Promoting Structure-awareness of Large Language Model for Graph-to-text Generation

Anonymous ACL submission

Abstract

Recent advancement of Large Language Models (LLMs) has remarkably pushed the boundaries towards artificial general intelligence (AGI), with their exceptional generation and reasoning abilities. Despite this progress, a critical gap remains in employing LLMs to proficiently understand graph data. In this paper, we propose a new framework, named StructLLM to enhance the graph capabilities of large language models. Our framework first uses a structure-aware pre-training stage to pre-train a graph model to capture the structural information. Subsequently, we introduce four structure-aware instruction tasks to train a graph-to-text projector which bridges the domain gap between graph and text. Finally, we fine-tune our system on the AMR-to-text and Kg-to-text generation tasks. Experimental results that our model obtains significantly better results compared to fine-tuned LLMs, surpassing state-of-the-art systems. Further analysis shows that our model can better process complex graphs.

1 Introduction

Graph-to-text generation aims to generate faithful and fluent natural language description that conveys the same meaning as the input graphs (Konstas et al., 2017; Gardent et al., 2017). Sitting at the intersection between graphs and texts, this task can further facilitate the applicability of graphs in more downstream tasks, such as knowledge-grounded reason (Moon et al., 2019; Lv et al., 2020; Liu et al., 2021; Sun et al., 2023) and generation tasks (Tuan et al., 2019; Zhang et al., 2020a; Li et al., 2022; Gopalakrishnan et al., 2023).

Recently, Large language models (LLMs) have showcased remarkable performance on a wide array of text tasks, including language understanding (Touvron et al., 2023a), reasoning (Zhang et al., 2022), and text generation (Goyal et al., 2022; Chen et al., 2023). The primary idea is that LLMs acquire massive world knowledge when pre-trained on large-scale text data so that knowledge can be transferred to downstream tasks. Despite great success with text, LLMs suffer from salient limitations when processing graphs, thus are sub-optimal to graph-to-text generation. As shown in Figure 1, the given abstract meaning representation (left) and knowledge graphs (right) exhibit a different structure from the text sequence, where text units are organized and connected in a non-linear way. This fact was also revealed by recent efforts (Wang et al., 2023; Chai et al., 2023; Ettinger et al., 2023), showing that LLM's performances on graph-related tasks are subpar.

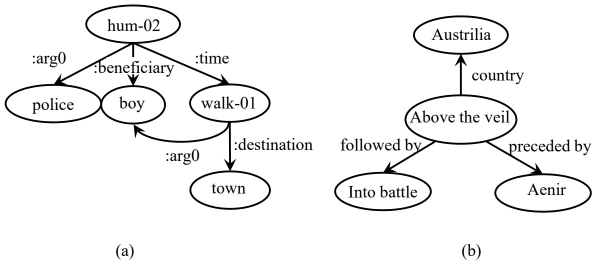


Figure 1: Illustration of two graphs: (a) an AMR meaning "*The police hummed to the boy as he walked to town.*"; (b) a knowledge graph meaning "*Above the Veil is an Australian novel and the sequel to Aenir. It was followed by Into the Battle.*".

To mitigate this issue, we aim to enhance the graph capabilities of large language models without compromising their original text knowledge. To this end, we propose StructLLM, a novel learning framework that allows LLMs to effectively understand and process graph structures. Our framework first pre-trains a graph encoder using structure-aware pre-training strategies to capture the structural information in the graph. Based on that, we perform structure-aware instruction tuning that bridges the modality gap between graph representation and text representation. Specifically, we design four self-supervised graph question-answering tasks to equip LLMs with the ability to leverage the encoded graph features for question answering. During structure-aware instruction tuning, we freeze the pre-trained graph model and language model and tune a graph-to-text projector. In this parameter-efficient way, we bridge the modality gap using a small amount of training data while maintaining the original distributional knowledge of LLMs. Finally, we fine-tune the resulting model on the task of graph-to-text generation to verify the effectiveness of our method.

We conduct experiments on two graph-to-text generation tasks: AMR-to-text generation and KG-to-text generation. Experimental results on standard benchmarks show that our model consistently achieves significant improvements over vanilla fine-tuned LLMs on both tasks and surpasses the state-of-the-art systems by a large margin. In addition, our method structure-enhanced LLM has better data efficiency than vanilla LLMs. Further analysis shows that our method is more effective for processing complex graphs. Our code will be released at `https:github.com/anonomy`.

2 Related Work

2.1 Large Language Models

Large language models (LLMs; Brown et al. 2020; Chowdhery et al. 2022; Touvron et al. 2023a) have substantially influenced the field of Natural Language Processing (NLP). As the pioneering work, Radford et al. (2019) and Brown et al. (2020) demonstrate the capability of language models to solve a task with minimal task supervision. The following work shows that LLMs are adept at leveraging textual instructions to perform various tasks including commonsense reasoning (Zhang et al., 2022), text summarization (Goyal et al., 2022; Chen et al., 2023), and massive multitask language understanding (Hendrycks et al., 2021). There have been recent attempts to adapt LLMs for processing graphs, by linearizing graphs into a sequence (Jiang et al., 2023), modifying model architecture to process graphs (Zhang et al., 2021; Xie et al., 2023), or continuously training LLMs using structure-aware training objectives (Sun et al., 2021). Different from the above work, we enhance the structure-awareness of LLMs without losing the structure information of the input graph while keeping the model architecture and knowledge distribution of LLMs unchanged.

