is well-defined:

$$\forall s, t \in \mathbb{V}, \mathbf{e}_{s,t} = \begin{cases} \mathbf{e}^{fwd} & \text{if } s_e = s \text{ and } t_e = t \\ \mathbf{e}^{bwd} & \text{if } t_e = s \text{ and } s_e = t \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{2}$$

Graph encoder: The graph encoder resembles the Transformer encoder, as shown in Figure 2. However, it incorporates a unique multi-head attention mechanism for graphs, as Figure 4 depicts. The node embedding is $V^{\text{n}} = K^{\text{n}} = Q^{\text{n}} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_{|\mathbb{V}|} \end{bmatrix}^\top$ and the edge embedding for a given node v is $\mathbf{E}_v = \begin{bmatrix} \mathbf{e}_{v,1} & \mathbf{e}_{v,2} & \cdots & \mathbf{e}_{v,|\mathbb{V}|} \end{bmatrix}^\top$.

We present a graph attention mechanism inspired by the work of Song et al. (2020). To leverage

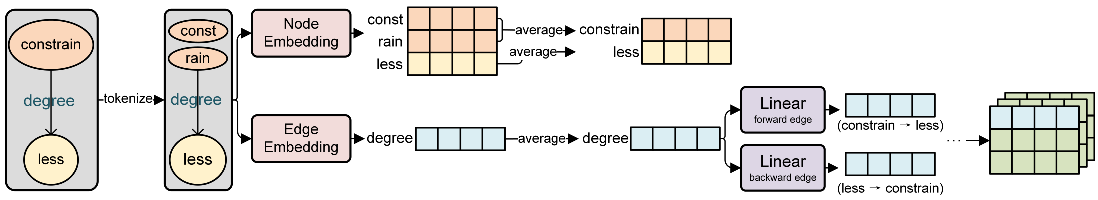

Figure 3: An example of graph embeddings. The nodes are "constrain" and "less". The label of the edge is "degree".

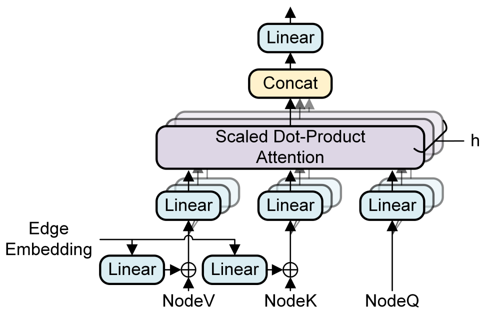

Figure 4: Graph multi-head attention.

edge information, we incorporate edge embeddings into the node value and node key components through two distinct linear projections from $\mathbb{R}^{d_{\text{edge}}}$ to $\mathbb{R}^{d_{\text{node}}}$ defined by matrices W_e^V, W_e^K and bias terms $\mathbf{b}^V, \mathbf{b}^K$, respectively. As discussed by Cai and Lam (2020), we treat the graph as fully connected with specialized edge labels, facilitating information exchange. The formulation of this attention mechanism is as follows:

$$\begin{cases} V_i = V^{\text{n}} + \mathbf{E}_v W_e^V + \mathbf{b}^V \\ K_i = K^{\text{n}} + \mathbf{E}_v W_e^K + \mathbf{b}^K \\ Q_i = Q_i^{\text{n}} \end{cases} \tag{3}$$

$$\begin{aligned} &\text{GraphAttention}(Q, K, V)_i = \\ &\qquad \text{Multihead-Attention}(Q_i, K_i, V_i) \end{aligned} \tag{4}$$

The graph encoder is "pretrained" in a unique way. Its structure is similar to the Transformer encoder, allowing the central part to be initialized by pretrained BART parameters, except for the two additional linear projections depicted in Figure 4. This initialization process can enhance the language capabilities of the graph encoder.

Hidden representation merging: To merge the hidden representations from the two encoders, we concatenate the two hidden representations and apply layer normalization (Ba et al., 2016).

Sequence decoder: The sequence decoder in DualGen follows the pretrained BART decoder, as illustrated in Figure 2.

