Mobius-Cycle: Multi-round Reverse Reconstruction of Graphs boosts the Consistency of Graph-to-Text Generation using LLMs

Anonymous ACL submission

Abstract

Graph-to-text generation (G2T) is a branch of Natural Language Generation (NLG) that aims to generate a textual output that can accurately and fluently describe the given graph. Recently, large language models (LLMs) have demonstrated remarkable in-context learning (ICL) capabilities in various NLG tasks. However, LLMs are prone to producing hallucinations, resulting in inconsistent content between the generated text and the original graph. In this paper, we are the first to comprehensively explore the performance of various LLMs in the G2T tasks on both KG and AMR data. For high-consistency G2T, we propose the LLM-based Mobius-Cycle (MoC) framework, a non-training inference-time scaling method that utilizes multi-round reverse reconstruction of graphs to enhance consistency. In each round, the LLMs perform three steps in sequence: reverse reconstruction, consistency judgement, and graph-to-text generation. We define the new metric, RRGScore, for LLM-as-a-Judge to measure the consistency between the reverse reconstructed graph (RRG) and the original graph. We also define a new metric, JCI (Judgement Confidence Index), to measure the confidence level of the LLMs during the consistency judgement process. We conduct extensive automatic evaluations and LLM-as-a-Judge evaluations on both KG and AMR datasets to assess the consistency of various LLMs. The experimental results indicate that our MoC framework can effectively improve the consistency of LLMs in G2T tasks and has good interpretability based on the consistency judgement. The code and data will be released upon acceptance.

1 Introduction

Graph-to-text generation (G2T) is an important branch of Natural Language Generation (NLG). It aims to generate natural language descriptions that fluently and precisely describe the given graph (Lin et al., 2024), such as Knowledge Graph (KG) or Abstract Meaning Representation (AMR). These structured graphs store rich information and are widely used in practical life and scientific research.

Early graph-to-text generation methods (Ribeiro et al., 2019; Zhao et al., 2020; Zhang et al., 2020b; Guo et al., 2019; Ribeiro et al., 2020) mainly used specialized graph encoders based on graph neural networks (GNNs) to capture structural information in the given graph. In the era of the "pre-train and fine-tune" paradigm, most graph-to-text generation systems (Song et al., 2020; Wang et al., 2021b; Li et al., 2021; Cheng et al., 2022; Bai et al., 2022) utilized supervised data from downstream tasks to fine-tune general pre-trained language models (PLMs) through multi-task learning to alleviate the loss of structured information when linearizing graphs. However, these methods require a large amount of supervised data for fine-tuning, and the trained models have insufficient generalization for unseen tasks.

Recently, large language models (LLMs) have demonstrated remarkable in-context learning (ICL) capabilities in various natural language processing tasks, making it possible for graph-to-text generation that does not require further training. However, LLMs are prone to producing hallucinations, resulting in inconsistent content between the generated text and the original graph. Furthermore, existing research (Yuan and Färber, 2023; Tan et al., 2024) is limited to superficial testing of the effectiveness of LLMs on KG data and still lacks exploration of AMR data.

To address these challenges, we proposed the Mobius-Cycle (MoC) framework, a non-training inference-time scaling method for high-consistency graph-to-text generation. Specifically, the LLM-based Mobius-Cycle framework employs multi-round reverse reconstruction of graphs to enhance the consistency of graph-to-text generation. As shown in Figure 1, in each round of the Mobius-Cycle, the LLMs perform three steps in sequence:

(1) reverse reconstruction; (2) consistency judgement; and (3) graph-to-text generation. The reverse reconstruction in the n-th round of Mobius-Cycle aims to re-extract the corresponding graph from the text generated by the LLMs in the previous round. The consistency judgement aims to determine the degree of consistency between the reconstructed graph and the original graph using LLMs. This step will provide feedback from LLMs, including overall alignment, modification suggestions, and our newly defined consistency score (RRGScore). Different from the conventional G2T task, we added the text generated in the previous round and feedback into the input of the third step, thus achieving the accumulation of information between multiple rounds.

Our contributions are summarized as follows:

- To our knowledge, we are the first to comprehensively explore the performance of various LLMs in the graph-to-text generation tasks on both KG and AMR data.
- We propose the LLM-based Mobius-Cycle (MoC) framework, a non-training inference-time scaling method for high-consistency graph-to-text generation.
- We define the new metric, RRGScore, for LLM-as-a-Judge to measure the consistency between the reverse reconstructed graph (RRG) and the original graph. We also define a new metric, JCI (Judgement Confidence Index), to measure the confidence level of the LLMs during the consistency judgement.
- We conduct extensive automatic evaluations and LLM-as-a-Judge evaluations on both KG and AMR datasets to assess the consistency of various LLMs. The experimental results indicate that our MoC framework can effectively improve the consistency of LLMs in graph-to-text generation tasks and has good interpretability.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 presents the details of our MoC framework. Section 4 shows experimental results and the case study, followed by a conclusion in Section 5.

2 Related Work

2.1 Graph-to-Text Generation

Graph-to-text generation (G2T) is a branch of data-to-text generation (D2T) that aims to generate a textual output that can accurately and fluently describe non-linguistic structured data (Lin et al., 2024). Since the commonly used graph data mainly includes Knowledge Graph (KG) and Abstract Meaning Representation (AMR), G2T can be further subdivided into KG-to-text generation (KG2T) and AMR-to-text generation (AMR2T). Early work (Ribeiro et al., 2019; Zhao et al., 2020; Zhang et al., 2020b; Guo et al., 2019; Ribeiro et al., 2020) mainly used graph encoders based on graph neural networks (GNNs) or variant recurrent neural networks (RNNs) to capture structural information in graph data. In the era of the "pre-train and fine-tune" paradigm, most graph-to-text generation systems utilize training data from downstream tasks to pre-train language models from scratch (Chen et al., 2020; Bai et al., 2022) or fine-tune general pre-trained language models (PLMs, such as GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020)) by introducing new training parameters (Ribeiro et al., 2021b; Wang et al., 2021a; Li et al., 2022) or optimization objectives (Guo et al., 2020; Cheng et al., 2022; Wang et al., 2023; Li et al., 2021) to capture structural information and thus enhance the model's understanding of graph data. After entering the large language models (LLMs) era, recent research (Guo et al., 2023; Tang et al., 2024; Ye et al., 2024; Wang et al., 2024; Tan et al., 2024) has begun to explore improving the performance on graph-related downstream tasks by constructing graph-oriented instruction data to perform supervised fine-tuning on general LLMs. Different from the above studies, we mainly focus on improving the consistency of LLMs in graph-to-text generation tasks in a non-training manner.