2.2 Graph-to-text Generation

On a coarse-grained level, we categorize existing graph-to-text generation approaches into two main branches: The first branch focuses on graphs, aiming to better capture structural information in the input graph. Such as employing graph encoders (Beck et al., 2018; Damonte and Cohen, 2019; Zhu et al., 2019; Zhang et al., 2020b) or training neural networks with structure-aware learning objective (Song et al., 2020; Bai et al., 2020). For example, early studies on graph-to-text generation rely on statistical methods. Flanigan et al. (2016) convert input graphs to trees by splitting re-entrances, before translating these trees into target sentences with a tree-to-string transducer; Pourdamghani et al. (2016) apply a phrase-based MT system on linearized AMRs; Song et al. (2017) design a synchronous node replacement grammar to parse input graphs while generating target sentences.

The other branch investigates pre-trained language models to generate fluent text. For example, Mager et al. (2020) finetune a GPT model based on linearized input graphs. Ribeiro et al. (2021a) continually train language models using domain-adaptive training objectives. Bevilacqua et al. (2021) jointly train AMR parsing and AMR-to-text tasks using a pre-trained BART. Bai et al. (2022) train a BART model on graph data using graph-aware learning tasks. Wang et al. (2021) introduce a two-step structured generation approach based on pre-trained language models for KG-to-text generation.

Our method integrates the advantage of both graph structure encoding and pre-trained language models, using a graph-to-text projector and structure-aware learning tasks. The closest to our work, Ribeiro et al. (2021b) integrate AMR structures into pre-trained T5 (Raffel et al., 2020) using adapters (Houlsby et al., 2019) for AMR-to-text generation. However, they do not pre-train on graphs, and their method can not be used for decoder-only large language models.

3 Approach

Notation. Formally, denoting a graph as $\mathcal{G} = (\mathcal{V}, `\mathcal{E})$ where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ represents the nodes set and $\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}$ represents the edges set. An edge can further be denoted by a triple $\langle v_i, r_{ij}, v_j \rangle$, showing that node v_i and v_j are connected by relation r_{ij}. The goal of graph-to-text generation is to generate a word sequence $\mathcal{Y} = \{y_1, y_2, \ldots, y_M\}$ which conveys the same meaning as the input graph $\mathcal{G}$.

We propose StructLLM, a new graph-language joint learning framework that improves the graph awareness of pre-trained large language models. As shown in Figure 2(a), our model consists of three modules: a graph encoder, a graph-to-text projector, and a Language Model (LM). To capture the implicit structure of graphs, we first pre-train the graph encoder on a large-scale of unlabeled graphs using self-supervised learning strategies (see Section 3.1). Subsequently, we introduce three structure-aware instruction-tuning tasks to train the projector, aiming to bridge the gap between the pre-trained graph encoder and large language models (see Section 3.2). Finally, we perform a parameter-efficient fine-tuning of the trained model on the graph-to-text generation task (see Section 3.3).

3.1 Structure-aware Pre-training

Since there are no suitable pre-trained graph models for AMR and KG graph representation learning, we first employ a structure-aware pre-training step to pre-train a graph encoder. This step is designed to customize the model's learning behavior to meet the requirements of different downstream graph learning tasks. We employ the following two self-supervised learning tasks to pre-train a graph encoder on large-scale unlabeled graphs.

Graph De-noising. We train the model to learn contextualized representations using the graph denoising task. Given the input graph, we apply a noise function on its nodes/edges/subgraphs to construct a noisy graph, then we train the model to recover the original graph based on the noisy one. As shown in Figure 2(b), we implement the noise function by randomly masking, i.e. randomly replacing nodes, edges and sub-graphs with special `[MASK]` tokens with a probability of 15%.

Formally, given an input graph $\mathcal{G}$, and the noisy graph is denoted as $\hat{\mathcal{G}}$, the graph encoder is trained to minimize the following training objective:

$$\mathcal{L}_{denoising} = -\sum_{\mathcal{G} \in \mathcal{D}_{pretrain}} \log P(\mathcal{G}|\hat{\mathcal{G}}), \quad (1)$$

where $\mathcal{D}_{pretrain}$ denotes the pre-training dataset.

We follow RoBERTa (Liu et al., 2019) and use dynamic masking, where we generate the masking pattern every step instead of performing masking during data preprocessing.

