Evaluating and Improving Graph-to-Text Generation with Large Language Models

Jie He[1*] **Yijun Yang**[1*] **Wanqiu Long**[1] **Deyi Xiong**[3]
Víctor Gutiérrez Basulto[2†] and **Jeff Z. Pan**[1†]
[1] School of Informatics, University of Edinburgh, UK
[2] School of Computer Science and Informatics, Cardiff University, UK
[3] College of Intelligence and Computing, Tianjin University, Tianjin, China
`{j.he, j.z.pan}@ed.ac.uk, thomasyyj@outlook.com, gutierrezbasultov@cardiff.ac.uk`

Abstract

Large **l**anguage **m**odels (LLMs) have demonstrated immense potential across various tasks. However, research for exploring and improving the capabilities of LLMs in interpreting graph structures remains limited. To address this gap, we conduct a comprehensive evaluation of prompting current open-source LLMs on graph-to-text generation tasks. Although we explored the optimal prompting strategies and proposed a novel and effective diversity-difficulty-based few-shot sample selection method, we found that the improvements from tuning-free approaches were incremental, as LLMs struggle with planning on complex graphs, particularly those with a large number of triplets. To further improve LLMs in planning with graph sequences and grounding in truth, we introduce a new graph-to-text dataset, PlanGTG, annotated with two sub-tasks: reordering and attribution. Through extensive automatic and human evaluations, we demonstrate significant improvements in the quality of generated text from both few-shot learning and fine-tuning perspectives using the PlanGTG dataset. Our study paves the way for new research directions in graph-to-text generation[1]

1 Introduction

Recent advancements in large language models (Chowdhery et al., 2022; OpenAI, 2022, 2023; Touvron et al., 2023; Jiang et al., 2023) have revolutionized **n**atural **l**anguage **p**rocessing (NLP) due to their remarkable zero- and few-shot capabilities. While LLMs have been explored for structured graph tasks (Rong et al., 2020) and graph classification (Errica et al., 2020), their potential in verbalizing graphs in natural language (graph verbalization) remains underexplored. Graph-to-text generation (Koncel-Kedziorski et al., 2019; Ribeiro et al., 2021) is a challenging task that yields text from different graph structures and requires semantic alignment between graph and text.

Knowledge **G**raphs (KGs) (Pan et al., 2017b,a) store graph-like knowledge in triplets $\langle h, r, t\rangle$, stating that the head entity h is related to the tail entity t through the relation type r. Verbalizing triplets from KGs is essential for a wide range of tasks, such as knowledge graph completion (Hu et al., 2023b; Geng et al., 2023; Wiharja et al., 2020) as relation prediction, entity typing (Hu et al., 2022, 2023a) and negative triples (Arnaout et al., 2022a,b, 2021a,b) for answer validation, as well as creating QA datasets from graph data; e.g. CommonsenseQA (Talmor et al., 2019; Romero et al., 2019) and SciGraphQA (Li and Tajbakhsh, 2023). It also plays a key role in mitigating hallucinations of LLMs (Agrawal et al., 2023; Zhao et al., 2023b; Yang et al., 2024; Zheng et al., 2024).

To advance the KG-to-text generation task in the era of LLMs (Pan et al., 2023), we perform a preliminary evaluation on how well open-source LLMs perform on different prompts both in zero and few-shot scenarios. The prompt searching results emphasized the role of detailed instructions in unleashing LLMs' potential to generate fluent and accurate text from graphs. At the same time, we did not observe improvements of LLMs over various prompt optimizations in the zero-shot case (e.g. different linearizations and triplet expressions). For few-shot prompts, we empirically showed that choosing moderately hard and diverse prompts results in the best performance and propose a novel **d**ifficulty-**d**iversity balanced **d**emonstration selection method (DDD), which outperforms both simple difficulty or diversity-based method. However, the absolute value of improvements is marginal, reflecting the limitations of in-context learning. The analysis of graphs with varying complexity, measured by the number of triplets and graph diam-

* Equal Contribution.
† Corresponding author
[1]PlanGTG datasets can be found in https://github.com/probe2/kg_text..

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 10219–10244
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

eters (the longest shortest path between any two vertices), further suggests that current LLMs struggle with planning when handling graphs with a lot of triplets and with small diameters. This motivates us to use instruction-tuning over sub-tasks to strengthen LLMs' planning abilities, thus improving the performance of graph-to-text generation.

Inspired by Zhao et al. (2023a), we investigate whether instructing LLMs to explicitly output their decision process for graph-to-text generation can improve the quality of generated text. In addition, we design two subtasks: 1) **Reordering**: reorder the given KG triplets to better align with the generated text. 2) **Attribution**: attribute triplet indexes in the generated text using sequential numbers to enhance the interpretability of the generated text. To achieve this, we create **PlanGTG** (Sec.4), a new instruction dataset containing approximately 30,000 data pairs, featuring annotated attributions of triplets from the text. This dataset is generated using seeds rewritten from GraphNarritive (Shi et al., 2023) and GPT-3.5-turbo, incorporating the two subtasks sequentially.

We fine-tune LLMs with PlanGTG (Sec.5), conducting extensive automatic and human evaluations. Human evaluation results suggest that models fine-tuned with instructions are capable of successfully adjusting the order of KG triplets and correctly marking the sequence numbers in the generated text in most cases. Comparatively, models trained with PlanGTG outperform those using the EventNarrative (Colas et al., 2022), TEKGEN (Agarwal et al., 2021), and GraphNarrative (Shi et al., 2023) datasets in both zero-shot generalization and full-shot fine-tuning.

In summary, our main contributions are:

- We conduct comprehensive preliminary evaluations of graph-to-text tasks on LLMs, and explore the most effective prompting strategies in zero-shot and few-shot cases. We propose a novel and effective demonstration selection method DDD, and point out the remaining challenge of planning on complex graphs.
- We construct the PlanGTG dataset by adding two new subtasks, reordering and attribution, to study how LLMs can be improved through instruction fine-tuning with enriched auxiliary task information.
- Extensive experiments and evaluations have validated the effectiveness and utility of PlanGTG. Additionally, we explored how a curriculum learning approach, which strategically organizes the sequence of training data, can further enhance model performance.

2 Related Work

Various approaches have been proposed for transforming knowledge graphs into text. These include graph neural network based methods (Ribeiro et al., 2020) and language model based approaches (Liu et al., 2022b; Zhao et al., 2023a; Wu et al., 2024). Graph neural network based methods typically encode structured inputs explicitly as model representations (Puduppully et al., 2019; Guo et al., 2019; Koncel-Kedziorski et al., 2019). Among LLM based approaches, after LLM based sub-graph retrieval (Huang et al., 2024), a critical step is linearizing input triplets (Zhao et al., 2020), and recent efforts have introduced various planning techniques (Zhao et al., 2020, 2023a), including effective CoT based summaries (Wu et al., 2024).

There are studies that have focused on evaluating graph-to-text generation quality using pre-trained language models. Shi et al. (2023) address hallucinations in open-domain graph-to-text generation, while Yuan and Färber (2023) evaluate closed-source LLMs such as GPT-3 and ChatGPT only under the zero-shot setting. In contrast, our work focuses on open-source LLMs, aiming to provide a comprehensive evaluation across various aspects of the graph-to-text conversion process, including linearization, demonstration selection, and model scaling. Furthermore, we propose two new subtasks, reordering and attribution. These tasks are designed to enhance transparency in the generation process of LLMs and improve their performance.

Broadly speaking, there are also works on using knowledge graphs and ontologies for text generation (Mellish and Pan, 2008; Parvizi et al., 2014; Melnyk et al., 2022).

3 Preliminary Study

To investigate how to improve the performance of LLM's graph-to-text generation, we first perform a preliminary study to answer 1) How can different prompts influence the performance of graph-to-text generation, and will prompt engineering be relevant to attain such improvements? 2) Will scaling the parameters of open-sourced LLMs improve their graph-to-text verbalization abilities? We performed experiments following the settings in Section 3.1. For detailed experiment results, please refer to App.

C.

3.1 Experimental Setup

Datasets We conducted all of our experiments on the following three benchmarks: **WebNLG17**, **WebNLG20**, **DART**. Details about the datasets and experimental setup can be found in App. A.

Evaluation Metrics We used both automatic evaluations and human evaluations for our experiments. Four popular metrics BLEU (B-4) (Papineni et al., 2002), METEOR (ME) (Banerjee and Lavie, 2005), CHRF++ (CF) (Popović, 2015) and BartScore (BS) (Yuan et al., 2021) are adopted as evaluation metrics. Detailed explanations for all metrics are given in App. B.

Model All preliminary experiments except the scaling study are performed on Mistral-7b-instruct-v0.2 (Jiang et al., 2023).

3.2 Prompt Searching

We first explored how different instructions and the textual representation of triplets influence downstream performance in zero-shot cases. *This allows us to fix the optimal prompts as the prompt format for the rest of the paper.* In summary, more detailed prompts for explaining the formulation of the provided triplets and the task's goal result in more accurate and coherent generations. Regarding triplet formats, "*head* | *relation* | *tail*" performs slightly better than other representations such as "$\langle$*head*$\rangle$ $\langle$*relation*$\rangle$ $\langle$*tails*$\rangle$", but not significantly. Similarly, although several works have demonstrated the significance of different linearization techniques in graph-to-text tasks (Yang et al., 2020; Hoyle et al., 2021; Li et al., 2021), we found that LLMs are robust over different linearization and we, therefore, applied the default linearizations provided in the datasets in the rest of experiments. We conjecture that this robustness is due to LLMs being more invariant with the position and order, likely resulting from updated position embeddings such as RoPE (Su et al., 2024). Detailed experiments and discussions are available in App. C.1.

3.3 Example Selection

As the prompt templates are fixed during the prompt searching, we explored how few-shot demonstrations selection can influence and improve the performance of graph-to-text generation motivated by research emphasizing the importance of demonstrations selection (Luo et al., 2023; Drozdov et al., 2023). We conducted experiments on selecting demonstrations based on difficulty and diversity. Here difficulty is assessed by computing the cosine similarity between examples and input, with the assumption that following similar demonstrations facilitates easier graph-to-text generation. Diversity is measured by whether examples are within different k-means clusters. We will summarize our main findings due to the page limits. Please refer to App. C.2 for detailed experiment results.