Dataset	Train	Dev	Test
AMR2.0	36,521	1,368	1,371
AMR3.0	55,635	1,722	1,898

Table 1: Statistics of AMR2.0 and AMR3.0.

3.3 Two-Stage Training

Existing AMR datasets have limited size and may be inadequate for training effective graph encoders. We employ a two-stage training strategy to align with prior research (Bai et al., 2022; Bevilacqua et al., 2021; Ribeiro et al., 2021a).

For the first stage, we employ model-generated silver data for pretraining. We randomly sample 200k entries from the Gigaword dataset (LDC2011T07) (Parker et al., 2011). We use the AMR parsing model `parse_xfm_bart_base` from amrlib (Jascob, 2020) to generate the corresponding AMR graphs and remove those not following AMR rules. For the second stage, we employ existing AMR datasets for fine-tuning.

4 Experiments

We assess the performance of DualGen compared to state-of-the-art models on authoritative datasets. We investigate the influence of graph complexity and evaluate the models' capacity to process graph structure through human evaluation. Additionally, we compared DualGen's performance with the most potent PLMs, including LLaMA (Touvron et al., 2023) and GPT-4 (OpenAI, 2023).

4.1 Dataset

Following previous works (Bai et al., 2022; Ribeiro et al., 2021b; Bevilacqua et al., 2021) , we evaluate our model using the two most prevalent and authoritative AMR datasets, AMR2.0 (LDC2017T10)(Knight et al., 2017) and AMR3.0 (LDC2020T02) (Knight et al., 2016) datasets. Table 1 presents dataset statistics for both.

4.2 Evaluation Metrics

Following previous work (Bai et al., 2022; Bevilacqua et al., 2021), we use three automated evaluation metrics: BLEU (Papineni et al., 2002), Meteor (Banerjee and Lavie, 2005), and chrF++ (Popović, 2015). We also perform a human evaluation to assess language quality and semantic similarity.

4.3 Compared Models

We select the following representative methods for comparison, including the state-of-the-art approach. (1) Guo et al. (2019), a g2s model that uses densely connected graph convolutional networks with attention mechanisms; (2) Song et al. (2020), a g2s model that uses a structure-aware Transformer encoder with vectorized edge information; (3) Ribeiro et al. (2021a), a s2s model based on PLMs [2]; (4) Bevilacqua et al. (2021), a s2s model based on PLMs that uses special linearization method and vocabulary; (5) Ribeiro et al. (2021b), a s2s model based on PLMs that includes a graph convolutional network adapter; (6) Bai et al. (2022), the state-of-the-art method, a s2s model based on PLMs that uses a unified graph pretraining framework.

4.4 Settings

We use the BART-large model (Lewis et al., 2020) as the base model of DualGen. DualGen comprises 12 sequence encoder layers, 12 graph encoder layers, and 12 sequence decoder layers. The sequence encoder and decoder need minimal fine-tuning since they share BART's architecture; the graph encoder requires more fine-tuning with a different architecture. Consequently, we employ three distinct learning rates for the three components.

We select hyperparameters by validation set performance. For silver-data training, the model undergoes 6,000 steps over 20 epochs with updates every 8 steps, with a scale tolerance of 0.5 to filter out low-quality data. For fine-tuning, the model undergoes 13,000 steps over 65 epochs, with updates every 4 steps. In both phases, the initial learning rates are 1×10^{-6} for the sequence encoder, 4×10^{-5} for the graph encoder, and 8×10^{-6} for the sequence decoder. We use Adam (Kingma and Ba, 2015) as optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$, and a clipping threshold of 0.1.

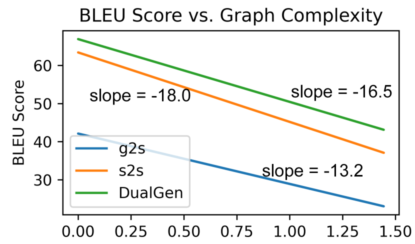

Figure 5: The impact of graph complexity on model performance.