2.2 Reverse Engineering

Recently, reverse engineering has aroused great interest in the research community of large language models. Most researchers (Kazemi et al., 2023; Xue et al., 2023; Weng et al., 2023; Chen et al., 2024; Xi et al., 2024; Jiang et al., 2024; Deng et al., 2024; Yuan et al., 2024; Zheng et al., 2024) introduce reverse engineering into the thinking process of large language models to improve reasoning ability, such as logical reasoning, com-

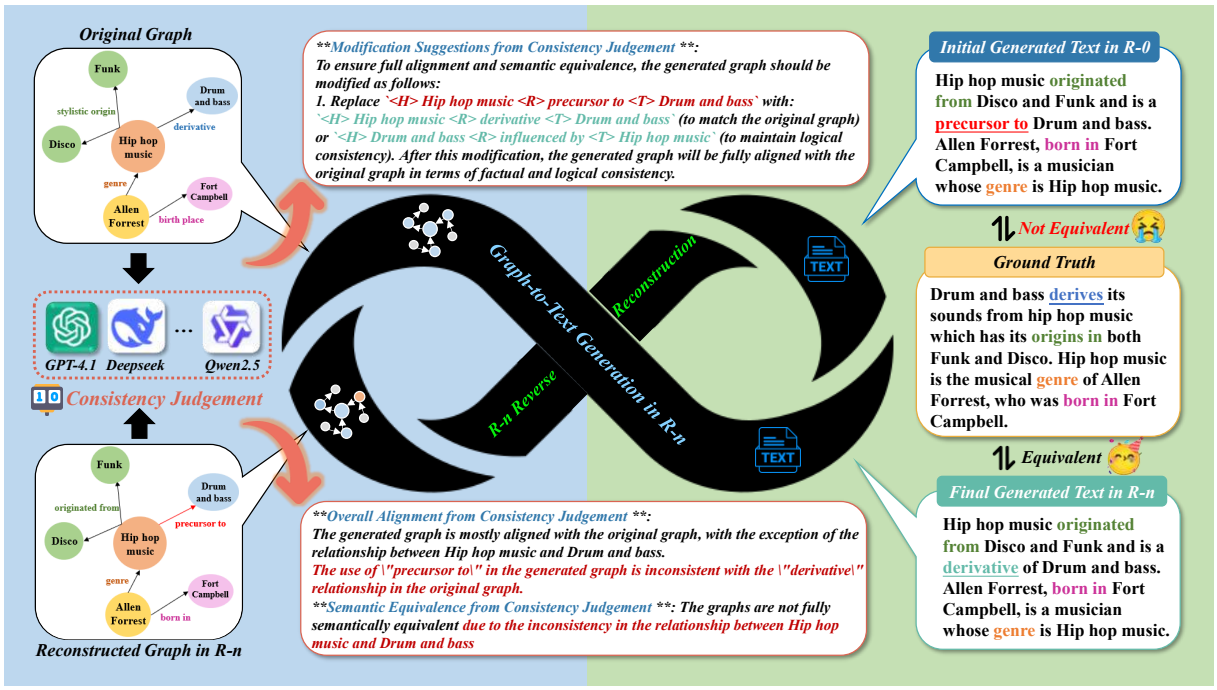

Figure 1: Overview of our Mobius-Cycle (MoC) framework. R-n denotes the n-th Round Mobius-Cycle. In R-0, only one graph-to-text generation is performed, and the output result is the initial generated text, represented as X^0. In R-n, the LLMs perform three steps in sequence: (1) Reverse Reconstruction; (2) Consistency Judgement; (3) Graph-to-Text Generation. The final generated text is represented as X^n.

monsense reasoning, and mathematical verification. Early graph-to-text generation methods (Song et al., 2020; Wang et al., 2021b; Li et al., 2021; Cheng et al., 2022; Bai et al., 2022) were also inspired by reverse engineering, which reconstructed input graphs through multi-task learning to alleviate the loss of structured information when linearizing graphs. Du et al. (2024) also proposed the bi-directional multi-granularity generation framework that jointly optimizes bidirectional objectives on Flan-T5 (Chung et al., 2024) for the KG-to-text generation. However, existing research has not yet explored the performance of large language models in AMR-to-text generation tasks. Therefore, we mainly explore a LLM-based graph-to-text generation framework that employs reverse reconstruction but does not require training, and comprehensively evaluate it on both KG and AMR.

3 Methodology

In this section, we present our proposed **Mobius-Cycle (MoC)** framework (as shown in Figure 1), a non-training LLMs-based graph-to-text generation approach that utilizes multi-round reverse reconstruction of graphs to enhance consistency. In each round, the LLMs perform three steps in sequence.

3.1 Step 1: Reverse Reconstruction

The reverse reconstruction in the n-th round of Mobius-Cycle aims to re-extract the corresponding graph $G^n = [g_1^n, ..., g_k^n]$ from the text $X^{n-1} = [x_1^{n-1}, ..., x_m^{n-1}]$ generated by the LLMs in the previous round, as shown in the Equ(2):

$$P(G^n|X^{n-1}) = \prod_{t=1}^{k} P(g_t^n|X^{n-1}, G_{<t}^n) \quad (1)$$

where X^0 denotes the initial generated text. Taking KG as an example, the prompt template used for its corresponding reverse reconstruction is shown in Figure 4.

3.2 Step 2: Consistency Judgement

As shown in Figure 1 and Figure 5, consistency judgement in the n-th round of Mobius-Cycle aims to determine the degree of consistency between the reconstructed graph $G^n = [g_1^n, ..., g_k^n]$ and the original graph $G^0 = [g_1^0, ..., g_k^0]$ using LLMs and provide feedback F^n including overall alignment, modification suggestions, and the consistency score (RRGScore, see Section 4.2). We have also added an **Early Exit Mechanism**: when the consistency score (RRGScore) of the reconstructed graph G^n reaches the set consistency threshold σ, the Mobius-Cycle can end prematurely.

3.3 Step 3: Graph-to-Text Generation

In this step, we first define the differences from conventional graph-to-text generation tasks. In addition to the original graph $G^0 = [g_1^0, ..., g_k^0]$, we also added the text $X^{n-1} = [x_1^{n-1}, ..., x_m^{n-1}]$ generated in the previous round and feedback F^n on the consistency judgment in the input, thus achieving the accumulation of information (I_{acc}^n) between multiple rounds. The target in this step is to generate the new text $X^n = [x_1^n, ..., x_j^n]$, which can be formulated as follows:

$$\begin{aligned} P(X^n|G^0, I_{acc}^n) &= P(X^n|G^0, X^{n-1}, F^n) \\ &= \prod_{t=1}^{j} P(x_t^n|G^0, X^{n-1}, F^n, X_{<t}^n) \end{aligned} \quad (2)$$

where n belongs to the set of positive integers. Due to limited page space, the more detailed prompts used in the Mobius-Cycle framework are shown in Appendix A.