Dataset	# of shots	B-4	ME	CF	BS
DART	1	21.33	33.69	55.10	-2.36
	3	21.30	33.78	55.17	**-2.33**
	5	22.22	33.90	55.47	-2.35
	10	**23.32**	**34.20**	**56.00**	-2.35
WebNLG17	1	23.48	35.49	57.66	-1.92
	3	25.07	**35.98**	58.60	**-1.91**
	5	24.83	35.92	58.72	-1.94
	10	**25.28**	35.87	**58.85**	-1.96
WebNLG20	1	26.07	**38.01**	58.64	**-1.97**
	3	**27.54**	36.09	**58.93**	-2.00
	5	27.42	35.95	58.58	-2.03
	10	27.36	36.01	58.87	-2.03

Table 1: Comparison between different numbers of few-shot selections on three popular graph-to-text datasets. B-4, ME, CF, BS refer to BLEU-4, METEOR, CHRF++ and BartScore respectively

Finding 1: **Selecting moderately difficult and diverse demonstrations yields the best results, but with fluctuations.** We separate the level of difficulties into 5 levels by uniformly selecting examples ranging from the most similar to the least similar examples between the input and training sets considering both graph similarity and generated text similarity in the one-shot scenario. For the performance shown in Fig. 6 and Fig. 7, we found that in most cases, the performance peaks at the middle level of both difficulty and diversity. This suggests that LLMs generalize better due to not relying on memorized shortcuts from overly similar demonstrations or samples closely resembling the input. Although these findings give insights on the demonstrations selection and motivate us to propose the **d**ifficulty-**d**iversity balanced **d**emos selection (DDD), there are cases where results fluctuate unpredictably based on different demonstrations. We attribute this variability to potential limitations in the engineering of demonstrations.

Finding 2: **DDD selections perform better than simple diversity and difficulty selection, but marginally.** Based on our previous findings, we

Number		(#samples)	B-4	ME	CF	BS
triplets	1	(848)	23.36	38.93	61.22	-1.92
	2	(797)	17.55	32.98	53.59	-2.36
	3	(821)	17.09	32.57	53.24	-2.22
	4	(869)	17.21	31.68	52.89	-2.29
	5+	(1762)	11.79	29.59	47.08	-2.63
Diameters	0	(575)	23.29	38.92	61.19	-1.93
	1	(3463)	14.18	30.30	48.65	-2.53
	2	(850)	18.60	33.20	55.09	-2.00
	3	(201)	17.75	32.37	54.41	-2.02
	4	(8)	20.57	33.96	59.36	-2.24

Table 2: Comparison of performance between different triplet numbers and graph diameters on DART dataset.

propose a DDD selection approach. This method employs a dual-phase process where initially, leveraging findings from our diversity investigation, we recall all samples within the same cluster. These samples are then sorted by their difficulties (i.e. the cosine similarity from the input graph). For instance, in a 3-shot scenario, we select samples ranked at 25%, 50%, and 75% in terms of similarity. The comparative analysis presented in Table 14 shows that DDD is more effective across various datasets than strategies solely focused on difficulty or diversity. This highlights that concurrently considering difficulty and diversity in the selection of demonstration samples could be the best sample selection strategy. However, despite the good comparative results, the absolute value of improvement is incremental both from the best result of one-shot to three-shot and from DDD selection to difficulty/diversity selection. This suggests that few-shot example selections yield minimal improvements.

Finding 3: **Increasing the number of shots does not help.** To mitigate the high randomness of sampling demonstrations, we investigate the influence of increasing the number of demonstrations based on the DDD selection methods. We picked 1, 3, 5, and 10 samples and reported the results in Table 1. We found that increasing the number of samples does not always help. This can possibly explained by the fact that the samples selected from DDD methods are already diverse enough for LLMs to learn the task format and those difficult demonstrations did not provide additional relevant knowledge to help LLMs translate the input graph.

3.4 Improvment Space

Finally, we analyze the behavior of models over different complexity of graphs to explore the space for improvements. We represent the graph complexity through the number of triplets and the graph diameters (the longest-shortest path between any two vertices) and analyze the results on Mistral-7b-instruct-v0.2 in Table 2[2]. Firstly, as the number of triplets increases, a consistent decline in performance is shown across all metrics. This shows the difficulties that LLMs face in text generation from complex graphs. This motivates us to consider improving the planning abilities of LLM to handle graphs with more triplets. For graph diameters, LLMs perform the worst when interpreting graphs with only one diameter. This is because when the number of triplets is small, LLMs may suffer from hallucinations in order to make the generation coherent and satisfying-looking. This further motivated us to design a dataset that allows LLMs to cite their generation in order to mitigate the hallucination.

4 The PlanGTG Dataset

In the perspective of fine-tuning for improving the performance of graph-to-text generation, we construct PlanGTG (**Plan**ning for **G**raph-to-**T**ext **G**eneration), a graph-to-text paired instruction-tuning dataset, annotated for both reordering and attribution subtasks. The construction of PlanGTG is guided by several key objectives: (1) ensuring the diversity of both the structure of graphs (size and diameter) and the topic of texts; (2) avoiding text-graph misalignment (i.e. textual descriptions containing information not found in input graphs), a significant cause of hallucination (Shi et al., 2023); and (3) ensuring interpretability by annotating the attribution triplets from the text descriptions and automatically formatting the linearization labels. These goals are achieved through sequential generation and parallel annotation.

4.1 Dataset Construction

The flow chart for the process of generating PlanGTG is shown in (a) of Fig. 1, which consists of three parts: seed data preparation, sequential graph-text pair generation and parallel attribution annotations. We apply GPT-3-turbo-1106 as the base model for data generation. For seed preparation, sequential pairs generation and the parallel annotation, the used prompts are shown in Fig. 9, Fig. 10 and Fig. 8 in Appendix respectively.

Seed Data Creation We begin with the graph-text pairs in GraphNarritive (Shi et al., 2023). As a

[2] For results on all datasets, see Table 15 and Table 16 in App. C.4.

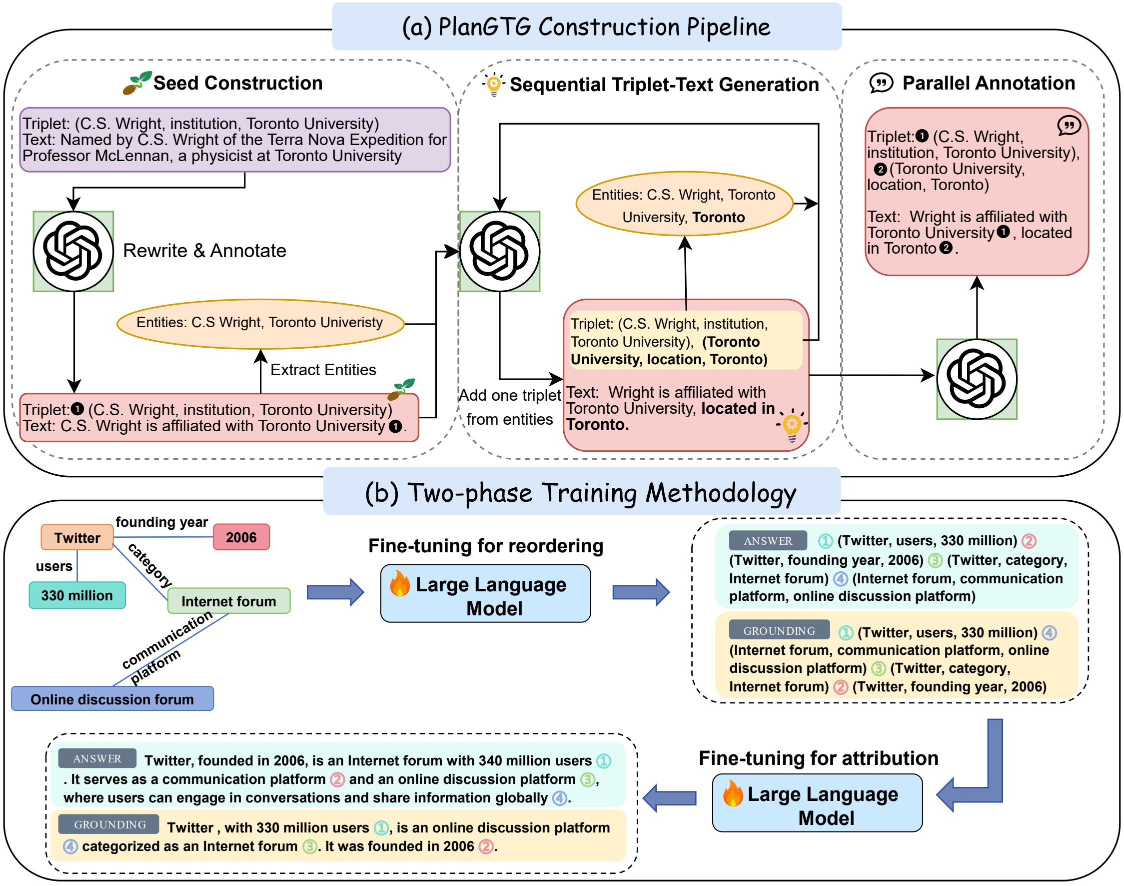

Figure 1: (a) The flow chart of the construction for the PlanGTG dataset, the squared text refers to the output of GPTs and the circled text represents the result extracted automatically by rules. The newly added information by GPTs is marked in bold. (b) Our training pipeline: The training methodology consists of two phases: planning-guided generation and attribution generation. It enables LLMs to first generate triplets that follow a more natural language order and subsequently guide the generation of attributed answers.

start, we choose data-text pairs with only one triplet. To ensure diversity, we uniformly selected n random samples from each type of relation present in GraphNarritive. Next, we prompt GPT-3.5-turbo-1106 to 'regenerate' the text based on the source text and graphs, incorporating the annotation derived from the graph. For instance, given the triplet "*C.S. Wright | institution | Toronto University*" alongside the text "*named by C.S. Wright of the Terra Nova Expedition for Professor McLennan, a physicist at Toronto University*", we use GPT to regenerate the text as "*C.S. Wright is affiliated with Toronto University. (1)*". This allows us to make both the initial attribution annotation and discard the redundant information from the initial description text. We create about 3 thousand one-triplet seeds to serve as the foundation data.