Graph Contrastive Learning. This task trains the model to learn the overall representation of a graph using the contrastive learning mechanism (Hadsell et al., 2006; Frosst et al., 2019; Gao et al., 2021). The graph contrastive learning task aims to pull semantically close (or positive) graph pairs and push apart unpaired (or negative) examples. In particular, for a given graph $\mathcal{G}$, we define the positive example as the graph that is obtained by applying noise on the graph, and the negative examples are graphs in the same mini-batch during training.

Formally, we take the hidden state of the root node as the global representation of the graph, let $h^{\mathcal{G}_i}$ denote the representation of the ith graph in the dataset, the training objective is:

$$\mathcal{L}_{con} = -\log \frac{\exp(\mathrm{sim}(h^{\mathcal{G}_i}, h^{\hat{\mathcal{G}}_i})/\tau)}{\sum_{j \in \mathcal{N}(i)} \exp(\mathrm{sim}((h^{\mathcal{G}_i}, (h^{\mathcal{G}_j})/\tau)}, \quad (2)$$

where $\mathrm{sim}(\cdot, \cdot)$ denotes the similarity function[1], $\mathcal{N}(i)$ collects neighbor index of the ith example in the mini-batch, and $\tau > 0$ denotes the temperature hyper-parameter.

The graph encoder is trained by optimizing the total loss of the above two tasks:

$$\mathcal{L}_{pretrain} = \mathcal{L}_{denoising} + \alpha \mathcal{L}_{con}, \quad (3)$$

where α is a hyper-parameter that controls the importance of graph contrastive learning loss. Our pre-training framework is architectures-flexible and can accommodate various models, including both Graph Neural Networks (GNN) and Transformers (See section 4.4 for comparison.).

3.2 Structure-aware Instruction Tuning

The pre-trained graph models and LLMs are independently trained in an unimodal setting, making it challenging to align the graph and text representation. To this end, the structure-aware instruction tuning tasks are designed to bridge the modality gap between the pre-trained graph models and large language models.

As shown in Figure 2(b), we propose four structure-aware instruction tuning tasks to learn the interaction between the graph and text information. The first two tasks are node/edge-aware tasks which focus on local information. The last two are sub-graph level tasks, thus learning graph-level information. All tasks are unified as a graph

[1] We adopt the cosine similarity in experiments.

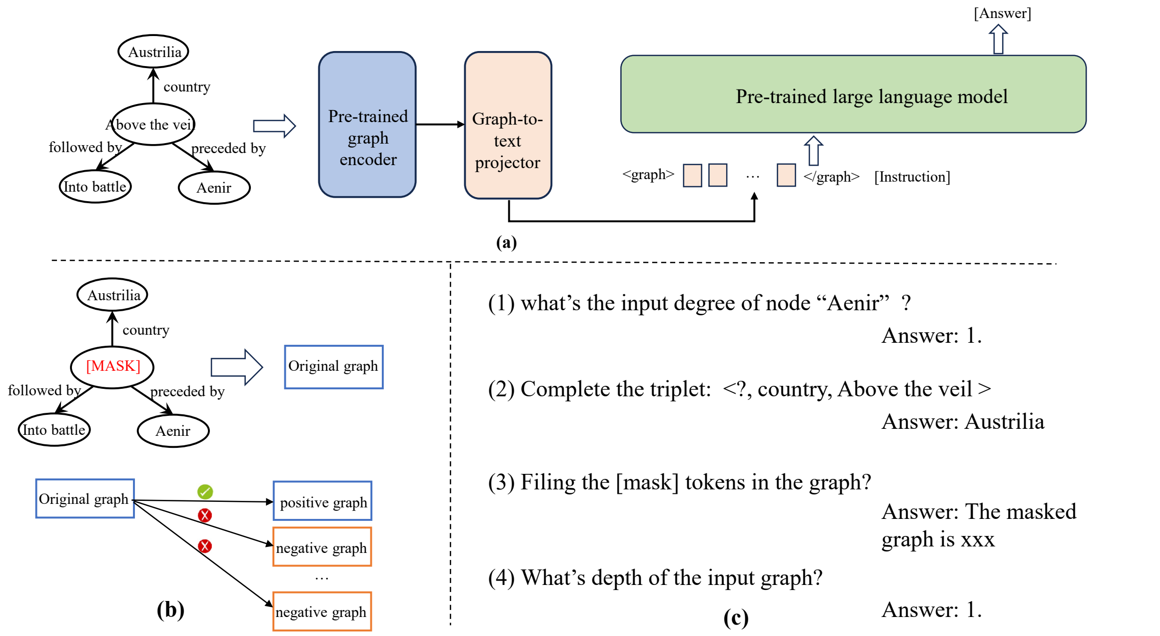


Figure 2: Illustration of the model architecture (a), structure-aware pre-training (b) and structure-aware instruction following (c).

question-answering format thus facilitating knowledge transfer from graph to text.

Node degree prediction. This task predicts the input and output degree of a specific node so that neural networks can capture the local structure of graphs. For example, the input and output degrees of node "Aenir" are 1 and 0, respectively.

Triplet completion. This task aims to complete the given triplet $\langle v_i, r_{ij}, ?\rangle$ according to the input graph, which guides models to learn the relationships between nodes. For example, node xx has a xxx relation with node xxx.