4.5 Main Results

Table 2 shows the results. DualGen outperforms all other methods on all three metrics. Compared to the state-of-the-art model (Bai et al., 2022), it achieves a 1.8-point improvement in BLEU, 2.3 points in Meteor, and 0.8 points in chrF++ on AMR2.0 dataset. Similarly, on AMR3.0, DualGen achieves a 2.6-point increase in BLEU, 2.8 points in Meteor, and 1.1 points in chrF++.

Models utilizing s2s PLMs consistently outperform un-pretrained g2s models. This suggests that pretraining on large corpora significantly enhances model performance, confirming the validity of our choice to employ PLM-based methods.

Utilizing silver data leads to better performance than methods not incorporating such augmentation. This highlights the effectiveness of our use of model-generated silver data.

Compared with Ribeiro et al. (2021a), which shares the same architecture and method as DualGen without graph encoders, DualGen consistently achieves superior performance. This underscores the effectiveness of incorporating a graph encoder in AMR-to-text generation. Further details of ablation studies can be found in Appendix A.

4.6 Impact of Graph Complexity

To determine the robustness of DualGen across varying levels of graph complexity and its effectiveness in processing graph structure, we investigate how graph complexity affects the performance of g2s models, s2s models, and DualGen. We choose Guo et al. (2019) and Ribeiro et al. (2021a)[3] as the representative g2s and s2s models, respectively.

A higher edge-to-node ratio suggests a more

[2] Ribeiro et al. (2021a) uses the original Bart which shares the same architecture and training method as DualGen without graph encoders, with only minor vocabulary differences.

[3] We use the model in Ribeiro et al. (2021a) without silver data pretraining, which is the original Bart model. It shares architecture and method with DualGen without graph encoder.

Dataset	Model	Silver Data	BLEU	Meteor	chrF++
AMR2.0	Guo et al. (2019)†	0	27.6	33.1‡	57.3
	Song et al. (2020)†	0	34.2	38.0	68.4‡
	Ribeiro et al. (2021a) ($\text{Bart}_{\text{large}}$)	0	43.5	42.9	73.9‡
	Ribeiro et al. (2021a) ($\text{Bart}_{\text{large}}$)	200k	44.7	43.7	-
	Bevilacqua et al. (2021) ($\text{Bart}_{\text{large}}$)	200k	45.9	41.8	74.2
	Ribeiro et al. (2021b) (T5_{base})	0	44.0	41.9‡	71.2
	Ribeiro et al. (2021b) (T5_{large})	0	46.6	42.8‡	72.9
	Bai et al. (2022)($\text{Bart}_{\text{base}}$)	200k	46.6	41.4	74.6
	Bai et al. (2022)($\text{Bart}_{\text{large}}$)	200k	49.8	42.6	76.2
	DualGen ($\text{Bart}_{\text{large}}$)	0	47.9	43.3	74.6
	DualGen ($\text{Bart}_{\text{large}}$)	200k	**51.6**	**44.9**	**77.0**
AMR3.0	Song et al. (2020)†	0	37.9‡	39.4‡	70.8‡
	Bevilacqua et al. (2021) ($\text{Bart}_{\text{large}}$)	200k	46.5	41.7	73.9
	Ribeiro et al. (2021b) (T5_{base})	0	44.1	42.8‡	73.4
	Ribeiro et al. (2021b) (T5_{large})	0	48.0	44.0‡	73.2
	Bai et al. (2022)($\text{Bart}_{\text{base}}$)	200k	45.9	40.8	73.8
	Bai et al. (2022)($\text{Bart}_{\text{large}}$)	200k	49.2	42.3	76.1
	DualGen ($\text{Bart}_{\text{large}}$)	0	49.5	43.9	75.7
	DualGen ($\text{Bart}_{\text{large}}$)	200k	**51.8**	**45.1**	**77.2**

Table 2: Results of AMR-to-text generation for the AMR2.0 and AMR3.0 test sets. Models marked with † are g2s models. We calculate results marked with ‡ as they are not reported in the original paper. The Silver Data column indicates how many data entries are used for pretraining. The best results within each dataset are denoted in bold.

complex graph with intricate node relationships. We use this ratio to measure graph complexity and conduct regression analysis to examine its connection with model performance, measured by the BLEU score. A steeper regression slope indicates better graph processing ability. A higher regression line indicates superior overall performance.