PlanGTG Generation We then generate the PlanGTG dataset from the foundation data in a sequential way. Specifically, for each triplet-text pair, we ask ChatGPT to (1) generate the graph by adding one new triplet that integrates well with the existing triplets, (2) then update the corresponding text description incorporating information of the added triplet and (3) make the attribution annotation. We conduct one inference for Steps 1 and 2 and perform a separate inference for Step 3 and different demonstrations are provided for Step 3. This is because we empirically found that integrating the attribution annotation numbers may harm the performance of generating the text description of the new triplets. We also provide the existing entities from the triplet list and guide GPT to choose one of the entities in Step 1 to ensure the connectivity of the generated graph. The above process shows the steps to craft one sample with $n+1$ triplets from n triplets. For each foundation seed, we do these steps iteratively to attain 2 to 10 triplets. Eventually, we created PlanGTG with 28,837 training

Dataset	Hallucinated Entities↓	Missed Entities↓	Hallucinated Relations↓	Missed Relations↓	GR↑
TEKGEN	0.84	0.08	0.92	0.07	4.48
EVENT	0.69	0.05	0.70	0.08	4.69
GN	0.62	0.02	0.74	0.03	4.73
PlanGTG	0.36	0.01	0.42	0.03	4.78

Table 3: Human evaluation on pretraining datasets. Cohen's kappa coefficients for labeling three factors are as follows: 0.82, 0.79, and 0.77. GN represents the GraphNarrative dataset. GR means the Grammar.

points and 996 development points after filtering data points with wrong patterns.

Dataset Description In general, PlanGTG consists of 28,837 training pairs and 996 development graph-text pairs, with an average of 5.48 triplets within graphs and 37.6 words in the text. In the Appendix D.2, Fig 3 shows the distribution of the number of triplets in the dataset. Fig 4 shows the distribution of the diameters in the graph. Fig 5 shows the distribution of the words contained in the text. To ensure that PlanGTG does not overlap with the test sets, we checked all triplets in PlanGTG. The percentages of overlapping triplets across datasets are 0% for DART, 0% for WebNLG 2017, and 0.00019% for WebNLG 2020 test sets. Additionally, none of the input graphs overlap with these test sets.

4.2 Dataset Quality

Two professional human annotators assess the quality of the generated graph-text pairs in PlanGTG. We randomly select 200 examples and the annotators evaluate the hallucination, missing information, grammatical correctness and fluency of the generated text (using a 5-point Likert (Likert, 1932)). Detailed explanations for all metrics are given in App. B. The scores from both annotators are averaged. At the same time, we also evaluate the quality of automatically extracted pre-training texts (EVENT, TEKGEN, and GraphNarrative) in the same way. Table 3 reveals that, on average, there are 0.36 hallucinated entities and 0.42 hallucinated relations per graph-text pair in PlanGTG. This reflects that our instruction dataset, although generated by ChatGPT, maintains high quality due to carefully designed generation instructions and meticulous post-processing. This quality is notably superior to automatically extracted pre-training texts, with GraphNarrative exhibiting the highest indicators of 0.62 for hallucinated entities and 0.74 for hallucinated relations.

In contrast, our evaluation shows low rates of missing entities and relations at 0.01 and 0.03, respectively, indicating that ChatGPT consistently incorporates graph information into text without significant information loss. Regarding language naturalness, the score 4.84 demonstrates that the generated text is highly fluent with minimal grammatical errors. These results highlight the high quality of our generated text compared to automatically extracted pre-training texts. For instance, GraphNarrative's best scores for missing entities and relations are 0.02 and 0.03, respectively, and its language naturalness score is 4.73, indicating slightly lower performance in these aspects.

5 Experiments

5.1 Experimental setting

To evaluate whether PlanGTG can enhance LLMs generalizability in graph-to-text tasks, we conducted experiments on both zero- and full-shot learning. In our zero-shot experiments, as shown in (b) of Figure 1, we fine-tuned LLMs on the PlanGTG dataset and subsequently evaluated their performance on WebNLG 2017, WebNLG 2020, and DART datasets. For the baselines, we fine-tuned LLMs on other graph-to-text generation datasets. We compared three classic datasets: EVENT (EventNarrative) (Colas et al., 2022), GN (GraphNarrative) (Shi et al., 2023), and TEKGEN (Agarwal et al., 2021). In our training phases, we use the commonly employed cross-entropy loss for generation tasks to align the model's predictions with the grounding generation, which is our PlanGTG dataset. In the full-shot experiments, after fine-tuning on PlanGTG, we continued to fine-tune the models on the training sets of WebNLG 2017. For more details on our training experiments, please refer to the App. E.

5.2 Model Performance

Zero-shot: Table 4 demonstrates that integrating the two introduced subtasks significantly enhances our fine-tuned model compared to the baseline LLaMA2-7b-chat and Mistral-7b-chat models. Specifically: 1) Compared to the untuned LLaMA2-7b-chat model, we achieve an average increase of 5.99 points in the BLEU metric and an average increase of 0.3 points in the BARTScore metric; 2) For the other automatically extracted datasets, we observe that their performance is even worse than that of the untuned LLMs. This suggests that domain differences between the extracted

Model	Dataset	WebNLG17				WebNLG20				DART			
	#Metrics	B-4	ME	CF	BS	B-4	ME	CF	BS	B-4	ME	CF	BS
LLaMA2-7b-chat	Zero	17.32	28.11	45.82	-2.85	17.45	23.62	38.94	-2.81	14.14	29.29	46.58	-2.85
	EVENT	5.26	20.93	32.03	-3.32	5.83	21.37	30.89	-3.67	5.47	20.48	32.13	-3.49
	GN	9.99	22.78	36.05	-3.14	9.04	20.76	33.14	-3.35	12.33	22.63	39.48	-3.55
	TEKGEN	7.21	15.64	27.06	-3.96	4.74	14.22	24.89	-3.78	12.04	22.06	33.76	-3.70
	Ours	**28.76**	**30.88**	**50.80**	**-2.38**	**20.60**	**25.50**	**42.16**	**-2.51**	**21.44**	**31.58**	**51.30**	**-2.72**
Mistral-7b-chat	Zero	16.60	26.14	45.98	-2.67	17.42	23.07	37.50	-2.74	15.25	26.23	47.71	-3.07
	EVENT	6.48	23.06	34.86	-3.01	6.34	22.34	33.71	-2.99	7.87	23.50	37.70	-3.48
	GN	10.44	23.23	35.27	-3.08	10.20	20.98	31.75	-3.08	17.44	24.43	41.95	-3.22
	TEKGEN	8.18	21.46	32.81	-2.82	4.92	18.86	29.11	-2.81	13.44	25.52	37.93	-3.10
	Ours	**29.76**	**31.30**	**51.17**	**-2.36**	**19.95**	**25.79**	**42.06**	**-2.43**	**27.46**	**30.50**	**50.21**	**-2.69**

Table 4: Zero-shot performance of different methods for graph-to-text generation on three domains. B-4, ME, CF and BS are short for BLEU-4, Meteor, CHRF++ and Bartscore. GN is short for GraphNarrative.

#Metrics	B-4			ME			CF			BS		
Methods	All	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen
Direct FT	45.32	49.83	39.53	35.47	37.50	33.06	62.04	65.16	58.18	-2.17	-2.06	-2.29
Instruction FT	36.34	40.09	31.54	33.39	34.89	31.62	56.10	58.61	53.01	-2.54	-2.48	-2.61
EVENT	36.76	39.16	33.69	33.39	33.72	32.99	56.36	57.38	55.13	-2.4	-2.46	-2.34
GN	43.18	47.58	37.56	36.86	38.64	34.76	61.95	65.12	58.03	-2.38	-2.28	-2.40
TEKGEN	44.12	48.40	38.68	37.02	38.76	34.95	62.55	65.50	58.93	-2.30	-2.14	-2.38
Ours	**46.35**	**50.40**	**41.17**	**37.94**	**39.71**	**35.84**	**63.92**	**66.95**	**60.18**	**-1.91**	**-1.83**	**-2.00**

Table 5: Performance of LLaMA2-7b-chat on WebNLG17 test set when fine-tuned with EVENT, GraphNarrative, TEKGEN and further fine-tuned with WebNLG17. Direct FT denotes that we directly fine-tune the model on WebNLG17 without adding instructions. Instruction FT adopts the same instructions as the second instruction in Appendix E.

Dataset	WebNLG17				WebNLG20				DART			
#Metrics	B-4	ME	CF	BS	B-4	ME	CF	BS	B-4	ME	CF	BS
PlanGTG	21.13	28.69	48.96	-2.41	17.62	23.20	39.04	-2.60	20.45	29.37	47.10	-2.82
+reorder	10.24	25.95	42.71	-2.72	9.45	21.35	35.19	-2.75	9.45	26.64	43.17	-2.98
+attribution	18.41	26.85	46.18	-2.54	16.10	23.87	38.86	-2.71	15.84	27.08	44.48	-2.89
+both (ours)	**28.76**	**30.88**	**50.80**	**-2.38**	**20.60**	**25.50**	**42.16**	**-2.51**	**21.44**	**31.58**	**51.30**	**-2.72**

Table 6: Ablation study for different modules on WebNLG17, WebNLG20 and DART.

datasets and the downstream tasks are magnified by LLMs. The datasets previously suitable for conventional model pretraining may not translate effectively to LLMs (Gardent et al., 2017b; Li et al., 2020; Nan et al., 2021). Thus, fine-tuning LLMs on PlanGTG demonstrates enhanced domain generalization capabilities.

Full-shot: To validate the benefit of additional fine-tuning models with PlanGTG on downstream tasks, we conducted additional fine-tuning on WebNLG 2017 for 5 epochs using the checkpoint from the last epoch for testing. This fine-tuning employed the LLaMA2-7b-chat model. For baselines, in addition to the pre-training datasets mentioned in the zero-shot section, we also fine-tuned LLMs using the WebNLG 2017 training dataset, both with and without instructions, following the methodology in our zero/few-shot experiments.