Sub-graph infilling. This task aims to fill the masked sub-graph according to its neighbor graph, thus helping models to learn sub-graph level structural information.

Graph depth prediction. This task predicts the depth of the input graph which refers to the length of the longest path from the root to that particular node. This task helps to achieve a deeper understanding of the graph structure. For example, the depth of the graph in Figure 2 is 1.

We follow the instruction-prompt scheme to design the prompt template, containing three parts: `System Message`, `Task Instruction`, `Answer`. In addition, we add two special tokens (<Graph>, </Graph>) to differentiate text representations from graph representations.

Formally, given a graph $\mathcal{G}$, a task instruction I and its corresponding answer A, we compute the probability of the target answers A by:

$$P(A|\mathcal{G}, I) = \prod_{i=0}^{|A|} P(X_{A_i}|\mathcal{G}, I, A_{<i}), \quad (4)$$

where $A_{<i} = \{a_1, a_2, ..., a_{i-1}\}$ represents generated answer. The training objective of the model is to minimize the negative log-likelihood of conditional word probabilities over all training examples:

$$\mathcal{L}_{IT} = -\sum_{<\mathcal{G},I,A>\in\mathcal{D}_{IT}} \log P(A|\mathcal{G}, I), \quad (5)$$

where $\mathcal{D}_{IT}$ denotes the instruction tuning dataset.

To reduce computation costs and avoid the issue of catastrophic forgetting, the pre-trained graph and language models remain frozen during structure-aware instruction tuning.

3.3 Task-specific Fine-tuning

After finishing structure-aware instruction tuning, we fine-tune the resulting model on the graph-to-text generation task. This step aims to adapt the model's generation behavior to meet the task of graph-to-text generation. Formally, assuming the input graph is denoted as $\mathcal{G}$, the corresponding text is denoted as $\mathcal{Y}$, and the task instruction is denoted as $\hat{I}$, the training objective is:

$$\mathcal{L}_{task} = -\sum_{<\mathcal{G},\hat{I},Y>\in\mathcal{D}_{task}} \log P(Y|\mathcal{G}, \hat{I}), \quad (6)$$

Datasets	AMR2.0	AMR3.0	WebNLG
Train	36521	55635	18102
Valid	1368	1722	872
Test	1371	1898	1862

Table 1: Benchmark graph-to-text generation datasets.

where $\mathcal{D}_{task}$ denotes the graph-to-text dataset and $\log P(Y|\mathcal{G}, \hat{I})$ is calculated in the same way as Equation 4.

In this stage, we freeze the backbone model and use the low-rank adaptation (LoRA; Hu et al. 2022) for parameter-efficient tuning.

4 Experiments

4.1 Datasets

Our method is evaluated on two graph-to-text benchmarks: AMR-to-text generation and KG-to-text generation.

Pre-training. For AMR graph pre-training, we collect about 1M silver AMR graphs parsed by AMRBART (Bai et al., 2022). These data are randomly selected from the Wikitext corpus. For KG graph pre-training, we collect about 250k KG graphs from DBpedia.

Instruction tuning. For AMR instruction tuning, we randomly sample 50k silver data from the pre-training corpus to construct the structure-aware instruction tuning dataset. For KG instruction tuning, we randomly sample 50k instances from the pre-training corpus.

Downstream Task. For AMR-to-text, we use the **AMR2.0** (LDC2017T10)[2] and **AMR3.0** (LDC2020T02)[3] corpora for task-aware fine-tuning and evaluation. For KG-to-text, we use the **WebNLG**[4] which is extracted from DBpedia. The test set contains two subsets, the seen part, and the unseen part. The unseen instances are from the five unseen domains. The UNSEEN part is designed to evaluate models' generalizability to out-of-domain instances.

Table 1 summarizes the statistics of downstream datasets used in our evaluation.

4.2 Settings

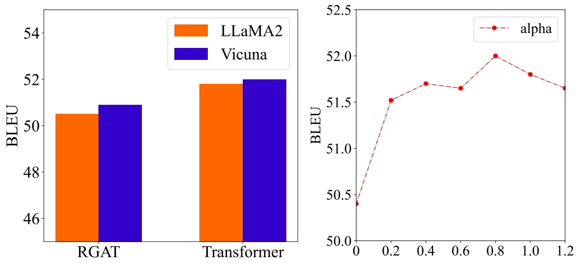
Model Configuration. We explore two types of architecture for graph encoding: Relational graph attention networks (RGAT; Busbridge et al. 2019) and Transformers. For RGAT, we use a hidden size of 512 and set the number of graph layers as 12. With regard to Transformers, we take the roberta-large (Liu et al., 2019) as the initial model. We take the last layer's hidden states as graph representations. For the frozen large language model, we explore the widely-used LLaMA-2-7b (Touvron et al., 2023b) model and Vicuna-7b-v1.5 (Chiang et al., 2023). For the graph-to-text projector, we adopt a two-layer perception with a GLEU (Hendrycks and Gimpel, 2016) activation function. In the task-specific fine-tuning stage, we set the LoRA rank as 64 and set the alpha as 16. We train for 1 epoch in the structure-aware pre-training stage, 5 epochs in the structure-aware instruction tuning stage, and 5 epochs in the structure-aware instruction tuning stage We use a batch size of 1024, 128 and 128 for graph encoder pre-training, structure-aware instruction tuning, and task-specific fine-tuning, respectively. The learning rates are set as 5e-5, 1e-3, and 1e-4 for the pre-training stage, instruction tuning stage and task-specific fine-tuning stage, respectively. We train models using 8 $\times$ A800 (80G) GPU, our largest model with Vicuna-7b-v1.5 requires less than 1 day for the first stage, less than 10 hours for the second stage, and less than 6 hours for the third stage.