4 Experiments and Analysis

In this section, we describe our experimental setup and report the evaluation results. Through comparative analysis with other baselines, we demonstrate the efficacy of our Mobius-Cycle framework on various LLMs and graph-to-text generation datasets.

4.1 Datasets

In order to conduct a more comprehensive study, we selected typical datasets for both KG and AMR for experimentation.

(1) KG-to-Text Generation

WebNLG (Gardent et al., 2017) is a frequently used benchmark dataset in the KG-to-text generation task. Each instance of WebNLG contains a KG from DBPedia and target text with one or more sentences describing the graph. Following previous work (Ribeiro et al., 2021a), we prepend <H>, <R>, and <T> tokens before the head entity, the relation, and the tail entity of a triple. We conducted experiments on a test set containing a total of 1,862 samples (971 *seen* and 891 *unseen*).

(2) AMR-to-Text Generation

The datasets we used in the AMR-to-text generation are **AMR 2.0** (LDC2017T10[1]) and **AMR 3.0** (LDC2020T02[2]) corpora releases. In an AMR graph, nodes represent concepts, and edges represent semantic relations. AMR 3.0 is a more comprehensive version of AMR 2.0 that includes more AMR-text data to cover wider situations. There are 55,635 AMR-text pairs in AMR 3.0 and 36,521 pairs in AMR 2.0. Following Mager et al. (2020), we linearize the AMR graphs using the PENMAN notation. The test sets for AMR 2.0 and AMR 3.0 contain 1,371 and 1,898 samples, respectively.

[1]https://catalog.ldc.upenn.edu/LDC2017T10

[2]https://catalog.ldc.upenn.edu/LDC2020T02

4.2 Evaluation Metrics

In this work, we not only used traditional automatic evaluation metrics but also introduced the LLM-as-a-Judge in our framework to measure the consistency between the original graph and the reverse reconstructed graph (both KG and AMR).

(1) Automatic Evaluation Metrics

We evaluate the generated description text from the following two aspects: (a) we first assessed the informativeness of the generated texts using **METEOR** (Lavie and Agarwal, 2007) and **ROUGE-L** (Lin, 2004), which measure lexical similarity by calculating the overlap of n-gram at the word level between the generated texts and the ground-truth descriptions; (b) we also computed the contextual embedding-based metric **BERTScore** (Zhang et al., 2020a) to evaluate the semantic similarity between the generated texts and the golden descriptions.

(2) LLM-as-a-Judge

We also evaluate the reverse reconstructed graph (RRG) using the powerful LLMs. Therefore, we propose a new LLM-based metric **RRGScore** to comprehensively measure the factual consistency score FCS and logical consistency score LCS between the reconstructed and original graphs, as shown in the Equ(3):

$$RRGScore = \lceil \alpha \cdot FCS + (1-\alpha) \cdot LCS \rceil \quad (3)$$

where α is a hyperparameter set to 0.6. Further details on the scoring rubrics for Consistency Judgement using LLMs can be found in Appendix B.

We also defined a new metric **JCI** (Judgement Confidence Index) to measure the confidence level of the LLMs during the consistency judgment process, as shown in the Equ(4):

$$JCI = \frac{AVG_{R-Top}}{AVG_{R-Final}} \quad (4)$$

where AVG_R denotes the average rounds.

4.3 Baselines

In these experiments, we mainly take into account the following strong baseline methods based on PLMs and LLMs, respectively:

Method	WebNLG			AMR 2.0			AMR 3.0		
	MTE	ROG	BERTScore	MTE	ROG	BERTScore	MTE	ROG	BERTScore
Baseline methods									
Fully Fine-tuned T5-large	44.18	68.24	-	43.85	-	-	-	-	-
Fully Fine-tuned BART-large	42.23	65.61	-	42.88	-	-	-	-	-
BiBL (BART-large)	-	-	-	43.20	-	-	43.40	-	-
text-davinci-003	26.95	45.64	-	-	-	-	-	-	-
gpt-3.5-turbo-0301	23.89	35.87	-	-	-	-	-	-	-
MuseGraph (♣ LLaMA-V1-7B)	43.86	69.73	-	-	-	-	-	-	-
Powerful LLMs-based methods under few-shot setting									
Qwen2.5-7B-Instruct	52.87	55.14	93.86	08.37	07.68	58.13	10.02	07.50	64.31
Qwen2.5-7B-Instruct w/ MoC-R@6	53.54	55.57	93.90	14.55	07.96	72.07	14.35	07.74	71.48
Qwen2.5-72B-Instruct	53.72	55.28	93.72	43.20	48.96	91.27	42.24	50.37	91.39
Qwen2.5-72B-Instruct w/ MoC-R@6	55.30	55.69	93.88	**45.72**	**51.05**	**91.70**	**43.54**	50.90	**91.43**
LLaMA-3.1-8B-Instruct	45.95	50.25	92.45	25.12	35.02	88.11	23.50	31.81	87.79
LLaMA-3.1-8B-Instruct w/ MoC-R@6	51.99	51.07	92.86	29.69	38.00	88.63	33.42	39.63	88.23
LLaMA-3.1-70B-Instruct	34.53	42.83	90.14	37.10	40.73	90.37	36.45	40.94	90.42
LLaMA-3.1-70B-Instruct w/ MoC-R@6	51.77	53.23	93.32	41.56	45.12	90.70	39.74	45.03	90.47
GPT-4.1 mini	54.78	55.57	93.79	26.81	43.53	88.75	37.62	44.80	87.67
GPT-4.1 mini w/ MoC-R@6	56.34	56.58	94.06	32.44	44.35	89.94	40.50	47.65	88.47
DeepSeek-V3	53.10	55.29	93.59	30.02	48.50	89.70	37.08	51.69	90.33
DeepSeek-V3 w/ MoC-R@6	**56.70**	**56.89**	**94.07**	32.53	50.47	90.23	38.03	**52.51**	90.81

Table 1: Performance comparisons of the automatic evaluation on the WebNLG and two AMR2Text datasets in the 3-shot setting. MTE: METEOR. ROG: ROUGE-L. **MoC-R@n** denotes the n rounds Mobius-Cycle. ♣ denotes the instruction-tuned model. **Bold** and underline represent the best result and the second-best result under the MoC-R@n setting.