Results presented in Table 5 reveal a surprising decrease in performance when instructions are added compared to fine-tuning alone. This might be attributed to the model's utilization of instructions during inference, leading to outputs that deviate from the standard graph-to-text task format. However, our method, which incorporates instruction-tuning, exhibits significant improvements over all baselines. Regarding results on both seen and unseen data, our method outperforms the best baseline (Direct FT) by 0.57 BLEU and 0.23 BARTScore points on seen data, and by 1.64 BLEU and 0.29 BARTScore points on unseen datasets. Similar trends could be observed across the METEOR and CHRF++ metrics, indicating that our method can effectively enhance the model's generalizability capabilities on downstream tasks.

5.3 Ablation Study

To further analyze the impact of the two proposed subtasks on model performance, we fine-tuned the LLaMA2-7b-chat model on each subtask and evaluated its zero-shot performance.

From results in Table 6, we observe that the performance of PlanGTG with instruction fine-tuning surpasses that of the baseline in Table 4. This high-

Dataset	WebNLG17				WebNLG20				DART			
Methods	B-4	ME	CF	BS	B-4	ME	CF	BS	B-4	ME	CF	BS
One-pass	23.85	34.44	55.67	-2.16	25.61	**30.97**	47.60	-2.34	21.67	31.00	51.41	-2.54
Baby-steps	**28.47**	**38.00**	**57.84**	-2.39	**27.82**	30.60	**49.96**	-2.61	**26.99**	**33.30**	**54.60**	-2.81
Annealing	25.40	34.09	55.16	**-2.10**	27.00	30.09	49.60	**-2.33**	25.17	32.79	53.29	**-2.51**
Ours	21.13	28.69	48.96	-2.41	17.62	23.20	39.04	-2.6	20.45	29.37	47.10	-2.82

Table 7: Results achieved by fine-tuning the LLaMA2-7b-chat model on the PlanGTG dataset, without incorporating reordering and attribution, using various curriculum methods.

Dataset	Hallucinated Entities	Missed Entities	Hallucinated Relations	Missed Relations	Grammar
Gold	0.56	0.08	0.71	0.07	4.58
LC	2.81	1.95	1.81	1.77	3.09
Ours	1.63	1.27	1.49	1.12	3.88

Table 8: Human evaluation of sentences generated from our model trained with PlanGTG and LLaMA2-7b (LC).

lights the effectiveness of PlanGTG even without incorporating the two subtasks. Moreover, incorporating these subtasks leads to additional improvements in model performance, indicating their beneficial impact on enhancing task understanding and output quality.

However, when evaluating the subtasks individually (+reorder and +attribution), the results are less favorable. Introducing the reorder task alone may introduce noise as the model may not fully grasp the significance of the sequence of numbers. Similarly, introducing the attribution task alone could cause a significant mismatch between the model-generated text and the sequence of triplets, thereby degrading text quality.

5.4 Human Analysis

Human Evaluation Following the standards in Section 3.1, we conducted a human evaluation involving two assessments of the generated text. We randomly selected 200 graph-to-text pairs from the WebNLG17 dataset, paired with generated texts from the LLaMA2-7b-chat model and LLaMA2-7b-chat model trained with PlanGTG.

Results in Table 8 show that our model produces texts with higher fidelity and fluency compared to baseline models. In our approach, the reordering subtask improves alignment between the model-generated text and the knowledge graph, while the attribution subtask enhances the model's interpretability of its generated text.

5.5 Impact of Curriculum Learning

It has been observed that the selection of demonstrations plays a significant role and the difficulty of training examples has an impact on the model results during instruction fine-tuning (Lee et al., 2024). However, it is unknown whether this also applies to graph-to-text tasks. Given that knowledge graphs contain structured information, we used the number of triplets in a KG as a measure of complexity. We leveraged curriculum learning to analyze the impact of progressing from simple to complex learning on the model performance in the graph-to-text instruction fine-tuning process, using PlanGTG without the two subtasks for training. Specifically, we tested three classic curriculum algorithms: (1) **One-pass** (Bengio et al., 2009), (2) **Baby-steps**, (Spitkovsky et al., 2010) (3) **Annealing** (Xu et al., 2020), more details are shown in App. G. The rest of the training settings are consistent with the experiments in Sec. 5.3.

Results in Table 7 show that compared to instruction fine-tuning without using curriculum learning, all three curriculum learning methods enhance model performance. Among them, Baby-steps obtains the best results, followed by Annealing, with One-step being the least effective. This suggests that the model may gradually forget the learning of simple graphs (e.g, with 1 or 2 triplets) for text alignment during the learning process from simple to complex. This could potentially impair the model's ability to generate simple sentences, thereby impacting its overall performance.

6 Conclusion

We have conducted a comprehensive analysis over both zero- and few-shot scenarios in graph-to-text generation tasks to evaluate the capabilities and challenges of LLMs. Our findings reveal that (1) LLMs struggle to understand complex graphs and (2) moderately difficult and diverse demonstrations may help LLMs for translating graphs to text. Based on these findings, we propose solutions from both few-shot learning and fine-tuning perspectives to enhance the effectiveness of LLMs. In terms of few-shot prompting, we propose the DDD method to select samples considering both difficulty and diversity simultaneously, leading to improvements in both the one-shot and few-shot cases. For model fine-tuning, we construct a high-quality graph-to-text dataset, PlanGTG, and develop two new subtasks. Fine-tuning LLMs on PlanGTG demonstrates a significant improvement in the alignment generation and generalization abilities of LLMs. Additionally, through learning from simple to complex data, the model's ability to generate text from graphs is further enhanced. Our work lays the groundwork for future research aimed at effectively enabling LLMs to reorganize graph structures and identify sequential information in generated texts.

Limitations

We identify the following limitations related to our approach and experiments. Firstly, due to computational resource constraints, we do not evaluate larger models, such as LLaMA2-70b. Moreover, during fine-tuning, we adopt LoRA, a parameter-efficient fine-tuning method, which, compared to full-parameter fine-tuning, may result in some performance trade-offs compared to full-parameter fine-tuning approaches. Additionally, we face limitations in both budget and computational resources, restricting us from scaling up the dataset or conducting fine-tuning on larger datasets. As a result, PlanGTG is not very large in size. Furthermore, given the focus of the paper on exploring graph-to-text generation tasks in the era of LLMs, we have not extensively investigated the two new subtasks introduced.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Garima Agrawal, Tharindu Kumarage, Zeyad Alghami, and Huan Liu. 2023. Can knowledge graphs reduce hallucinations in llms? : A survey.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z. Pan. 2021a. Negative Knowledge for Open-world Wikidata. In *In the Web Conference (WWW 2021)*, pages 544–551.

Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z. Pan. 2021b. Negative Statements Considered Useful.

Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z. Pan. 2022a. UnCommonSense: Informative Negative Knowledge about Everyday Concepts. In *In the 31st ACM International Conference on Information and Knowledge Management (CIKM 2022)*.

Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z. Pan. 2022b. UnCommonSense: Informative Negative Knowledge about Everyday Concepts. In *In the 31st ACM International Conference on Information and Knowledge Management (CIKM 2022)*.

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.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for 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, and . Maarten Bosma etc. 2022. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 2022. Eventnarrative: A large-scale event-centric dataset for knowledge graph-to-text generation.

Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald Metzler, et al. 2023. Parade: Passage ranking using demonstrations with large language models. *arXiv preprint arXiv:2310.14408*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. 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.

Yuxia Geng, Jiaoyan Chen, Wen Zhang, Jeff Z. Pan, Mingyang Chen, Huajun Chen, and Song Jiang. 2023. Relational Message Passing for Fully Inductive Knowledge Graph Completion. In *In the 39th IEEE International Conference on Data Engineering (ICDE 2023)*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely Connected Graph Convolutional Networks for Graph-to-Sequence Learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

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

Zhiwei Hu, Victor Basulto, Zhiliang Xiang, Ru Li, and Jeff Z. Pan. 2023a. Multi-view Contrastive Learning for Entity Typing over Knowledge Graphs. In *In Proc. of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023),*, pages 12950–12963.

Zhiwei Hu, Victor Gutierrez-Basulto, Zhiliang Xiang, Ru Li, and Jeff Z. Pan. 2022. Transformer-based Entity Typing in Knowledge Graphs. In *In the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*.

Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Ru Li, and Jeff Z. Pan. 2023b. HyperFormer: Enhancing Entity and Relation Interaction for Hyper-Relational Knowledge Graph Completion. In *In Proc. of the 32nd ACM International Conference on Information and Knowledge Management (CIKM 2023)*.

Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Z. Pan. 2024. Less is More: Making Smaller Language Models Competent Subgraph Retrievers for Multi-hop KGQA. In *In the Findings of the Association for Computational Linguistics: EMNLP 2024 (EMNLP 2024 Findings)*, pages 15787–15803.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Bruce W. Lee, Hyunsoo Cho, and Kang Min Yoo. 2024. Instruction tuning with human curriculum.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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 pretrained language models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1558–1568, Online. Association for Computational Linguistics.

Shengzhi Li and Nima Tajbakhsh. 2023. Scigraphqa: A large-scale synthetic multi-turn question-answering dataset for scientific graphs.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Xintong Li, Aleksandre Maskharashvili, Symon Jory Stevens-Guille, and Michael White. 2020. Leveraging large pretrained models for WebNLG 2020. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic*

Web (WebNLG+), pages 117–124, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022a. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Jin Liu, Chongfeng Fan, Zhou Fengyu, and Huijuan Xu. 2022b. Syntax controlled knowledge graph-to-text generation with order and semantic consistency. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1278–1291, Seattle, United States. Association for Computational Linguistics.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y Zhao. 2023. Dr. icl: Demonstration-retrieved in-context learning. *arXiv preprint arXiv:2305.14128*.

Chris Mellish and Jeff Z. Pan. 2008. Natural Language Directed Inference from Ontologie". In *Artificial Intelligence Journal*.