Evaluation Metrics. Regarding AMR-to-text, we use three common Natural Language Generation measures, including BLEU (Papineni et al., 2002), CHRF++ (Popović, 2017) and METEOR (Banerjee and Lavie, 2005), tokenizing with the script provided with JAMR (Flanigan et al., 2014). For KG-to-text, we use the same metrics as AMR-to-text. We adopt the official *WebNLG Challenge*'s script to tokenize the text and evaluation.

4.3 Compared Systems

We compare our method with fine-tuned LLMs as well as other state-of-the-art methods. We consider two types of fine-tuned LLMs as baselines: 1) Vicuna-FT, a full-parameter fine-tuned Vicuna-7b-v1.5 on graph-to-text dataset; 2) Vicuna-LoRA, a parameter-efficient fine-tuned Vicuna-7b-v1.5 using LoRA. For **AMR-to-text generation**, the additional compared models are: 1) Zhu et al. (2019), a Transformer-based model that enhances self-attention with graph relations; 2) Zhang et al. (2020c), a graph-to-sequence model which uses dynamic graph convolutional networks for better graph modeling; 3) Bai et al. (2020), a graph en-

[2] https://catalog.ldc.upenn.edu/LDC2017T10
[3] https://catalog.ldc.upenn.edu/LDC2020T02
[4] https://synalp.gitlabpages.inria.fr/webnlg-challenge/challenge_2017/



(a) (b)

Figure 3: (a) Impact of model backbones; (b) Impact of the hyper-parameter α.

Model	BLEU	CHRF++	MET.
AMR2.0			
Zhu et al. (2019)	31.8	64.1	36.4
Zhang et al. (2020c)	33.6	63.2	37.5
Bai et al. (2020)	34.2	65.7	38.2
Mager et al. (2020)†	33.0	63.9	37.7
Ribeiro et al. (2021a)†	46.6	72.9	-
Bevilacqua et al. (2021)†	45.9	74.2	41.8
Bai et al. (2022)†	49.8	76.2	42.6
Cheng et al. (2022)†	51.5	77.6	45.2
Vicuna-7b-LoRA†	44.5	73.8	40.7
Vicuna-7b-FT†	49.6	76.2	42.3
Ours†	**52.7**	**78.4**	**46.7**
AMR3.0			
Zhang et al. (2020c)	34.3	63.7	38.2
Bevilacqua et al. (2021)†	46.5	73.9	41.7
Bai et al. (2022)†	49.2	76.1	44.3
Cheng et al. (2022)†	50.7	76.7	45.0
Vicuna-7b-LoRA†	44.2	73.0	40.1
Vicuna-7b-FT†	49.3	75.9	44.8
Ours†	**52.0**	**77.7**	**45.9**

Table 2: AMR-to-text results on AMR2.0 and AMR3.0. MET.=METEOR. Models marked with † are based on PLMs. The best result within each row block is shown in bold.

coder (Zhu et al., 2019) with a structural decoder that jointly predicts the target text and the input structure; 4) Mager et al. (2020), a fine-tuned GPT that predicts text based on a PENMAN linearized AMR graph; 5) Bevilacqua et al. (2021), a fine-tuned BART that predicts text based on a DFS linearized AMR graph; 6) Ribeiro et al. (2021a), a parameter-efficient model that uses a structural adapter to enhance a pre-trained T5 language model. 7) Bai et al. (2022), a model pre-trained on AMR data using graph pre-training strategies based on BART. 8) Cheng et al. (2022), a fine-tuned BART on AMR data using bidirectional bayesian learning.

For **KG-to-text generation**, the compared models are: 1) Moryossef et al. (2019), an end-to-end neural system based an LSTM decoder with attention; 2) Castro Ferreira et al. (2019), a Transformer-based model using sequences of KG triples as input; 3) Zhao et al. (2020), a dual encoding model that can narrate the gap between encoding and decoding; 4) Harkous et al. (2020), an end-to-end data-to-text generation system based on GPT-2 equipped with a semantic fidelity classifier; 5) Nan et al. (2021), a fine-tuned BART that trained on an open-domain structured data-to-text dataset; 6) Ribeiro et al. (2021a), a fine-tuned BART that represents the KG as a linear traversal; 7) Li and Liang (2021), a parameter-efficient tuning method that tunes soft prefixes based on GPT2-large.