Figure 5 presents the regression results. From the regression slopes, we infer that g2s has the best ability to process graph, and DualGencomes next, performing better than s2s, showcasing the usefulness of the additional graph encoder.

Regarding language skills measured by intercepts, s2s and DualGen perform similarly, surpassing g2s. This confirms the dual encoder-decoder architecture maintains comparable language skills to PLM-based s2s methods.

4.7 Model Failures

To explore the shortcomings of the above three models Guo et al. (2019), Ribeiro et al. (2021a), and DualGen, we analyzed the failed cases. Entries with a BLEU score below 25 are considered failed.

The results are presented in Table 3. Compared with g2s and s2s models, for failed instances, DualGen exhibits fewer edges and nodes, fewer node reentrance, and lower graph depth, indicating more superficial graph structures. As the s2s model is the same as DualGen without graph encoders, the results imply that DualGen is less sensitive to intricate graph architectures. This underscores the efficacy of the graph encoder in processing AMR graphs.

4.8 Human Evaluation

To further assess the performance of the models, we conduct a human evaluation. Following previous work (Ribeiro et al., 2021b,a), we randomly select 100 AMR graphs from the AMR2.0 test set. Six annotators with an English background assessed these samples, scoring 0 to 10 for language quality and semantic similarity. Each entry was assigned to three annotators to assess the performance of the six tested models. Further details can be found in Appendix C. Table 4 shows human evaluation results.

For language quality, PLM-based s2s approaches consistently outperform the g2s method, indicating superior language proficiency. DualGen achieves language quality scores comparable to other PLM-based methods, affirming its similar language capabilities to PLMs.

Model	Architecture	# Failed	$\overline{\text{Edge}}$	$\overline{\text{Node}}$	$\overline{\text{Reentrance}}$	$\overline{\text{Depth}}$
Guo et al. (2019)	g2s	751	19.37	18.55	1.82	3.39
Ribeiro et al. (2021a)	s2s	347	18.68	17.91	1.77	3.23
DualGen	dual encoder	**260**	**18.22**	**17.65**	**1.57**	**3.10**

Table 3: Results of model failure analysis. All models are trained without silver data. # Failed indicates the number of failed cases. $\overline{\text{Edge}}$, $\overline{\text{Node}}$, $\overline{\text{Reentrance}}$, and $\overline{\text{Depth}}$ indicate the average number of edges, average number of nodes, average number of reentrance nodes, and average graph depth of the failed cases, respectively.

Model	Architecture	Silver Data	quality	similarity
Song et al. (2020)	g2s	0	8.22	8.01
Ribeiro et al. (2021a) ($\text{Bart}_{\text{large}}$)	s2s	0	9.26	8.26
Bevilacqua et al. (2021)($\text{Bart}_{\text{large}}$)	s2s	200k	9.11	8.35
Bai et al. (2022) ($\text{Bart}_{\text{large}}$)	s2s	200k	<u>9.42</u>	8.57
DualGen ($\text{Bart}_{\text{large}}$)	dual encoder	0	9.29	8.59
DualGen ($\text{Bart}_{\text{large}}$)	dual encoder	200k	<u>9.38</u>	**8.98**

Table 4: Results of human evaluation on the AMR2.0 test set. Our model significantly outperforms comparison methods, as indicated by T-tests with a significance level of $p < 0.05$. The best language quality scores are underlined; the best semantic similarity scores are in bold.

Model	SD	BLEU	Meteor	chrF++
LLaMA	0	38.9	40.3	72.2
	200k	44.5	41.9	73.8
DualGen	0	47.9	43.3	74.6
	200k	51.6	44.9	77.0

Table 5: Results of fine-tuned LLaMA-2-7B on the AMR2.0 dataset. SD stands for Silver Data.