Igor Melnyk, Pierre L. Dognin, and Payel Das. 2022. Knowledge graph generation from text. In *EMNLP (Findings)*, pages 1610–1622. Association for Computational Linguistics.

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

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. `https://openai.com/blog/chatgpt`.

OpenAI. 2023. Gpt-4 technical report.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large Language Models and Knowledge Graphs: Opportunities and Challenges. pages 1–38.

J.Z. Pan, D. Calvanese, T. Eiter, I. Horrocks, M. Kifer, F. Lin, and Y. Zhao, editors. 2017a. *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Querying Answering*. Springer.

J.Z. Pan, G. Vetere, J.M. Gomez-Perez, and H. Wu, editors. 2017b. *Exploiting Linked Data and Knowledge Graphs for Large Organisations*. Springer.

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.

Artemis Parvizi, Chris Mellish, Kees van Deemter, Yuan Ren, and Jeff Z. Pan. 2014. Selecting Ontology Entailments for Presentation to Users. In *Proc. of the International Conference on Knowledge Engineering and Ontology Development (KEOD 2014).*

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

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

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6908–6915.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. 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 FR Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense Properties from Query Logs and Question Answering Forums. In *Proc. of 28th ACM International Conference on Information and Knowledge Management (CIKM 2019)*, pages 1411–1420.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

Xiao Shi, Zhengyuan Zhu, Zeyu Zhang, and Chengkai Li. 2023. Hallucination mitigation in natural language generation from large-scale open-domain knowledge graphs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12506–12521, Singapore. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California. Association for Computational Linguistics.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment.

Kemas Wiharja, Jeff Z. Pan, Martin J. Kollingbaum, and Yu Deng. 2020. Schema Aware Iterative Knowledge Graph Completion. *Journal of Web Semantics*.

Yike Wu, Yi Huang, Nan Hu, Yuncheng Hua, Guilin Qi, Jiaoyan Chen, and Jeff Z. Pan. 2024. CoTKR: Chain-of-Thought Enhanced Knowledge Rewriting for Complex Knowledge Graph Question Answering. In *In Proc. of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*, pages 3501–3520.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104, Online. Association for Computational Linguistics.

Yijun Yang, Jie He, Pinzhen Chen, Victor Gutierrez Basulto, and Jeff Z. Pan. 2024. UniArk: Improving Generalisation and Consistency for Factual Knowledge Extraction through Debiasing. In *In Proc. of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*, pages 7018–7035.

Zixiaofan Yang, Arash Einolghozati, Hakan Inan, Keith Diedrick, Angela Fan, Pinar Donmez, and Sonal Gupta. 2020. Improving text-to-text pre-trained models for the graph-to-text task. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 107–116, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations (ICLR)*.

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

Feng Zhao, Hongzhi Zou, and Cheng Yan. 2023a. Structure-aware knowledge graph-to-text generation with planning selection and similarity distinction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8693–8703, Singapore. Association for Computational Linguistics.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023b. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the*

Dataset	# train	# dev	# test	Average triplets	Average Words
DART	30,526	2,768	5,097	3.62	20.95
WebNLG17 (seen)	6,940	872	971	3.02	20.26
WebNLG17 (unseen)			891	2.75	19.00
WebNLG20 (seen)	13,211	1,666	883	3.63	24.36
WebNLG20 (unseen)			896	2.71	19.64

Table 9: Statistics for the graph-to-text datasets.

Association for Computational Linguistics (Volume 1: Long Papers), pages 5823–5840, Toronto, Canada. Association for Computational Linguistics.

Danna Zheng, Danyang Liu, Mirella Lapata, and Jeff Z. Pan. 2024. TrustScore: Reference-Free Evaluation of LLM Response Trustworthiness. In *In Proc. of the ICLR 2024 Workshop on Secure and Trustworthy Large Language Models (Set LLM 2024)*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Datasets

WebNLG17 Challenge (Gardent et al., 2017a): A standard graph-to-text dataset, with each instance being composed of a graph from DBpedia and corresponding text annotated by humans. The test set is divided into the seen and unseen partitions respectively. The unseen partition includes 5 categories absent from the training and development sets.

WebNLG20 Challenge (Castro Ferreira et al., 2020): It includes 10 categories carried over from WebNLG17 and 5 additional new categories that are not present in the 2017 dataset. Furthermore, this edition introduces a brand-new category “Company”.

DART (Nan et al., 2021): A collection of graph-to-text pairs, which have been compiled from multiple sources, such as WebNLG and E2E (Dušek et al., 2018), along with sentences obtained via crowd-sourcing and matching tables sourced from WikiSQL (Zhong et al., 2017) and WikiTableQuestions (Pasupat and Liang, 2015). We perform the same partition as (Zhao et al., 2023a). The statistics of the datasets are shown in Table 9.

B Evaluation Metric Details

Human Evaluation Following (Shi et al., 2023), we assessed the quality of various pretraining datasets and the sentences generated by models. Specifically, our evaluation examined if sentences either from the dataset or generated by models introduced facts not included in the corresponding graphs or failed to mention details. Our analysis employed four metrics: the number of **hallucinated entities** (entities mentioned in the sentence but absent in the graph), **missed entities** (entities omitted in the sentence but present in the graph), **hallucinated relations** (relations mentioned in the sentence but absent in the graph), and **missed relations** (relations omitted in the sentence but present in the graph). Besides, we evaluated the **grammatical correctness and fluency** of the generated text. This evaluation utilized a 5-point Likert (Likert, 1932) scale, ranging from 1-point (indicating "very poor") to 5-points (indicating "highly satisfactory"). We present an annotation interface and a corresponding example in Fig. 2 to demonstrate how humans annotate the quality of PlanGTG.

Automatic Evaluation We used four common automatic metrics to assess graph-to-text generation: BLEU (B-4) (Papineni et al., 2002), METEOR (ME) (Banerjee and Lavie, 2005), CHRF++ (CF) (Popović, 2015) and BartScore (BS) (Yuan et al., 2021). Specifically, BLEU measures the n-gram overlaps between the generated text and reference text. We set n to 4. CHRF++ computes the F-score averaged on both character and word-level n-grams. METEOR considers the semantic matches between source and reference text and BartScore uses BART (Lewis et al., 2020) to measure the quality of the generated text.

C Preliminary Evaluation

C.1 Prompt Searching

Instruction Format Exploration Since prompt is the key to the interaction between the LLMs and humans, we started by investigating how the verbalization of instructions influences the downstream performance. For system prompts, we constructed a simple version where only a general instruction is given and a detailed version where the definition of triplets and a detailed instruction are provided. For the user prompt, we constructed four instructions sorted by their level of detail in the prompts from A to D. In addition, to simulate the controllable text generation in industrial applications, we constructed D*, where an additional task of writing triplet dollar signs after generating the text is added. Table 11 presents the performance results for WebNLG20 and WebNLG. The prompt templates used are shown as follows

Task Description

You are asked to annotate the quality of a graph-text pair based on the following criteria:

- **Hallucinated Entities**: The number of entities mentioned in the sentence but absent in the graph.
- **Missed Entities**: The number of entities present in the graph but omitted in the sentence.
- **Hallucinated Relations**: The number of relations mentioned in the sentence but absent in the graph.
- **Missed Relations**: The number of relations present in the graph but omitted in the sentence.
- **Grammatical Correctness and Fluency**: Provide a score between 1-5 where:
  - 1: Very poor
  - 2: Poor
  - 3: Neutral
  - 4: Fluent
  - 5: Very natural

Graph

Indonesia | citytown | Bogor (1) Bogor | population | 950,334 (2) Bogor | altitude | 200 meters (3)

Text

Located in Indonesia, Bogor is a city with a population of 950,334 and an altitude of 200 meters.

Annotation Information

- **The number of hallucinated entities:** ________________________
- **Missed entities:** ________________________
- **Hallucinated relations:** ________________________
- **Missed relations:** ________________________
- **Grammatical correctness and fluency:** ________________________ (1-5)

Figure 2: Human evaluation task annotation interface for the PlanGTG dataset.

Method	WebNLG17_unseen				WebNLG20_unseen				DART			
	B-4	ME	CF	BS	B-4	ME	CF	BS	B-4	ME	CF	BS
<head>	18.46	34.53	54.15	-2.13	22.87	34.74	56.10	-2.03	11.57	29.97	45.68	-2.70
<head></head>	18.64	33.81	53.37	-2.17	**24.34**	30.54	50.46	-2.45	12.86	30.54	47.26	-2.68
head | relation	**19.67**	**34.99**	**54.97**	**-2.10**	23.32	**35.48**	**57.07**	**-1.97**	**13.43**	**31.09**	**48.17**	**-2.60**

Table 10: Influence of the triplet formulation.

SYS Prompt	USR Prompt	WebNLG20_all				WebNLG17_all				DART			
		B-4	ME	CF	BS	B-4	ME	CF	BS	B-4	ME	CF	BS
Simple	A	8.97	24.12	42.59	-2.70	9.03	23.90	41.94	-2.81	5.57	21.79	36.92	-3.29
	B	10.80	25.09	43.86	-2.58	9.35	22.40	41.84	-2.91	5.77	21.32	35.80	-3.41
	C	10.03	24.76	42.53	-2.67	9.68	24.75	42.27	-2.66	5.58	22.08	36.91	-3.17
	D	*24.22*	*34.65*	*58.07*	-2.07	**22.72**	*34.65*	*53.12*	**-2.09**	**15.26**	*30.71*	*47.87*	*-2.62*
	D*	21.66	33.84	54.89	-2.10	18.12	32.53	51.78	-2.22	*13.44*	30.52	47.33	-2.68
Detailed	A	15.28	31.64	49.70	-2.10	13.77	31.29	47.87	-2.33	10.12	28.32	43.12	-2.79
	B	14.33	31.07	48.70	-2.21	14.37	31.68	48.64	-2.32	10.49	28.60	43.79	-2.81
	C	20.88	34.27	55.74	*-2.03*	17.69	33.59	52.34	-2.17	12.38	29.75	46.22	-2.70
	D	24.00	**36.13**	**58.17**	**-1.91**	*19.39*	**34.93**	**54.95**	**-2.09**	13.43	**31.00**	**48.04**	**-2.61**
	D*	**24.29**	29.69	49.17	-2.61	18.06	33.64	43.00	-2.85	11.03	26.42	40.70	-3.09

Table 11: Zero-shot LLaMA2 performance between different prompts. The best results are marked **bold** and the second best results are marked *italics*

Simple: "Following the questions and give directly the answers. Do not include any additional information or outputs."
Detailed:"You are skilled in interpreting knowledge graphs. Your task is to transform a series of triplets, each consisting of a subject, predicate, and object, into a well-written, coherent paragraph. These triplets are formatted as 'subject | predicate | object' and are separated by lines. Please provide only the transformed text as your output. Do not include any additional information or outputs."