4.4 Development Experiments

To assess the impact of various graph encoders and language models, we conducted a development experiment. Specifically, we compare the performance of different graph encoders and language models on the development dataset of AMR2.0 to evaluate their respective effects on the overall system's performance. As shown in Figure 3(a), the Transformer-based graph encoder obtains higher BLEU scores than the RGAT-based encoder in both settings. In addition, two LLMs achieve similar results, and Vicuna obtains slightly better results than LLaMA2. We thus chose the Transformer-based graph encoder and the Vicuna decoder as the backbone for the rest of our experiments.

We also study the impact of hyper-parameter α in graph pre-training. Figure 3(b) shows the performance of different values of α on the development dataset of AMR2.0. It can be observed that there are improvements when increasing the coefficient from 0, indicating that the graph contrastive learning task has a positive influence on graph-to-text generation. The BLEU score finally reaches the peak at α=0.8. We thus set α=0.8 for the rest of our experiments.

4.5 Results on AMR-to-text Generation

Table 2 lists the results of different systems on the testset of AMR2.0 and AMR3.0, respectively. Although having more parameters, Vicuna-7b-LoRA and Vicuna-7b-FT obtain lower results than Cheng et al. (2022) which is based on BART. This verifies our motivation that LLMs are weak in graph-aware tasks. Compared with Vicuna-7b-FT, our method achieves significantly ($p < 0.01$) better results on both datasets, improving the baseline model by 3.1 and 2.7 points in terms of BLEU on AMR2.0 and

AMR3.0, respectively. This indicates that our training framework can effectively improve the graph-awareness of large language models.

Compared with previous work, our method consistently outperforms the previous state-of-the-art system of Cheng et al. (2022) on both datasets in terms of all evaluation metrics, achieving 52.7 and 52.0 BLEU scores on the testset of AMR2.0 and AMR3.0, respectively. To our best knowledge, these are the best-reported results.

4.6 Results on KG-to-text Generation

Table 3 records the performance of different systems on the testset of WebNLG. We report results on all, seen and unseen testsets, respectively. Similar to AMR-to-text generation, Vicuna-7b-LoRA gives weaker results than previous systems, showing that LLMs are sub-optimal for processing graphs. Compared with Vicuna-7b-FT, our method gives consistently better results on all testsets regarding all metrics, with an average improvement of 1.3 BLEU on all testsets. In particular, the improvement on the unseen testset is larger than the seen testset, indicating that our method has a strong generalization capacity.

Compared with other state-of-the-art systems, the proposed method sees better performance, and our model obtains a BLEU of 61.1, 66.5, and 54.5 on all, seen and unseen, respectively. This result shows that the proposed method can effectively bridge the gap between graph and text, thereby performing better in translating graph to text.

5 Analysis

To have a deeper understanding of our model, We further analyze the behavior of the proposed model on AMR-to-text and KG-to-text datasets.

5.1 Ablation

We first study the effectiveness of individual components of our method. Specifically, we compared the full system with the following models: 1) model without structure-aware pre-training (w/o Struct_PT): we replace the pre-trained graph encoder with a randomly initialized encoder. 2) pre-training graph encoder with graph de-noising (w/o Graph_De)/graph contrastive learning (w/o Graph_CL) only; 3) model without structure-aware instruction tuning (w/o Struct_IT): we replace the trained graph-to-text projector with a randomly initialized one. 4) structure-aware instruction tuning without one of the node degree prediction (w/o NDP)/triplet completion (w/o TC)/sub-graph infilling (w/o SI)/graph depth prediction (w/o GDP) tasks.

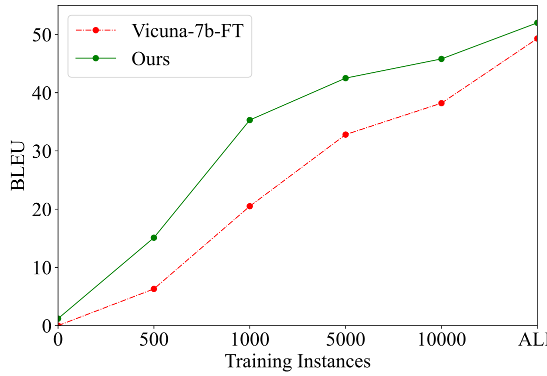


Figure 4: Performance comparison on AMR3.0 dataset with different training data.

Table 4 compares the performance of different systems on the testset of AMR3.0 and of WebNLG. First, it can be observed that structure-aware pre-training has a positive impact on graph-to-text generation, and removing this task leads to obvious performance reduction. Additionally, the graph de-noising task is more important than the graph contrastive learning task. Moreover, removing the structure-aware instruction tuning task also results in lower performance, showing that this task helps improve the structure-awareness of our model. Finally, all structure-aware instruction tuning tasks have overall positive impacts. Among the four tasks, triplet completion contributes most to model performance, and sub-graph infilling helps the least.