(1) PLMs-based Methods

We have chosen two mainstream pre-trained language models (PLMs) as baselines for the graph-to-text generation tasks:

- **T5**: a text-to-text transfer transformer model (Raffel et al., 2020). We directly presented the results of fully fine-tuned T5-large from previous work (Ribeiro et al., 2021a).

- **BART**: a denoising autoencoder (Lewis et al., 2020) for pre-training sequence-to-sequence models with transformers. We directly presented the results of fully fine-tuned BART-large from previous work (Ribeiro et al., 2021a). In addition, we also add a strong baseline **BiBL** (Cheng et al., 2022) based on BART on the AMR datasets, which uses data-efficient bidirectional Bayesian learning.

(2) Large Language Models (LLMs)

Yuan and Färber (2023) explored the capability of ChatGPT (*text-davinci-003* and *gpt-3.5-turbo-0301*) to generate descriptive text from KG data in a zero-shot setting. MuseGraph (Tan et al., 2024) employed a graph-aware instruction tuning on the LLaMA-V1-7B (Touvron et al., 2023) model for the KG-to-text generation. However, there is still a lack of research exploring the effectiveness of LLMs on AMR data. In this paper, we also add a baseline method (ICL) that directly uses the following LLMs to accomplish the graph-to-text generation (both KG and AMR) in a few-shot setting.

The family of Qwen2.5 (Yang et al., 2024) models. We chose the following advanced models for extensive experimentation: Qwen2.5-7B-Instruct, Qwen2.5-72B-Instruct.

The family of LLaMA-3.1 models[3]**.** LLaMA-3.1 is competitive with leading foundation models across a range of tasks. We choose the following advanced models for experimentation: LLaMA-3.1-8B-Instruct and LLaMA-3.1-70B-Instruct.

Deepseek-V3 (Liu et al., 2024). It is a strong Mixture-of-Experts (MoE) language model with 671B total parameters, of which 37B are activated for each token. It is pre-trained on 14.8 trillion diverse and high-quality tokens, followed by SFT and RL stages to harness its capabilities fully.

GPT-4.1 mini (Achiam et al., 2023) is a significant leap in small model performance, even beating GPT-4o in many benchmarks. It matches or exceeds GPT-4o in intelligence evals while reducing latency by nearly half and reducing cost by 83%.

4.4 Implementation Details

Our experiments are conducted on a workstation running Ubuntu 20.04.6 LTS, with two Intel (R) Xeon (R) Platinum 8336C CPUs, four NVIDIA

[3]https://ai.meta.com/blog/meta-llama-3-1/

LLMs with MoC-R@6	Initial Text			Final Text				Top Text			
	FCS	LCS	RRGScore	FCS	LCS	RRGScore	AVG_R	FCS	LCS	RRGScore	AVG_R (JCI)
WebNLG dataset											
Qwen2.5-7B-Instruct	7.71	7.68	7.70	8.74	8.93	8.82	2.66	9.04	9.34	9.16	$2.15_{(0.81)}$
Qwen2.5-72B-Instruct	8.58	**9.01**	**8.75**	9.17	**9.55**	9.32	1.52	9.21	9.06	9.37	$1.39_{(0.91)}$
LLaMA-3.1-8B-Instruct	6.40	5.48	6.03	7.39	6.84	7.17	4.03	8.29	7.92	8.14	$2.78_{(0.69)}$
LLaMA-3.1-70B-Instruct	5.42	5.37	5.40	8.11	8.24	8.16	1.84	8.31	8.46	8.37	$1.74_{(0.95)}$
DeepSeek-V3	8.33	8.45	8.38	9.20	9.40	9.28	1.48	9.30	9.52	9.39	$1.43_{(0.97)}$
GPT-4.1 mini	**8.68**	8.83	8.74	**9.31**	9.48	**9.38**	**1.44**	**9.39**	**9.55**	**9.45**	$\mathbf{1.35}_{(0.94)}$
AMR 2.0 dataset											
Qwen2.5-7B-Instruct	3.70	3.50	3.62	7.30	7.40	7.34	**2.60**	9.00	**9.40**	**9.16**	$1.80_{(0.69)}$
Qwen2.5-72B-Instruct	8.00	7.80	7.91	**9.00**	**8.90**	**8.95**	**2.60**	**9.20**	8.90	9.08	$2.20_{(0.85)}$
LLaMA-3.1-8B-Instruct	7.30	6.80	7.10	8.70	8.50	8.62	4.10	9.10	9.10	9.10	$1.90_{(0.46)}$
LLaMA-3.1-70B-Instruct	**8.10**	**8.10**	**8.10**	8.30	8.50	8.38	3.30	8.60	8.60	8.60	$\mathbf{1.00}_{(0.30)}$
DeepSeek-V3	7.10	6.40	6.82	8.20	7.90	8.08	4.10	8.60	8.50	8.55	$2.00_{(0.49)}$
GPT-4.1 mini	6.90	6.30	6.65	8.30	8.40	8.34	3.40	8.60	8.70	8.63	$1.60_{(0.47)}$
AMR 3.0 dataset											
Qwen2.5-7B-Instruct	5.50	5.70	5.58	7.50	7.50	7.50	3.70	9.10	8.90	9.02	$\mathbf{2.10}_{(0.57)}$
Qwen2.5-72B-Instruct	7.40	7.30	7.36	8.20	8.10	8.16	**3.00**	8.40	8.50	8.44	$\mathbf{2.10}_{(0.70)}$
LLaMA-3.1-8B-Instruct	**8.70**	**8.70**	**8.70**	**9.20**	**9.40**	**9.28**	5.20	**9.20**	**9.40**	**9.28**	$2.40_{(0.46)}$
LLaMA-3.1-70B-Instruct	7.40	6.60	7.08	8.20	8.30	8.24	5.90	8.50	8.30	8.42	$2.90_{(0.49)}$
DeepSeek-V3	6.70	6.00	6.42	7.20	6.60	6.95	4.90	8.40	8.30	8.35	$3.00_{(0.61)}$
GPT-4.1 mini	7.50	6.90	7.26	8.00	8.20	8.08	4.20	8.60	8.70	8.64	$2.60_{(0.62)}$

Table 2: Performance comparisons of the LLM-as-a-Judge evaluation in the 3-shot setting. AVG_R denotes the average rounds. **Bold** indicates the best, and underline indicates the second best.

A800 GPUs, and 1.0TiB of memory. For models with parameters exceeding 70B, we choose to deploy the quantized version locally. For LLaMA-3.1-70B-Instruct, we directly use the Meta-Llama-3.1-70B-Instruct-AWQ-INT4 model. For Qwen2.5-72B-Instruct, we directly use the Qwen2.5-72B-Instruct-AWQ model. For DeepSeek-V3, we directly use the DeepSeek-V3 Chat model. We use the same hyperparameters for all LLMs: max_new_tokens=1024, top-k=1, and top-p=0.95. To minimize the impact of randomness, we use a temperature of 0.001 without any frequency penalty for the reverse reconstruction and the graph-to-text generation. As for the consistency judgement, we set the temperature to 0.9 to make the feedback provided by the LLMs more creative and diverse. We use the default settings for other parameters.