A:"{triplets} || Text: "
B:"Graph: {triplets} || Text: "
C:"Graph: {triplets} || Convert the graph into text: "
D:"Following is a set of knowledge graph triplets delimited by triplet backticks, each on a separate line, in the format: subject | predicate | object.
‘
‘ {triplets}
‘
‘ Only use information from the provided triplets and convert the graph into a coherent piece of text:"
D*:"Following is a set of knowledge graph triplets delimited by triplet backticks, each on a separate line, in the format: subject | predicate | object.
‘
‘ {triplets}
‘
‘ Only use information from the provided triplets and generate a coherent piece of text that contains all of the information in the triplets. After you finish writing the piece of text, write triplet dollar signs (i.e.: $$$)."

From the results in Table 11, we can draw the following conclusions: (1) When the user prompt is less detailed, giving detailed system prompts in-

Dataset	Linearization	B-4	ME	CF	BS
WebNLG17_all	RS	28.39	28.61	52.21	-2.56
	ORI	28.55	28.70	52.15	**-2.54**
	BFS	28.61	**28.74**	**52.37**	**-2.54**
	DFS	28.59	28.65	52.20	**-2.54**
	GPT	**28.89**	28.34	51.64	-2.56
WebNLG20_all	RS	31.64	30.22	54.93	-2.42
	ORI	32.37	30.60	55.60	-2.37
	BFS	**32.44**	30.64	55.62	-2.36
	DFS	32.34	30.55	55.47	-2.37
	GPT	30.58	**32.13**	**57.66**	**-2.26**
DART	RS	**22.45**	**27.83**	51.79	-2.99
	ORI	20.32	27.69	51.72	-3.01
	BFS	20.35	27.72	**51.80**	**-3.02**
	DFS	20.32	27.70	51.76	**-3.02**
	GPT	20.93	27.40	51.04	**-3.02**

Table 12: Comparison between different linearizations

stead helps the quality of the generation, especially for the semantics level since a significant improvement is observed on CHRF++ and Bart score. This suggests that a detailed description of the task may hint the model to generate more fluent and human-preferred answers. (2) More detailed user prompts may result in better generations as stable improvements for all metrics are obtained for prompts of type A to D. We also observe a sudden growth in the performance for both system settings for types C to D in all datasets. (3) Including a controlled text generation task (user prompt D*) may harm the quality of the generation since a significant drop in all metrics is observed. This is possibly due to the weaker capabilities of smaller LLMs such as LLaMA-7b, where multiple tasks may interfere with each other. Based on the above findings, we therefore choose the prompt settings with the best performance (a detailed system prompt with the template D and the third triplet format) as the input in the other experiments.

triplet representation We also want to explore the influence of different verbalizations of triplets within the graphs. To this end, we experiment with three popular verbalizations of triplets. Table 10 presents the results, where a close performance of each verbalization is observed. This suggests that LLMs have a good understanding of triplet expressions.

Influence of Linearization Massive works have demonstrated the impact of various linearization techniques on graph-to-text tasks (Yang et al., 2020; Hoyle et al., 2021; Li et al., 2021). Here we investigated how different linearizations may affect the results of LLMs. Beyond the original linearization (ORI) provided in a dataset, we explored alternative arrangements by reordering the triplets according to fixed tree traversal methods, including breadth-first search (BFS), depth-first search (DFS), and a random sequence (RS). Additionally, we employed GPT to generate a 'silver' linearization derived from the ground truth text, detailed prompts can be found in the Fig 11. The results of experiments performed on the Mistral-7b-instruct-v0.2 are shown in Table 12. They reveal marginal differences among the linearizations, with BFS and DFS exhibiting slightly enhanced performance. The discrepancy between random and silver linearizations is also minimal, indicating that LLMs demonstrate robustness to the variety of input graph linearizations. Consequently, we opt for the ORI linearization for subsequent experiments in our study.

C.2 Few-shot Sample Selection

Few-shot demonstrations have widely been shown to be crucial on the performance of generations (Luo et al., 2023; Drozdov et al., 2023). We conducted experiments on how different example selection strategies influence the performance of the graph-to-text generation, focusing on criteria such as difficulty and diversity. Experiment results inspire us to propose an optimal strategy for demonstration selection.

C.2.1 Demonstration Selection Methods

To measure the diversity and difficulty, we first map the graph into continuous vectors using a state-of-the-art sentence encoder[3] and then use the embeddings to achieve the difficulty and diversity-based demonstration selection.

We assess difficulty by computing the cosine similarity between examples and input. It is based on the assumption that following similar demonstrations facilitates graph-to-text generation. Consistent patterns in similar demonstrations enable LLMs to produce coherent text that fits the norms of the demonstrated examples. We categorize difficulty into five levels based on cosine similarity scores between input embeddings and all embeddings in the training sets. For experimentation, we uniformly select examples ranging from the easiest (Difficulty level 0) to the most challenging (Difficulty level 4) and conducted one-shot inference.

Inspired by Liu et al. (2022a), we measure the diversity of examples based on different k-means clusters and conducted a 3-shot experiment. We

[3]We used the SoTA sentence embedding encoder UAE (Li and Li, 2023) on the MTEB leader board.

Models	WebNLG17_all				WebNLG20_all				DART			
	B-4	ME	CF	BS	B-4	ME	CF	BS	B-4	ME	CF	BS
Instruct-Models												
ChatGLM3-6b	12.69	26.28	45.27	-2.53	17.16	29.03	49.75	-2.34	17.21	30.61	50.27	-2.30
Vicuna-7b-v1.5	13.89	30.70	47.45	-2.34	17.20	30.86	49.04	-2.21	14.23	31.03	47.87	-2.36
Zephyr-7b-beta	17.05	34.74	53.74	-2.08	21.51	36.01	56.83	-1.95	12.27	31.50	47.49	-2.28
Falcon-7b-instruct	13.77	26.65	43.87	-2.96	15.35	26.79	44.42	-2.86	10.95	22.10	38.06	-3.33
LLaMA2-7b-chat-hf	21.51	35.67	56.38	-2.04	24.68	35.84	57.82	-1.96	15.47	31.94	49.70	-2.27
Gemma-1.1-7b-it	19.77	30.30	50.40	-2.72	18.10	27.08	45.48	-3.01	12.36	25.56	42.33	-3.09
Mistral-7b-Instruct-v0.2	**27.09**	27.29	45.43	-2.73	**31.12**	29.35	49.12	-2.54	21.48	24.22	41.00	-3.07
LLaMA2-13b-chat-hf	22.51	<u>36.31</u>	<u>57.62</u>	<u>-1.95</u>	27.70	<u>37.63</u>	<u>60.47</u>	<u>-1.83</u>	17.47	<u>33.37</u>	52.04	<u>-2.14</u>
Vicuna-13b-v1.5	25.19	35.24	56.95	-2.06	<u>30.82</u>	35.60	57.88	-1.91	<u>22.42</u>	32.35	51.96	-2.28
Falcon-40b-instruct	21.02	34.17	55.03	-2.20	26.80	35.40	58.00	-2.07	20.78	33.31	<u>53.68</u>	-2.25
Mixtral-8 $\times$ 7b	<u>26.42</u>	**38.06**	**60.70**	**-1.84**	30.49	**38.41**	**62.07**	**-1.78**	**22.89**	**35.42**	**56.04**	**-2.00**
Base-Models												
Falcon-7b-base	3.82	15.93	26.17	-2.26	3.48	15.15	26.03	-2.36	3.36	16.27	25.16	-2.44
Gemma-7b	0.08	1.73	3.93	-5.25	0.05	1.44	3.56	-5.33	0.09	1.57	3.85	-5.18
LLaMA2-7b-hf	4.57	16.61	27.94	-2.83	4.42	16.13	28.64	-2.98	3.98	16.87	26.37	-2.82
LLaMA2-13b-hf	3.83	14.57	26.35	-3.02	4.20	15.11	27.51	-3.15	3.25	14.52	24.25	-3.11

Table 13: Graph-to-text generation performance of the tested LLMs. The best results are bold and the second best results are underlined.

Dataset	Method	Shot	BLE	MET	CHR+	BRT
DART	Difficulty	1	19.30	33.14	54.01	-2.39
	DDD	1	**21.33**	33.69	55.10	-2.36
	Diversity	3	21.31	33.71	55.00	-2.35
	DDD	3	21.30	**33.78**	**55.17**	**-2.33**
WebNLG17	Difficulty	1	22.46	35.27	57.20	-1.96
	DDD	1	23.48	35.49	57.66	-1.92
	Diversity	3	**25.19**	35.61	58.30	-1.93
	DDD	3	25.07	**35.98**	**58.60**	**-1.91**
WebNLG20	Difficulty	1	24.92	35.81	57.76	-1.98
	DDD	1	26.07	**38.01**	58.64	**-1.97**
	Diversity	3	26.65	35.92	58.32	-2.02
	DDD	3	**27.54**	36.09	**58.93**	-2.00

Table 14: Comparison between different samplings on DART, WebNLG17 unseen and WebNLG20 unseen datasets.

introduced a four-tier diversity-based sampling framework designed to enhance example selection. Initially, we identify the least diverse examples by selecting the nearest n points to the input embeddings (Level 0). For Level 1, we sample n points within the same cluster as the input. At Level 2, we select the centers of the n closest clusters to the input's cluster. Finally, Level 3 involves a uniform selection of cluster centers based on their proximity to the input, specifically choosing the centers of the first, fifth, and tenth nearest clusters from a total of ten in our evaluation.