Model	shot	BLEU	Meteor	chrF++
GPT-3.5	0	6.9	25.4	49.8
GPT-3.5	3	14.6	28.6	53.4
GPT-3.5	8	17.7	29.9	55.1
GPT-3.5	10	18.4	29.9	55.5
GPT-3.5	15	18.5	30.3	56.2
GPT-4	15	**30.8**	**36.7**	**64.7**

Table 6: Results of few-shot prompted GPT-3.5 and GPT-4 on the AMR2.0 test set.

Regarding semantic similarity, DualGen without silver data pretraining achieves a higher similarity score than other un-pretrained methods. DualGen with silver data pretraining significantly outperforms all other methods, demonstrating the benefits of the dual encoder architecture.

4.9 Comparison with the Most Powerful PLMs

Recently, LLMs have demonstrated impressive language generation capabilities on various NLP tasks. We evaluate the performance of LoRA(Hu et al., 2021) fine-tuned LLaMA(Touvron et al., 2023), GPT-3.5(OpenAI, 2021), and GPT-4(OpenAI, 2023) in AMR-to-text generation using the AMR2.0 test dataset. The results are presented in Table 5 and Table 6. Further details can be found in Appendix B.

LoRA fine-tuned LLaMA-2-7B model performs comparably with fully fine-tuned smaller models Ribeiro et al. (2021a), and performs worse than DualGen. With a s2s architecture, fine-tuned LLaMA cannot use complete graph structure information and struggles with entity relations.

Although GPTs perform exceptionally well in many language-related tasks, they encounter difficulties in AMR-to-text generation without fine-tuning. We design prompts for in-context learning with a maximum of 15 shots due to the token limitation. GPT-4 with 15 shots outperforms all other LLM settings but lags significantly behind fine-tuned PLM methods.

To conclude, LLMs, including GPTs and LLaMA, are not proficient in AMR-to-text generation, with DualGen yielding significantly better results after training. Exploring smaller models for these specific tasks is worthwhile, as LLMs cannot substitute these models.

<table>
<tr><th>AMR Graph</th><th>Text</th></tr>
<tr><td rowspan="5">(a / agitate-01
:ARG0 (s2 / spring-up-02
:ARG1 (s / scene
:quant (m2 / many)
:mod (h / heroic)
:mod (t2 / tragic)
:topic (a2 / and
:op1 (s3 / spear
:ARG1-of (s4 / shine-01))
:op2 (h2 / horse
:ARG1-of (a3 / armor-01)))
:ARG2-of (s5 / stir-02))
:location (m3 / mind
:poss i))
:ARG1 (s6 / string
:poss (m / memory
:poss (i / i))
:mod (t4 / thing
:ARG1-of (t3 / think-01)))
:frequency (o / occasional))</td><td>Reference answer: the thought-strings of my memory have been agitated from time to time - many heroic, stirring, and tragic scenes of shining spears and armored horses spring up in my mind.</td></tr>
<tr><td>Song et al. (2020): occasionally, my memory has been touched by many heroic scene in my mind springing up in shiney spears and armored horses.</td></tr>
<tr><td>Ribeiro et al. (2021a): my memory strings of thoughts are occasionally agitated by the stirring up of many heroic and tragic scenes of shining spears and armored horses in my mind.</td></tr>
<tr><td>Bai et al. (2022): many heroic and tragic scenes that spring up in my mind of stirring spears and armored horses occasionally agitate the strings of thought in my memory.</td></tr>
<tr><td>DualGen: occasionally, my memory's string of thoughts is agitated by the many stirring, heroic and tragic scenes of shining spears and armored horses that spring up in my mind.</td></tr>
</table>

Table 7: Case study. The AMR graph is illustrated in its linearized format on the left side of the table. On the right, we present the reference answer from the AMR3.0 dataset alongside the model-generated answers. Problematic text is underlined.

4.10 Case Study

Table 7 presents a case study from the AMR2.0 test set, highlighting the superior performance of DualGen. It showcases sequences generated by both DualGen and the baseline g2s (Song et al., 2020) and s2s models (Ribeiro et al., 2021a; Bai et al., 2022), alongside the reference answer provided by the AMR2.0 dataset.

The answer generated by Song et al. (2020) contains grammatical errors, such as "many heroic scene" instead of "many heroic scenes". Furthermore, the phrase "in my mind springing up in shiny spears and armored horses" is unclear and ambiguous. These examples highlight the limited language proficiency of the g2s model.