4.5 Main Results and Analysis

Table 1 and Table 2 present the automatic evaluation results between Mobius-Cycle (MoC) and other baselines and the LLM-as-a-Judge evaluation results on the WebNLG and two AMR2Text datasets, respectively.

As shown in Table 1, all LLMs under the MoC-R@6 setting achieved comparable performance to fully fine-tuned pre-trained language models (T5-large and BART-large) and MuseGraph on the WebNLG dataset. After introducing the proposed MoC framework, the performance of all six tested models improved, with the greatest improvement observed for LLaMA-3.1-70B-Instruct. DeepSeek-V3 achieved the best performance on the WebNLG dataset among all tested LLMs. However, on the two datasets of the AMR-to-text generation task, the initial performance of the LLMs was poor, especially for models with parameter sizes less than 10B. Obviously, the difficulty of the AMR 2.0 and AMR 3.0 datasets is significantly higher than that of the WebNLG dataset. After introducing our MoC framework, the performance improvement of the LLMs became even more significant on the AMR 2.0 and AMR 3.0 datasets, especially for Qwen2.5-7B-Instruct and LLaMA-3.1-8B-Instruct.

As shown in Table 2, more powerful models (GPT-4.1 mini, DeepSeek-V3, and Qwen2.5-72B-Instruct) tend to be more confident on the WebNLG dataset, with a higher JCI (over 0.90) and a higher RRGScore (over 9.00). In KG-to-text generation, GPT-4.1 mini and DeepSeeker-V3 (with MoC-R@6) achieved the top-two performance in both automatic evaluation and LLM-as-a-Judge evaluation. However, the performance of these two models in AMR-to-text generation is weaker than that of the Qwen2.5 and LLaMA-3.1 series models. Moreover, the Qwen2.5 series models are always more confident than the LLaMA-3.1 series models, with higher JCI and lower AVG_R. On the AMR 2.0 dataset, Qwen2.5-72B-Instruct with MoC-R@6 performs the best, with the highest RRGScore and the lowest AVG_R for final generated text, while

Method	METEOR	ROUGE	BERTScore
WebNLG dataset			
♡ LLaMA-3.1-8B-Instruct	49.39	46.71	90.56
- with 1-shot MoC-R@6	50.94	49.74	91.34
- wtih 2-shot MoC-R@6	51.69	50.45	91.78
- with 3-shot MoC-R@6	**51.99**	**51.07**	**92.86**
- with 4-shot MoC-R@6	51.74	50.97	91.96
AMR 2.0 dataset			
♡ LLaMA-3.1-8B-Instruct	26.25	35.40	88.23
- with 1-shot MoC-R@6	23.96	28.12	87.59
- with 2-shot MoC-R@6	24.44	31.31	88.06
- with 3-shot MoC-R@6	**29.69**	**38.00**	88.63
- with 4-shot MoC-R@6	24.67	37.48	**89.95**
AMR 3.0 dataset			
♡ LLaMA-3.1-8B-Instruct	29.64	32.54	87.05
- with 1-shot MoC-R@6	**36.53**	38.21	87.80
- with 2-shot MoC-R@6	30.32	39.15	87.57
- with 3-shot MoC-R@6	33.42	**39.63**	**88.23**
- with 4-shot MoC-R@6	32.03	36.23	87.87

Table 3: The ablation experimental results about the number of demonstrations on LLaMA-3.1-8B-Instruct model. ♡ denotes zero-shot MoC-R@6. **Bold** indicates the best, and underline indicates the second best.

Threshold	METEOR	ROUGE	BERTScore	AVG_R
LLaMA-3.1-8B-Instruct on the WebNLG dataset				
σ=7.00	51.51	**50.87**	**93.95**	**2.69**
σ=7.50	51.89	50.75	92.98	3.04
σ=8.00	**51.97**	51.11	92.86	4.03
σ=8.50	51.69	50.31	92.85	3.69
σ=9.00	51.59	49.81	92.10	4.06
Qwen2.5-7B-Instruct on the WebNLG dataset				
σ=7.00	53.53	**55.47**	**93.95**	**1.55**
σ=7.50	53.58	55.45	**93.95**	1.82
σ=8.00	53.54	55.14	93.90	2.66
σ=8.50	53.52	55.15	93.91	2.43
σ=9.00	**53.60**	55.11	93.90	2.63

Table 4: The ablation experimental results about the consistency threshold on LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct model in the 3-shot MoC-R@5 setting. AVG_R denotes the average rounds. **Bold** indicates the best, and underline indicates the second best.

also having the highest JCI. The average rounds required for the LLMs in the Mobius-Cycle to achieve the top RRGScore and exit early on the AMR 3.0 dataset is higher than that on the AMR 2.0 dataset, indicating that the AMR 3.0 dataset is more challenging. Surprisingly, the RRGScore of LLaMA-3.1-8B-Instruct (both in the final text and top text) on the AMR 3.0 dataset exceeded that of Qwen2.5-72B-Instruct. The experiment shows that our Mobius-Cycle method can effectively improve the consistency of LLMs (whether exceeding 70B or less than 10B) in graph-to-text generation tasks.

4.6 Ablation Study

We further explore the factors that affect the performance of the LLMs under the MoC-R@n setting through the following two aspects:

(1) Different number of demonstrations.

To investigate the impact of the number of demonstration examples on the LLMs under the MoC-R@6 setting, we tested the performance of the LLaMA-3.1-8B-Instruct from 0-shot to 4-shot on three datasets. As shown in Table 3, 3-shot is the optimal choice (either first or second) for comprehensive performance on these graph-to-text generation datasets under the MoC-R@6 setting. Continuing to increase the number of demonstration examples in the context leads to a decline in the performance of the large language models.

(2) Consistency threshold σ

To explore the impact of different consistency thresholds (7.00∼9.00) on the LLMs under the MoC-R@n setting, we tested the performance of two LLMs (LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instrut) in the 3-shot MoC-R@5 setting on the WebNLG dataset. We can see from Table 4 that for large language models with parameter sizes below 10B, the consistency threshold should be set to a value below 8 points (on a 10-point scale). Even on the WebNLG dataset with lower difficulty, setting the threshold too high will waste computational resources (requiring more rounds) and fail to improve the performance of large language models. For different downstream tasks, we need to explore based on the difficulty level of the task and the parameter size of the model before setting the optimal consistency threshold under the MoC-R@n setting.