5.2 Data Efficiency in Task Fine-tuning

Our model is pre-trained on graph data and further tuned on structure-aware QA tasks, thus is expected to have high data efficiency when tuning on graph-to-text generation tasks. To verify this, we evaluate the performance of our model when fine-tuned on data of different sizes, and compare results with Vicuna-7b-FT. We randomly sample 100, 500, 1000, 5000, 10000 data from AMR3.0 for fine-tuning.[5] The results are shown in Figure 4, where we report the BLEU score for our model and Vicuna-7b-FT.

As shown in the Figure, our model gives significantly (p<0.001) better results than Vicuna-7b-FT in all training datasets, especially when there are

[5]We chose AMR3.0 since AMR-to-text generation is more challenge than KG-to-text generation.

	BLEU			CHRF++			METEOR		
MODEL	All	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen
Moryossef et al. (2019)	47.2	53.3	34.4	-	-	-	39.0	44.0	21.0
Castro Ferreira et al. (2019)	51.7	56.4	38.9	-	-	-	32.0	41.0	21.0
Zhao et al. (2020)	52.8	64.4	38.2	-	-	-	41.0	46.0	37.0
Harkous et al. (2020)†	52.9	-	-	-	-	-	42.4	-	-
Nan et al. (2021)†	45.9	52.9	37.9	-	-	-	40.0	42.0	37.0
Ribeiro et al. (2021a)†	54.7	63.5	44.0	72.3	77.6	66.5	42.2	45.5	38.6
Li and Liang (2021)†	56.3	63.4	47.7	-	-	-	42.1	45.0	39.3
Vicuna-7b-LoRA†	55.6	62.8	47.0	72.1	77.3	66.2	41.6	44.4	38.5
Vicuna-7b-FT†	59.8	65.9	52.6	74.8	78.4	70.4	43.7	46.1	41.2
Ours †	**61.1**	**66.5**	**54.5**	**75.8**	**79.5**	**71.8**	**44.6**	**46.9**	**42.5**

Table 3: KG-to-text results on WebNLG. Models marked with † are based on PLMs. The best result within each row block is shown in bold.

Model	**AMR3.0**	**WebNLG (All)**
Vicuna-7b-FT	49.3	59.8
Ours (full)	52.0	**61.1**
w/o Struct_PT	49.6	60.0
w/o Graph_De	51.3	60.3
w/o Graph_CL	51.6	60.9
w/o Struct_IT	50.4	60.3
w/o NDP	51.4	60.7
w/o TC	50.9	60.4
w/o SI	**52.1**	60.7
w/o GDP	51.3	60.6

Table 4: BLEU on the testset of AMR3.0 and WebNLG.

Graph Size	1-10 (522)	11-20 (556)	>20 (293)
Vicuna-7b-FT	51.5	48.2	46.9
Ours	53.1	52.4	49.7
Graph Depth	1-3 (422)	4-6 (667)	>6 (282)
Vicuna-7b-FT	52.7	47.9	45.7
Ours	54.3	50.4	49.0
Reentrancies	0 (622)	1-3 (712)	>3 (37)
Vicuna-7b-FT	52.6	48.5	43.1
Ours	54.8	51.0	45.3

Table 5: AMR-to-text generation performance on different graph groups.

fewer than 5000 training instances. This indicates that the proposed model has better generalization abilities compared to Vicuna-7b-FT, thanks to the structure-aware pre-training and structure-aware instruction tuning stages. Interestingly, our model achieves a BLEU-4 score of 2.1 without any training instances, showing that our method inherently holds graph-to-text translation ability.

5.3 Impact of Graph Complexity

It is expected that the benefit of our method will be more evident for structure-complex graphs as the proposed method is trained to be aware of the input graph structure. Table 5 shows the effects of the graph size, graph depth and reentrancies on the performance. We split the test set of AMR2.0 into different groups and compare the performance of our method and the baseline model. We first consider graph size, which records the number of nodes in an AMR graph. Our model consistently outperforms the baseline model on both tasks, with the performance gap growing on larger graphs. In terms of graph depth, our model consistently outperforms the baseline model on all graphs, and the improvements are bigger on deeper graphs.

We further consider reentrancies, which count the number of node which has multiple parents. The more reentrancies, the harder the graph is to be understood. Our method achieves larger improvements when the input graphs have reentrancies. This means that our system has an overall better ability to learn reentrancies.

5.4 Case Study

We further provide some cases to help better understand the effectiveness of the proposed model, please refer to the appendix A for more details.

6 Conclusion

This work presents a structure-aware training framework to build a graph large language model, aiming at improving the graph learning capabilities of LLMs. The proposed framework, StructLLM, injects graph-specific structural knowledge into the LLM through structure-aware pre-training and structure-aware instruction tuning paradigm. By leveraging a simple yet effective graph-text alignment projector, we enable LLMs to comprehend and interpret the input graphs as text. Extensive experiments on two benchmarks demonstrate the effectiveness of our method.