The s2s PLM-based methods Ribeiro et al. (2021a); Bai et al. (2022) are proficient in generating grammatically correct and coherent sentences. However, Ribeiro et al. (2021a) overlooks specific entities, such as "spring up'. Both methods misinterpret edge relationships, failing to recognize that "heroic", "tragic", and "stirring up" should be juxtaposed. Furthermore, Bai et al. (2022) mistakenly employ "stirring" instead of "shining" to modify "spears".

Our model, DualGen, is free of grammatical errors, generates high-quality sentences, and accurately represents all node entities and edge relations. This demonstrates that our PLM-based model possesses strong language skills and simultaneously excels in managing graph structures.

5 Conclusion

We explore a dual encoder-decoder architecture model for the AMR-to-text generation task. This model comprises a graph encoder, a sequence encoder, and a sequence decoder. Our model's architecture is specially designed to be compatible with Transformer encoder-decoder architecture, and all three primary components, including the graph encoder, can be initialized by PLMs such as BART (Lewis et al., 2020), GPT2 (Radford et al., 2019), and T5 (Raffel et al., 2020). This dual encoder-decoder architecture enhances the model's capability to process graph structure information while maintaining language proficiency on par with PLMs. Our model surpasses the current state-of-the-art methods across multiple benchmarks for the AMR-to-text task.

6 Limitations

For the datasets, we only use AMR2.0 (Knight et al., 2017) and AMR3.0 (Knight et al., 2016) as golden AMR-text datasets. Although some prior works (Bai et al., 2022) use three additional datasets: The Little Prince (TLP), the Bio datasets from `https://amr.isi.edu/index.html`, and the New3 dataset (part of AMR3.0 but not AMR2.0), we omit them from our analysis as their size is relatively small and they are used for out-of-distribution evaluations in previous studies, which is not the focus of our paper.

For the experiments, we only test our dual encoder-decoder method based on the BART-large(Lewis et al., 2020) pretrained language model. We choose BART because it is suitable for generation tasks and has been frequently used in previous studies.

For Section 4.9 where we use LLaMA(Touvron et al., 2023) for comparison, we only tested the performance of the LoRA-finetuned model. We do not test the performance of fully-finetuned LLaMA.

7 Ethical Statement

We anticipate no ethics-related concerns in our research. All datasets and models used are open-source, and we will release our code publicly to facilitate reproducibility.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

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

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7464–7471.

Kris Cao and Stephen Clark. 2019. Factorising AMR generation through syntax. 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 2157–2163, Minneapolis, Minnesota. Association for Computational Linguistics.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Brad Jascob. 2020. amrlib. `https://github.com/bjascob/amrlib`.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING 2012*, pages 1359–1376, Mumbai, India. The COLING 2012 Organizing Committee.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of*

ACL 2018, System Demonstrations, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

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*.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2016. Abstract meaning representation (amr) annotation release 3.0. Online.

Kevin Knight, Bianca Badarau, Claire Bonial Laura Baranescu, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2017. Abstract meaning representation (amr) annotation release 2.0. Online. LDC Catalog Number: LDC2017T10, ISBN: 1-58563-802-1, ISLRN: 335-339-972-504-9, Release Date: June 15, 2017.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. 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 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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.

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

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.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

OpenAI. 2021. Gpt-3.5. `https://www.openai.com`.

OpenAI. 2023. Gpt-4. `https://openai.com/research/gpt-4`.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 257–266, New York, NY, USA. Association for Computing Machinery.

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.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition. Online. LDC Catalog Number: LDC2011T07, ISBN: 1-58563-581-2, ISLRN: 911-942-430-413-0, Release Date: June 17, 2011.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. *J. Mach. Learn. Res.*, 21(1).

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

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

Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. 2017. Attentive recurrent comparators. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3173–3181. PMLR.

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, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998, Online. Association for Computational Linguistics.

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.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. In *Proceedings of the Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.