For each strategy, different verbalization of demonstrations are also considered. we calculate the similarity or k-means not only between graphs of the demonstration-input pairs, but also between the text of the input obtained by a zero-shot inference. Furthermore, a standardized graph, where entities are replaced with "anonymous" entities $\langle ent \rangle$, is constructed to study whether it is the semantic information in the graph or its topology that primarily influences graph verbalization.

C.2.2 Difficulty-based Sampling

From the results in Figure 6, we observe that: (1) Sampling by text similarity shows good performance in the BLEU metric across all samples. This suggests that it could be a good strategy when prioritizing n-gram accuracy and being provided additional inference for zero-shot text from input graphs is feasible; (2) There is no significant difference on the performance of standardized graphs and original graphs, suggesting that LLMs learn more from the structure of graphs than from the semantic information in entities; (3) When comparing the difficulties between selected demonstrations, we observe that while an easy demonstration works better on the datasets where the triplets have appeared in the training sets (e.g. DART), this trend does not hold for datasets with unseen samples in the test set (e.g., WebNLG). Instead, optimal performance typically peaks with samples of moderate difficulty (levels 1-3), indicating that selecting moderately challenging samples may be the optimal strategy

C.2.3 Diversity-based Sampling

From Figure 7, two main findings are: Firstly, text embedding-based demonstrations generally outperform others, indicating that diversity has limited impact when using text similarity as the selecting criteria. This suggests that zero-shot learning offers a more straightforward and efficient method for accessing detailed information than relying on graph embeddings. Secondly, the selection of examples based on graph embeddings underscores the significance of diversity. Contrary to the lowest diversity level (level 0), optimal performance is typically achieved at moderate diversity levels (levels 1 or 2). This emphasizes the effectiveness of a balanced approach to select examples, favoring samples from nearby clusters rather than those within the same cluster or from distant clusters.

C.2.4 DDD Selection

The full comparison results of DDD selection against difficulty-based selection and diversity-based selection are shown in Table 14.

C.3 Scaling Evaluations

We evaluate popular LLMs with the chosen prompt settings (necessary modifications are made to adapt the templates for different models) and default configs from the HuggingFace text generation pipeline for experiments. We set the maximum token limit for each model to ensure that we obtain complete results. For models around 7 billion parameters, we test ChatGLM-6b (Zeng et al., 2023), LLaMA2-7b (Touvron et al., 2023), Vicuna-7b (Chiang et al., 2023), Zephyr-7b (Tunstall et al., 2023), Mistral-7b (Jiang et al., 2023) and Falcon-7b (Almazrouei et al., 2023). For larger LLMs, we test, falcon-40b and Mixtral-45b (the official 8*7b mixture of experts version).The results are presented in Table 13. We also evaluate 4 base models. However, since we found that all base models suffer from following instructions and only generate the reference, we draw our conclusion mainly based on instruct models.

Our key findings are as follows. Firstly, 13b models outperform almost all 7b models, indicating a positive correlation between the scaling of LLMs and the graph-to-text generation performance. However, the increasing trend becomes slower when we keep scaling up the parameters. We hypothesize that this is because 13b models are already capable of understanding and reasoning well over graphs, scaling parameters may help LLMs to memorize more facts, but this is not helpful for graph-to-text verbalization. Additionally, we observe that a fine-tuned version of LLMs usually performs worse than the vanilla ones, as seen from Zephyr-7b to Mistral-7b and Vicuna-7b to LLaMA2-7b. This suggests that task-specific fine-tuning may compromise the graph verbalization capabilities of LLMs, which further motivates the design of tasks aimed at instruction-tuning LLMs to enhance their graph-to-text capabilities.

Dataset	#triplets (#samples)	B-4	ME	CF	BS
DART	1 (848)	23.36	38.93	61.22	-1.92
	2 (797)	17.55	32.98	53.59	-2.36
	3 (821)	17.09	32.57	53.24	-2.22
	4 (869)	17.21	31.68	52.89	-2.29
	5+ (1762)	11.79	29.59	47.08	-2.63
WebNLG17	1 (454)	30.73	41.65	70.16	-1.61
	2 (349)	29.82	30.93	55.14	-2.63
	3 (386)	26.54	27.41	50.51	-2.97
	4 (363)	25.07	24.99	46.75	-3.05
	5+ (310)	19.87	21.71	41.34	-3.32
WebNLG20	1 (369)	30.64	41.51	69.18	-1.66
	2 (349)	32.76	32.72	57.10	-2.44
	3 (350)	30.92	30.18	54.51	-2.60
	4 (305)	28.61	26.93	49.81	-2.79
	5+ (406)	28.88	26.94	51.38	-2.85

Table 15: Comparison between different graph triplet numbers on DART, WebNLG17_all, WebNLG20_all.

C.4 Influence of Graph Complexity

We also investigated the impact of graph complexity on the performance of graph-to-text generation by LLMs. Conducted in a zero-shot framework using Mistral-7b-instruct-v0.2, our experiments in Table 15 reveal a consistent decline in performance across all metrics as the number of triplets in a graph increases. This shows the difficulties that LLMs face in text generation from complex graphs. Additionally, we examine the effect of graph diameters (the longest-shortest path between any two vertices) on LLM performance. The results presented in Table 16 show that for WebNLG20 the variance in performance across different graph diameters is minimal, potentially suggesting model stability across various structures. However, the results for WebNLG17 exhibit a clear negative correlation between performance metrics and graph diameters, highlighting the current limitations of LLMs in handling diverse graph structures. This inconsistency underscores the necessity for further research to enhance LLM stability across different graph configurations.

D PlanGTG Details

D.1 Details for prompts in PlanGTG

The used prompts are shown in Fig. 9, Fig. 10, Fig. 8 respectively.

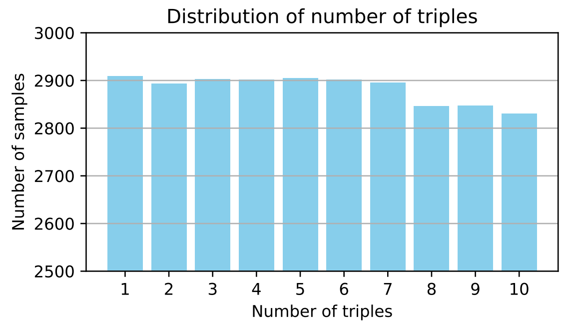

Figure 3: Bar chart of the distribution of the number of triplets in PlanGTG

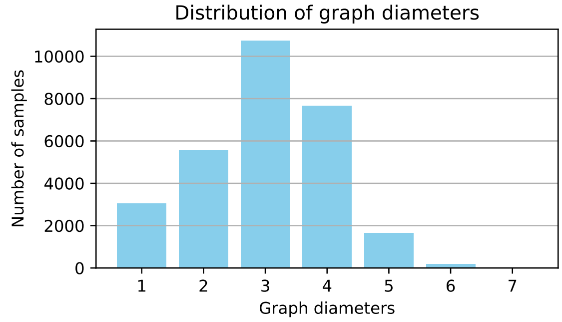

Figure 4: Bar chart of the distribution of the diameter of triplets in PlanGTG

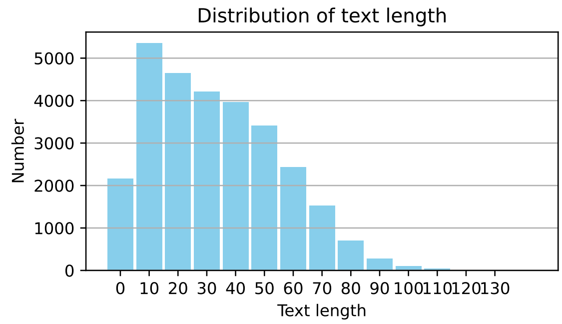

Figure 5: Bar chart of the distribution of the length of text in PlanGTG

D.2 Details for the distribution in PlanGTG

The distribution of number of triplets, graph diameters and text length of PlanGTG are shown in Fig. 3, Fig. 4 and Fig. 5 respectively.

E Details for our training experiments

We mainly used two popular LLMs to conduct the experiments: LLaMA2-7b-chat[4] and Mistral-7b-chat[5]. To optimize memory usage and accelerate training, we applied DeepSpeed Zero Stage 3 (Rasley et al., 2020) and bfloat16 mixed precision techniques. The learning rate was set at 2e-4 for all experiments with a batch size of 8, and the maximum length was 1024. All models were trained on 2 Tesla A100-80G GPUs.

[4]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[5]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

Dataset	#D (#samples)	B-4	ME	CF	BS
DART	0 (575)	23.29	38.92	61.19	-1.93
	1 (3463)	14.18	30.30	48.65	-2.53
	2 (850)	18.60	33.20	55.09	-2.00
	3 (201)	17.75	32.37	54.41	-2.02
	4 (8)	20.57	33.96	59.36	-2.24
WebNLG17	0 (454)	30.73	41.65	70.16	-1.61
	1 (930)	27.92	27.57	50.58	-2.84
	2 (337)	22.07	23.37	44.03	-3.18
	3 (136)	16.66	20.23	38.81	-3.44
	4 (5)	12.38	17.99	38.17	-3.64
WebNLG20	0 (369)	30.77	41.42	69.08	-1.66
	1 (1042)	29.84	28.47	51.86	-2.63
	2 (280)	30.60	27.71	52.66	-2.88
	3 (88)	30.80	29.29	55.57	-2.64

Table 16: Comparison between graph diameters (D).

When fine-tuning on our PlanGTG dataset, given that we involve two tasks, namely reorder and attribution, our instructions are set as follows:

> The following is a set of knowledge graph triplets delimited by triplet backticks, each on a separate line, in the format: subject | predicate | object. (number). The “number” indicates the sequence number of each triplet. “ triplets ” The task involves two steps: First, output the correct order of these knowledge triplets. Then, generate a coherent piece of text that incorporates all the information from the triplets. The generated text should include corresponding sequence numbers. Only the information provided in the triplets should be used. After you finish these two tasks, write triplet dollar signs (i.e.: $$$).