4.7 Case Study and Error Analysis

To observe the trend of consistency changes in the Mobius-Cycle process more intuitively, we selected two representative LLMs (DeepSeek-V3 and LLaMA-3.1-8B-Instruct) that performed well in the above experiment for visualization purposes. For the KG-to-text generation, we observe the trend of consistency changes from round 0 to round 2 on the WebNLG dataset. For the AMR-to-text generation, we chose the more challenging AMR 3.0 dataset to observe the trend of consistency changes from round 0 to round 4. For each combination, we randomly select 10 cases and record the consistency at two levels: semantic consistency (represented by the mean of three automatic evaluation metrics) and graph consistency (i.e., RRGScore). As shown in the Figure 2, during the Mobius-Cycle process, both models showed a significant improvement in consistency, especially

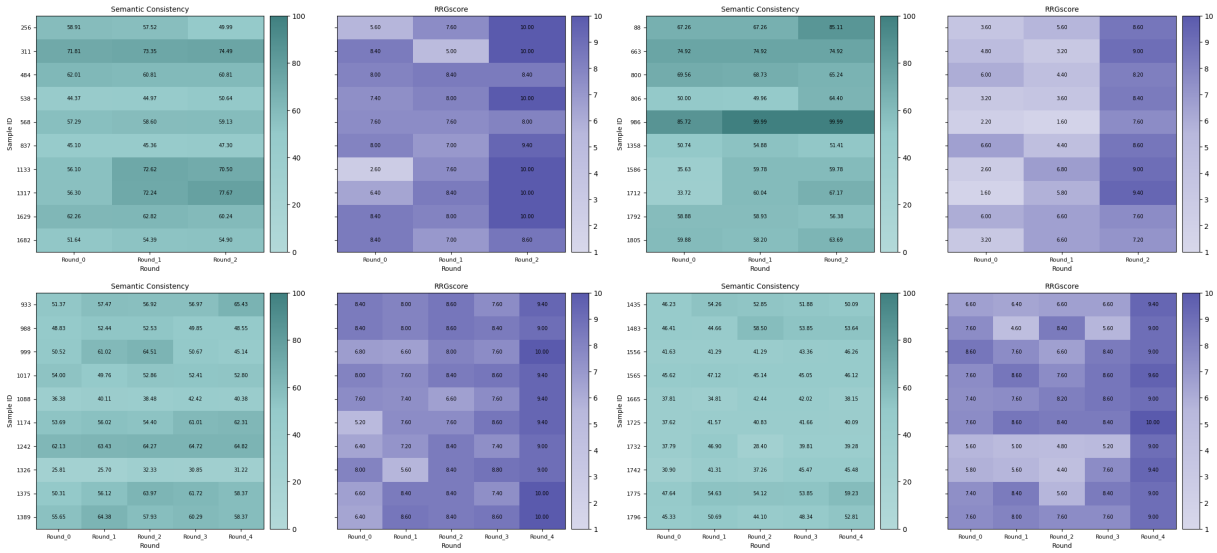

Figure 2: Performance visualization of DeepSeek-V3 (left) and LLaMA-3.1-8B-Instruct (right) on the WebNLG (top) and AMR 3.0 (bottom) datasets under the Mobius-Cycle settings. Each combination contains semantic consistency and RRGScore.

in terms of graph consistency. On the WebNLG dataset, 6 cases in DeepSeek-V3 achieved a perfect RRGScore, while LLaMA-3.1-8B-Instruct improved from only 3 cases, reaching a RRGScore of 6, to over 7 cases, reaching a RRGScore of 8. On the AMR 3.0 dataset, DeepSeek-V3 and LLaMA-3.1-8B-Instruct both reached the consistency threshold requirement of RRGScore (9.00) in round 4. The semantic consistency of most samples also improved during the Mobius-Cycle process.

However, there are still a few cases where graph consistency improves while semantic consistency decreases. After comparing the generated texts and graphs, we found that the low semantic consistency is mainly due to the semantic deviation between the reference text given in the dataset and the original graph. Taking the 256th sample in the WebNLG dataset as an example, its corresponding reference text is "*Ayam penyet is from Indonesia and is a popular dish in Malaysia*", but the original graph is "*<H>Ayam penyet<R>region<T>Malaysia<H>Ayam penyet<R>country<T>Indonesia*" which does not contain semantic information about "*popular dish*". For this case, even though the graph consistency of DeepSeek-V3 reached 10 points in round 2 of Mobius-Cycle, the semantic consistency was only 49.99, lower than the initial 58.91. It indicates that Mobius-Cycle can effectively improve graph consistency, but cannot compensate for semantic biases present in the original dataset.

5 Conclusion

In this paper, we are the first to comprehensively explore the performance of various LLMs in the graph-to-text generation tasks on both KG and AMR data. We propose the Mobius-Cycle (MoC) framework, a non-training inference-time scaling method for high-consistency graph-to-text generation. The LLM-based Mobius-Cycle framework utilizes multi-round reverse reconstruction of graphs to enhance consistency. In each round, the LLMs perform three steps in sequence: reverse reconstruction, consistency judgement, and graph-to-text generation. We define the new metric, RRGScore, for LLM-as-a-Judge to measure the consistency between the reverse reconstructed graph (RRG) and the original graph. We also define a new metric, JCI (Judgement Confidence Index), to measure the confidence level of the LLMs during the consistency judgement process. We conduct extensive automatic evaluations and LLM-as-a-Judge evaluations on both KG and AMR datasets to assess the consistency of various LLMs. The experimental results indicate that our MoC framework can effectively improve the consistency of LLMs in graph-to-text generation tasks and has good interpretability based on the consistency judgement.

Limitations

Our approach has these limitations: (1) Multi-round reverse reconstruction introduces computational overhead. The trade-off between cost and

performance requires adjusting the corresponding consistency threshold based on the actual scenario. (2) For the hyperparameter α of RRGScore, we found that 0.6 is better than 0.5. This enables the model to enhance logical consistency while ensuring factual consistency. However, a more granular analysis of this parameter can be further explored. (3) Although the LLM-as-a-Judge approach has been widely adopted, further human validation of the effectiveness of LLM-as-a-Judge can better demonstrate the effectiveness of the judge and alignment between the LLM judger and the human.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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.