Xinyuan Zhang, Ruiyi Zhang, Manzil Zaheer, and Amr Ahmed. 2021. Unsupervised abstractive dialogue summarization for tete-a-tetes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14489–14497.

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

A Ablation study

To further demonstrate the capabilities of each component within DualGen, we conducted an ablation study. This involved examining the performance of different model variations:

- DualGen w/o SE: DualGen without the sequence encoder;
- DualGen w/o GE: DualGen without the graph encoder;
- DualGen w/o GP: DualGen with the graph encoder trained from scratch.
- DualGen w/o SE w/o GP: DualGen without the sequence encoder, with the graph encoder trained from scratch.

We use GP to indicate graph pretraining, SE to indicate sequence encoders, and GE to indicate graph encoders. The outcomes for the above four model variants are presented in Table 8.

DualGen w/o SE w/o GP and DualGen w/o GP exhibit notably poor performance. This is because the AMR datasets are insufficient for training, given their limited size compared to the enormous size of the graph encoders. The training subsets of the AMR2.0 and AMR3.0 datasets comprise 36k and 56k entries, respectively. In contrast, the graph encoders contain 152M trainable parameters, akin in size to the Bart large encoders. In comparison, the full DualGen model encompasses 560M parameters, while the previously best-performing g2s model (Song et al., 2020) comprises a total of 62M parameters. Consequently, when fine-tuned on the AMR datasets, DualGen w/o SE w/o GP and DualGen w/o GP scarcely acquire meaningful information, consistently yielding a low BLEU score. This underscores the efficacy of our approach in "pretraining" the graph encoder in a specialized

Dataset	Model	Silver Data	BLEU	Meteor	chrF++
AMR2.0	DualGen w/o SE w/o GP	0	0.0	1.0	3.4
	DualGen w/o GP	0	0.1	4.4	15.5
	DualGen w/o SE	0	22.1	31.4	58.7
	DualGen w/o GE	0	43.8	42.1	72.1
	Ribeiro et al. (2021a)	0	43.5	42.9	73.9‡
	DualGen	0	47.9	43.3	74.6
	DualGen	200k	**51.6**	**44.9**	**77.0**
AMR3.0	DualGen w/o SE w/o GP	0	0.0	1.3	3.3
	DualGen w/o GP	0	0.0	1.0	4.1
	DualGen w/o SE	0	22.2	31.6	58.2
	DualGen w/o GE	0	45.7	42.9	73.4
	DualGen	0	49.5	43.9	75.7
	DualGen	200k	**51.8**	**45.1**	**77.2**

Table 8: Results of ablation study. We calculate results marked with ‡ as they are not reported in the original paper. The Silver Data column indicates the total number of data entries used for pretraining. The best results within each dataset are denoted in bold.

manner, initializing the GNN using Transformer encoder parameters.

DualGen w/o SE displays significantly lower performance compared to DualGen w/o GE and the full DualGenmodel. With only graph encoders, DualGen w/o SE encounters challenges in AMR-to-text generation. This is because the graph encoder is designed not to retain all information, particularly entity details of the nodes. Instead, it prioritizes structural information and facilitates information exchange between two nodes connected by an edge.

DualGen w/o GE performs similarly to the findings of Ribeiro et al. (2021a) without pretraining on silver data, aligning with our expected outcomes. Leveraging the strength of pretrained Transformer-based language models, the variant DualGen w/o GE notably outperforms the variant DualGen w/o SE.

The full DualGen model significantly surpasses DualGen w/o SE and DualGen w/o GE without individual encoders, highlighting the importance of incorporating both sequence and graph encoders for enhanced performance.

B Large language models experiment settings

For LLaMA, we fine-tune the LLaMA-2-7B model using the code offered by Meta Research in `https://github.com/facebookresearch/llama-recipes`. We employ Fully Sharded Data Parallel (FSDP) and Parameter-Efficient Fine-Tuning (PEFT) to fine-tune the model, where we choose LoRA (Hu et al., 2021) as the PEFT method. We set the learning rate to 1×10^{-4}, and trained 10 epochs.

parameter	value
temperature	0.01
top p	1.0
n	1
frequency penalty	0.0
max tokens	2048

Table 9: The settings of GPT-3.5 and GPT-4.