7 Limitations

One primary drawback of the proposed method revolves around the necessity of extensive graph data for pre-training our model. This requirement poses a significant limitation, particularly in low-resource settings where the availability of such data is uncertain or insufficient.

References

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.

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.

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.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. 2019. Relational graph attention networks. *CoRR*, abs/1904.05811.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proc. of EMNLP*.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*.

Yulong Chen, Yang Liu, Ruochen Xu, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Yue Zhang. 2023. UniSumm and SummZoo: Unified model and diverse benchmark for few-shot summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12833–12855, Toronto, Canada. Association for Computational Linguistics.

Ziming Cheng, Zuchao Li, and Hai Zhao. 2022. BiBL: AMR parsing and generation with bidirectional Bayesian learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5461–5475, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Allyson Ettinger, Jena Hwang, Valentina Pyatkin, Chandra Bhagavatula, and Yejin Choi. 2023. "you are an expert linguistic annotator": Limits of LLMs as analyzers of Abstract Meaning Representation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8250–8263, Singapore. 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.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Nicholas Frosst, Nicolas Papernot, and Geoffrey E. Hinton. 2019. Analyzing and improving representations with the soft nearest neighbor loss. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2012–2020. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. 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.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2023. Topical-chat: Towards knowledge-grounded open-domain conversations. *arXiv preprint arXiv:2308.11995*.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

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 *Proc. of COLING*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. 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.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yu Li, Baolin Peng, Yelong Shen, Yi Mao, Lars Liden, Zhou Yu, and Jianfeng Gao. 2022. Knowledge-grounded dialogue generation with a unified knowledge representation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 206–218, Seattle, United States. Association for Computational Linguistics.

Wenge Liu, Jianheng Tang, Xiaodan Liang, and Qingling Cai. 2021. Heterogeneous graph reasoning for knowledge-grounded medical dialogue system. *Neurocomputing*, 442:260–268.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8449–8456.

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.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 845–854.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proc. of NAACL*.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proc. of NAACL*.

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.

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

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, 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. *Journal of Machine Learning Research*, 21(140):1–67.

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

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

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. AMR-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13, Vancouver, Canada. Association for Computational Linguistics.

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.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Yi-Lin Tuan, Yun-Nung Chen, and Hung-yi Lee. 2019. DyKgChat: Benchmarking dialogue generation grounding on dynamic knowledge graphs. 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 1855–1865, Hong Kong, China. Association for Computational Linguistics.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? *arXiv preprint arXiv:2305.10037*.

Qingyun Wang, Semih Yavuz, Xi Victoria Lin, Heng Ji, and Nazneen Rajani. 2021. Stage-wise fine-tuning for graph-to-text generation. In *Proc. of ACL*.

Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl Yang, Yi Xu, et al. 2023. Graph-aware language model pre-training on a large graph corpus can help multiple graph applications. *arXiv preprint arXiv:2306.02592*.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020a. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2031–2043, Online. Association for Computational Linguistics.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2021. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*.

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

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

Zhuosheng Zhang, Shuohang Wang, Yichong Xu, Yuwei Fang, Wenhao Yu, Yang Liu, Hai Zhao, Chenguang Zhu, and Michael Zeng. 2022. Task compass: Scaling multi-task pre-training with task prefix. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5671–5685, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proc. of ACL.*

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.

KG#1: (<E> Weymouth Sands <R> preceded By (<E> A Glastonbury Romance))
Gold: A Glastonbury Romance preceded Weymouth Sands. **Baseline:** A Glastonbury Romance was preceded Weymouth Sands. **Ours:** Weymouth Sands was preceded by a Glastonbury Romance.
KG#2: (<E> Bacon sandwich <R> dish Variation (<E> BLT <R> ingredient (<E> Lettuce) <R> dish Variation (<E> Club sandwich)))
Gold: A variation on the club sandwich, BLT, has lettuce as an ingredient. A variation of the BLT is a bacon sandwich.
Baseline:BLT is a variation of the club sandwich and bacon sandwich. It includes lettuce .
Ours: lettuce is an ingredient in a blt which is a variation of a bacon sandwich and a club sandwich .

Table 6: Two KG-to-text generation cases. Given an AMR graph, we present the gold text and two generated outputs, given by baseline and our model, respectively.

A Appendix

A.1 Ablation Study

Table 6 lists two KG graphs and model outputs of our KG-to-text model and the baseline model. As shown in the first case, The output"*was preceded*" generated by the baseline model indicates that it exhibits deficiencies in its expression of temporal ordering, demonstrates inadequate control of verb tense, and ultimately fails to accurately convey the intended meaning. In the second case,the baseline model has not correctly captured the intended relation of the sentence due to the omission of relation, "*ingredient*". Ours model's output correctly identifies "*lettuce*" as an "*ingredient*" in the "*BLT*".Our model exhibits stronger performance in expressing tense and capturing relationships, as compared to the baseline model.