Justin Chih-Yao Chen, Zifeng Wang, Hamid Palangi, Rujun Han, Sayna Ebrahimi, Long Le, Vincent Perot, Swaroop Mishra, Mohit Bansal, Chen-Yu Lee, et al. 2024. Reverse thinking makes llms stronger reasoners. *arXiv preprint arXiv:2411.19865*.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8635–8648. 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.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Linger Deng, Yuliang Liu, Bohan Li, Dongliang Luo, Liang Wu, Chengquan Zhang, Pengyuan Lyu, Ziyang Zhang, Gang Zhang, Errui Ding, et al. 2024. R-cot: Reverse chain-of-thought problem generation for geometric reasoning in large multimodal models. *arXiv preprint arXiv:2410.17885*.

Haowei Du, Chen Li, Dinghao Zhang, and Dongyan Zhao. 2024. Bi-directional multi-granularity generation framework for knowledge graph-to-text with large language model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 147–152.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *10th International Conference on Natural Language Generation*, pages 124–133. ACL Anthology.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv e-prints*, pages arXiv–2305.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv:2006.04702*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Trans. Assoc. Comput. Linguistics*, 7:297–312.

Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James Kwok. 2024. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6647–6661.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. Lambada: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6547–6568.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation, WMT@ACL 2007, Prague, Czech Republic, June 23, 2007*, pages 228–231. 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*, page 7871. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Few-shot knowledge graph-to-text generation with pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1558–1568.

Liang Li, Ruiying Geng, Bowen Li, Can Ma, Yinliang Yue, Binhua Li, and Yongbin Li. 2022. Graph-to-text generation with dynamic structure pruning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6115–6127.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yupian Lin, Tong Ruan, Jingping Liu, and Haofen Wang. 2024. A survey on neural data-to-text generation. *IEEE Transactions on Knowledge and Data Engineering*, 36:1431–1449.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing amr-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, November 3-7, 2019*, pages 3181–3192.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Trans. Assoc. Comput. Linguistics*, 8:589–604.

Leonardo FR 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.

Leonardo FR 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.

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.

Yanchao Tan, Hang Lv, Xinyi Huang, Jiawei Zhang, Shiping Wang, and Carl Yang. 2024. Musegraph: Graph-oriented instruction tuning of large language models for generic graph mining. *arXiv preprint arXiv:2403.04780*.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

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

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13492–13510.

Qingyun Wang, Semih Yavuz, Xi Victoria Lin, Heng Ji, and Nazneen Rajani. 2021a. Stage-wise fine-tuning for graph-to-text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 16–22.

Tianming Wang, Xiaojun Wan, and Shaowei Yao. 2021b. Better amr-to-text generation with graph structure reconstruction. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 3919–3925.

Zhuoer Wang, Marcus Collins, Nikhita Vedula, Simone Filice, Shervin Malmasi, and Oleg Rokhlenko. 2023. Faithful low-resource data-to-text generation through cycle training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2847–2867.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin

Guo, Junzhe Wang, et al. 2024. Training large language models for reasoning through reverse curriculum reinforcement learning. In *International Conference on Machine Learning*, pages 54030–54048. PMLR.

Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. 2023. Rcot: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought. *arXiv preprint arXiv:2305.11499*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1955–1973.

Jiahao Yuan, Dehui Du, Hao Zhang, Zixiang Di, and Usman Naseem. 2024. Reversal of thought: Enhancing large language models with preference-guided reverse reasoning warm-up. *arXiv preprint arXiv:2410.12323*.

Shuzhou Yuan and Michael Färber. 2023. Evaluating generative models for graph-to-text generation. *arXiv preprint arXiv:2307.14712*.

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

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 2020, Online, November 16-20, 2020*, pages 2162–2172.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020. Line graph enhanced amr-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 732–741.

Li Zheng, Hao Fei, Fei Li, Bobo Li, Lizi Liao, Donghong Ji, and Chong Teng. 2024. Reverse multi-choice dialogue commonsense inference with graph-of-thought. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19688–19696.

A The prompt formats of Mobius-Cycle

To provide readers with a more intuitive and clear understanding of how our proposed Mobius-Cycle framework is implemented, we offer the detailed prompt formats used for each step.

For the KG-to-Text Generation, the prompt formats we used are shown in Figure 3 to 6.

For the AMR-to-Text Generation, the prompt formats we used are shown in Figures 7 to 10.

Initial Graph2Text Generation in R-0

Instrution: This task involves natural language generation from structured graph data, specifically focusing on converting a set of semantic triplets into a text. Given a collection of triplets in the format <H><R><T>, where H represents the head entity, R denotes the semantic relationship, and T corresponds to the tail entity, your objective is to generate a coherent and linguistically natural text that maintains fidelity to the original graph structure. The generated text should exhibit three key characteristics: (1) grammatical fluency and readability, (2) logical consistency in presenting entity relationships, and (3) accurate preservation of the graph's structural and semantic properties. Only output the generated text.
Input: {Original Graph}
Output:{Initial Text}

Figure 3: The prompt of initial Graph-to-Text Generation in round-0 for KG data.

Reverse Reconstruction in R-n

Instrution: This task involves converting textual information into a structured graph format. Your objective is to identify and extract a collection of semantic triplets from the given text, where each triplet is composed of a head entity (H), a relationship (R), and a tail entity (T), represented in the format <H><R><T>. The extracted triplets should accurately represent the semantic relationships and entities present in the text, forming a coherent graph that effectively captures the underlying information structure. Only output the generated graph.
Input: {Generated Text}
Output:{Reverse Reconstructed Graph}

Figure 4: The prompt of reverse reconstruction in round-n for KG data.

B Scoring Rubrics for LLM-as-a-Judge

The scoring rubrics for **Consistency Judgement** (factual consistency score and logical consistency score) using LLMs are shown in the Table 5.

Consistency Judgement in R-n

Instrution: Your task is to evaluate whether the factuality and logical consistency between two sets of triplets (each representing a graph) are aligned. The triplets are formatted as <H><R><T>. Provide a detailed analysis of their alignment, including whether the graphs are semantically equivalent, and offer potential modification suggestions if inconsistencies are found. Below are the evaluation standards for factual consistency and logical consistency between the original graph and the reconstructed graph, using a 10-point scale. Higher scores indicate better consistency: {Factual Consistency Scoring Rubrics} and {Logical Consistency Scoring Rubrics}
Input: {Original_graph} + {Reconstructed_graph}
Output:
Factual Consistency Analysis:[] \nFactual Consistency Score:[]
Logical Consistency Analysis:[] \nLogical Consistency Score: []
Final Suggestion:[]

Figure 5: The prompt of consistency judgement in round-n for KG.