For the experiment on GPTs, we use the OpenAI ChatCompletion API `https://platform.openai.com/docs/api-reference` provided by OpenAI, with the settings shown in table 9.

We use the following system prompt to instruct the model:

```
System:

Recover the text represented by
↪ the Abstract Meaning
↪ Representation graph (AMR
↪ graph) enclosed within triple
↪ quotes. Utilize only the
↪ information provided in the
↪ input. Output only the
↪ recovered text.
```

869

For few-shot prompting, we use the format illustrated in the following example:

User:

```
"""
(p / possible-01~e.1
  :ARG1 (m / make-05~e.2
    :ARG0 (c / company :wiki
    ↪  "Hallmark_Cards"
      :name (n / name :op1
      ↪  "Hallmark"~e.0))
    :ARG1 (f / fortune~e.4
      :source~e.6 (g / guy~e.8
        :mod (t / this~e.7)))))
"""
```

Assistant:

```
Hallmark could make a fortune off
↪  of this guy.
```

We evaluate GPT-3.5 using the entire AMR2.0 test set; for GPT-4, we assess its performance by randomly selecting and testing 400 entries from the AMR2.0 test set.

C Human evaluation settings

For human evaluation, we use the test set of AMR2.0. We filter out sentences shorter than 30 characters to eliminate meaningless entries like "2004-10-09". Following this, we randomly pick 100 entries and assign them IDs from 1 to 100.

Six volunteer annotators with an English education background carry out the annotation process. Three annotate entries 1 to 50, while the other three annotate entries 51 to 100.

Each entry i contains a reference text T_i from the AMR2.0 dataset and:

- the generated output P_i^1 of Song et al. (2020);
- the generated output P_i^2 of Ribeiro et al. (2021a);
- the generated output P_i^3 of Bevilacqua et al. (2021);
- the generated output P_i^4 of Bai et al. (2022);
- the generated output P_i^5 of DualGen without silver data pretraining;
- the generated output P_i^6 of DualGen with silver data pretraining.

For each assigned entry i, the annotator assigns scores $q_i^1, \cdots, q_i^6$ to rate the quality of sentence $P_i^1, \cdots, P_i^6$ and $s_i^1, \cdots, s_i^6$ to measure the similarity in meaning between T_i and $P_i^1, \cdots, P_i^6$. The scores $q_i^1, \cdots, q_i^6, s_i^1, \cdots, s_i^6$ are integers ranging from 0 to 10 (inclusive). The rating criteria are outlined in Table 10.

Score	Criteria for Quality Score	Criteria for Similarity Score
0	The sentence has numerous grammar errors or contains many irrelevant words or phrases, making it incomprehensible to readers.	The information conveyed in the generated output text is irrelevant to the information in the reference text.
2	The sentence has many errors in grammar, vocabulary, or word usage. Readers find it challenging to grasp the sentence's intended meaning.	The generated output primarily conveys information unrelated to the information in the reference text, only mentioning some of the concepts covered in the reference text.
4	The sentence has noticeable grammar, word, or phrase usage errors. Through careful reading, readers can generally understand the main points of the sentence.	The generated output conveys some information that aligns with the reference text, but there are apparent differences in their meanings.
6	The sentence has some grammatical errors or inappropriate word choices/phrases. The overall expression of ideas is somewhat coherent. Readers can generally understand the meaning.	The generated output primarily conveys the information covered in the reference text but either misses important details or includes some information not mentioned in the reference text.
8	The sentence contains a few grammar errors, uses words and phrases appropriately, expresses ideas coherently and naturally, and follows a logical structure that makes it easy for readers to understand the meaning.	The generated output conveys most of the information covered in the reference test but omits a few unimportant details or includes unimportant information not mentioned in the standard text.
10	The sentence is free of grammar errors, uses appropriate words and phrases, expresses ideas coherently and naturally, follows a logical structure, and can be easily understood by readers in terms of its meaning.	The generated output conveys the same information as the reference text, neither omitting details nor including information not mentioned in the reference.

Table 10: Rating criteria for human evaluation.