Graph2Text Generation in R-n

Instrution:
1. **Review and Analyze:**
  • Thoroughly examine the original graph and the provided feedback.
  • Identify any inconsistencies, deviations, or areas where the text fails to accurately represent the graph's structure.
2. **Refine the Text:**
  • **Consistency:** Ensure the text accurately reflects the relationships and entities in the graph, avoiding contradictions or misrepresentations.
  • **Fidelity:** Strictly adhere to the graph's structure, ensuring no key elements are omitted or misinterpreted.
3. **Integrate Feedback:**
  • Actively apply the specific suggestions from the feedback to improve the text's alignment and accuracy.
  • Iteratively refine the text to better capture the graph's relationships and structure, ensuring continuous improvement.
4. **Output:**
  • Deliver a polished textual description that is consistent, and faithful to the original graph, incorporating all feedback-driven enhancements.
  Only output the final text.
Input: {Original Graph} + {Generated Text} + {Judgement Feedback}
Output:{Final Text}

Figure 6: The prompt of Graph-to-Text Generation in round-n for KG.

Initial Graph2Text Generation in R-0 for AMR

Instrution: Your task is to transform an AMR graph into a coherent and natural-sounding text,Output only one generated text without providing any analysis. The input graph represents the semantics of a sentence in the form of nodes (concepts) and edges (relationships). Ensure that the output text is fluent, logically consistent, and accurately reflects the meaning, structure, and relationships encoded in the AMR graph. Pay attention to maintaining the semantic integrity of the graph while generating grammatically correct and contextually appropriate text.
Input: {Original Graph}
Output:{Initial Text}

Figure 7: The prompt of initial Graph-to-Text Generation in round-0 for AMR.

Reverse Reconstruction in R-n for AMR

Instrution: Your task is to extract an Abstract Meaning Representation (AMR) graph from the provided text. The AMR graph should consist of nodes representing concepts and edges denoting semantic relationships. Ensure that the extracted graph accurately captures the semantic structure and meaning of the text, maintaining fidelity to the original context and relationships.
Input: {Generated Text}
Output:{Reverse Reconstructed Graph}

Figure 8: The prompt of reverse reconstruction in round-n for AMR.

Consistency Judgement in R-n for AMR

Instrution: Your task is to evaluate whether the factuality and logical consistency between two AMR graphs are aligned.Provide a detailed analysis of their alignment, including whether the graphs are semantically equivalent, and offer potential modification suggestions if inconsistencies are found. Below are the evaluation standards for factual consistency and logical consistency between the original graph and the reconstructed graph, using a 10-point scale. Higher scores indicate better consistency: {Factual Consistency Scoring Rubrics} and {Logical Consistency Scoring Rubrics}
Input: {Original_graph} + {Reconstructed_graph}
Output:
Factual Consistency Analysis:[] \nFactual Consistency Score:[]
Logical Consistency Analysis:[] \nLogical Consistency Score: []
Final Suggestion:[]

Figure 9: The prompt of consistency judgement in round-n for AMR.

Graph2Text Generation in R-n for AMR

Instrution:
Task: Graph-to-Text Conversion with Feedback Integration
Objective: Generate coherent text reflecting the graph's structure, improving alignment and accuracy based on feedback.
Inputs:
1. Original AMR graph: the input graph represents the semantics of a sentence in the form of nodes (concepts) and edges (relationships).
2. Generated Text: Previously generated text from the graph.
3. Judgement Feedback: Evaluation of text alignment with graph, including suggestions for improvement.
Task Description:
1. Review the graph and feedback, addressing inconsistencies or deviations.
2. Improve the text for better alignment, ensuring:
  - Conciseness: Avoid unnecessary elaboration.
  - Consistency: Maintain accurate representation of graph relationships.
  - Fidelity: Stay true to the graph's structure.
3. Apply feedback suggestions to enhance alignment and accuracy.
Input: {Original Graph} + {Generated Text} + {Judgement Feedback}
Output:{Final Text}

Figure 10: The prompt of Graph-to-Text Generation in round-n for AMR.

Score	**Factual Consistency**: Whether the reconstructed graph accurately reflects the factual information in the original graph?
10	**Perfect Consistency**: The reconstructed graph retains all factual information from the original graph without any omissions or errors.
9	**Near-Perfect Consistency**: The reconstructed graph retains almost all factual information, with only minor and insignificant details missing or slightly incorrect.
8	**High Consistency**: The reconstructed graph retains most factual information, with a few minor details missing or slightly incorrect.
7	**Good Consistency**: The reconstructed graph retains the main factual information, but some important details are missing or incorrect.
6	**Moderate Consistency**: The reconstructed graph retains most of the main factual information, but several important details are missing or incorrect.
5	**Partial Consistency**: The reconstructed graph retains some main factual information, but many important details are missing or incorrect.
4	**Low Consistency**: The reconstructed graph retains only a small portion of the main factual information, with many important details missing or incorrect.
3	**Very Low Consistency**: The reconstructed graph retains very little factual information, with most important details missing or incorrect.
2	**Minimal Consistency**: The reconstructed graph retains almost no factual information, with only a few minor details correct.
1	**Almost No Consistency**: The reconstructed graph retains almost no factual information, with nearly all details missing or incorrect.
0	**No Consistency**: The reconstructed graph is completely inconsistent with the original graph, retaining no factual information.
Score	**Logical Consistency**: Whether the reconstructed graph maintains the logical relationships and structure of the original graph?
10	**Perfect Consistency**: The reconstructed graph fully maintains all logical relationships and structure from the original graph, with no logical errors or contradictions.
9	**Near-Perfect Consistency**: The reconstructed graph almost fully maintains the logical relationships and structure, with only minor and insignificant logical inconsistencies.
8	**High Consistency**: The reconstructed graph maintains most logical relationships and structure, with a few minor logical inconsistencies.
7	**Good Consistency**: The reconstructed graph maintains the main logical relationships and structure, but some important logical inconsistencies exist.
6	**Moderate Consistency**: The reconstructed graph maintains most of the main logical relationships and structure, but several important logical inconsistencies exist.
5	**Partial Consistency**: The reconstructed graph maintains some main logical relationships and structure, but many important logical inconsistencies exist.
4	**Low Consistency**: The reconstructed graph maintains only a small portion of the main logical relationships and structure, with many important logical inconsistencies.
3	**Very Low Consistency**: The reconstructed graph maintains very little of the main logical relationships and structure, with most important logical inconsistencies.
2	**Minimal Consistency**: The reconstructed graph maintains almost no logical relationships and structure, with only a few minor logical relationships correct.
1	**Almost No Consistency**: The reconstructed graph maintains almost no logical relationships and structure, with nearly all logical relationships incorrect or missing.
0	**No Consistency**: The reconstructed graph is completely inconsistent with the original graph, retaining no logical relationships or structure.

Table 5: Scoring rubrics for Consistency Judgement using LLM.