The input to the model follows the format: subject | predicate | object (number). And The model’s gold output format is: The correct triplet order is: subject | predicate | object (number). The generated text is: ... . Below is an example:

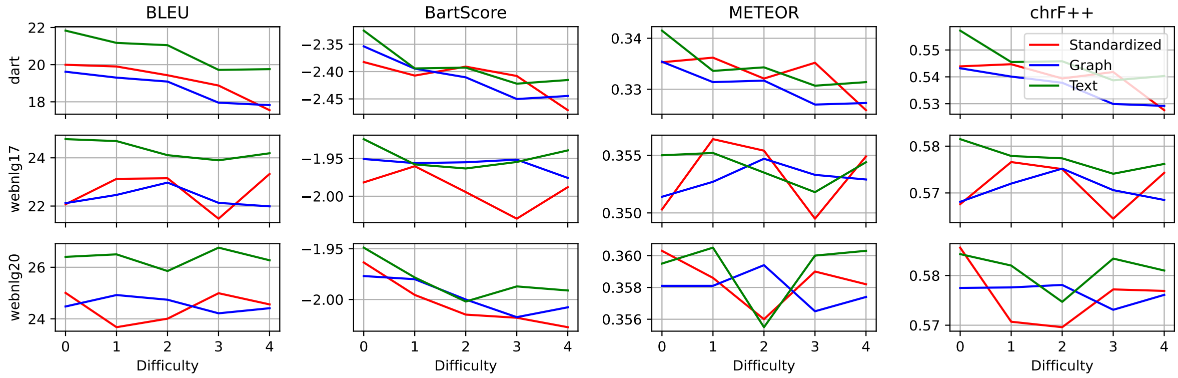

Figure 6: Results for all 4 metrics exploring the difficulty of examples in the one-shot learning setting

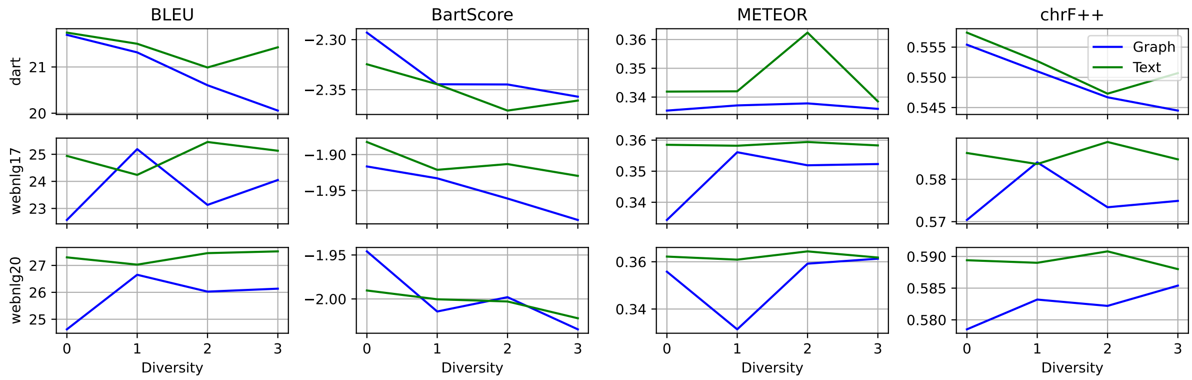

Figure 7: Results for all 4 metrics exploring the diversity of examples in the few-shot learning setting

Input:
Twitter | users | 330 million (1) Twitter | founding year | 2006 (2) Twitter | category | Internet forum (3) Internet forum | communication platform | online discussion platform (4)
Output:
The correct triplet order is: Twitter | users | 330 million (1) Internet forum | communication platform | online discussion platform (4) Twitter | category | Internet forum (3) Twitter | founding year | 2006 (2) **Then The generated text is**: Twitter has 330 million users (1), serving as an online communication platform for discussions (4), is categorized as an internet forum (3), and was founded in 2006 (2).

We finetune the model on our constructed PlanGTG instruction dataset, enabling the model to automatically learn to first output the correct order of the graph, and then generate text based on the reordered triplets. During inference, we extract the text following "Then the generated text is:" and remove the indices (e.g., (1)) before evaluating against the gold text. The fine-tuning loss we adopt is the commonly used cross-entropy loss for generation tasks, which measures the difference between the predicted probability distribution and the true distribution (i.e., the gold text). Specifically, the cross-entropy loss is defined as:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{V} y_{ij}\log(p_{ij})$$

where N is the sequence length, V is the vocabulary size, y_{ij} is the one-hot encoded vector for the gold token at position i, and p_{ij} is the predicted probability for token j at position i. The model is trained to minimize this loss, encouraging it to generate text that closely matches the gold output.

When fine-tuning on the WebNLG17 dataset, our instructions are set as follows because this dataset only provides annotations for graphs and their corresponding text:

The following is a set of knowledge graph triplets delimited by triplet backticks, each on a separate line, in the format: subject | predicate | object. " {triplets} " Generate a coherent piece of text that contains all of the information in the triplets. Only use information from the provided triplets. After you finish writing the piece of text, write triplet dollar signs (i.e.: $$$).

F Case study

Our model's superior performance is attributed to its ability to better align generated knowledge graph sequences with the target text, as demonstrated in Table 17. This alignment is achieved through the sequence reordering task, which guides the model to produce sequences that closely match the intended narrative structure, such as reordering from (1)→(2)→(3) to (3)→(2)→(1). This task also enables accurate annotation of sequence numbers within the generated text, enhancing coherence and fidelity.

In contrast, the outputs from the model without fine-tuning fail in both aspects, and their sequence annotations are disordered. Furthermore, these outputs sometimes include hallucinated information. In the first example, the second and third triplets do not exist, yet the model generates two unrelated and even hallucinated triplets, thus introducing errors into the generated text.

G Three Curriculum Algorithms

1) **One-pass algorithm** (Bengio et al., 2009): The training data D is sorted by graph difficulty and distributed into k number of buckets. We train the model from the simplest buckets to the most complex.

2) **Baby-steps curriculum** (Spitkovsky et al., 2010): which first distributes the sorted data into buckets (or shards/bins) from easy to hard and starts training with the easiest bucket. After a fixed number of training epochs or convergence, the next bucket is merged into the training subset. Finally, after all the buckets are merged and used, we also fine-tune on it once.

3) **Annealing scheduler** proposed by Xu et al. (2020): as with (1) and (2), we start training from the easiest bucket, but for the next training bucket, we randomly add 1/k examples from the current and previous bucket.

Input instruction and gold reference	**Sentence generated by LLaMA2-7b-chat**	**Sentence generated by our trained model**
Input: (Texas, LargestCity, Houston) ① **Reference:** Houston is the largest city in Texas	**Order:** (Texas, LargestCity, Houston) ①, (Texas, capital, Austin) ②, (Houston, LargestCity, Texas) ③ ✗ **Text:** In the state of Texas, the largest city is Houston, which is also the capital of Texas. ①, ②, ③ ✗	**Order:** (Texas, LargestCity, Houston) ①, **Text:** Houston is the largest city in Texas ①. ✓
Input: (Albert Jennings Fountain, DeathPlace, New Mexico Territory) ①, (Albert Jennings Fountain, BirthPlace, New York City) ②, (Albert Jennings Fountain, BirthPlace, Staten Island) ③ **Reference:** Albert Jennings Fountain was born in Staten Island, New York City and died in the New Mexico Territory.	**Order:** (Albert Jennings Fountain, DeathPlace, New Mexico Territory) ⓪, (Albert Jennings Fountain, BirthPlace, New York City) ①, (Albert Jennings Fountain, BirthPlace, Staten Island) ② **Text:** Albert Jennings Fountain was born in New York City ①, and later moved to New Mexico Territory ⓪ where he passed away. According to the knowledge triplets, he was born on Staten Island ② ✗	**Order:** (Albert Jennings Fountain, BirthPlace, Staten Island) ③ (Albert Jennings Fountain, BirthPlace, New York City) ②, (Albert Jennings Fountain, deathPlace, New Mexico Territory) ① **Text:** Albert Jennings Fountain, born in Staten Island ③, New York City ②, died in the New Mexico Territory ①. ✓

Table 17: Two sample texts generated by LLaMA2-7b-chat baseline and our trained model. The wrong outputs are marked red and the hallucinated outputs are marked purple. The first example shows that LLMs suffer less from hallucination after being tuned by the attribution sub-task. The second example shows that LLMs can better understand relations between triplets and generate more fluent text after unlocking the planning capability.

System prompt: Your task is to rewrite prompts to enhance the clarity and accuracy of attributions in the text for AI systems like ChatGPT and GPT-4. As a Prompt Rewriter specializing in knowledge graphs, you will transform a provided prompt by incorporating numbered markers. These markers correspond to specific knowledge triples, clearly indicating the source of each piece of information in the text.
Instructions for Rewriting Prompts:
1. **Identify Knowledge Triples**: Start by numbering each knowledge triple in the #Given Prompt#.
2. **Incorporate Markers into the Text**: Carefully insert these numbered markers into the appropriate places within the text of #Given Prompt#. Ensure each marker is placed where its corresponding knowledge is referenced or implied. You MUST insert each marker only ONCE to the text. That means that you should not repeat using the same numbered marker. Ensure you add all markers. That is, every numbered marker should be added to the text without being ignored.
3. **Maintain Original Text Integrity**: Add only the numbered markers to the text. Do not alter any other part of the #Given Prompt#.
4. **Ensure Accurate Attribution**: Place each marker as close as possible to the relevant piece of information, ensuring clear and correct attribution.
Example for Guidance:
#Given Prompt#:
knowledge triples: [["MacGyver (1985 TV series)", "tv", "American Broadcasting Company"], ["American Broadcasting Company", "founded", "1943"], ["American Broadcasting Company", "headquarters", "New York City"], ["New York City", "nickname", "The Big Apple"]]. Text: The distinguished 1980s television series "MacGyver" was transmitted by the American Broadcasting Company, known as a primary network established in The Big Apple, aka New York City, since its inception in 1943.
#Rewritten Prompt#:
Knowledge triples: (1)["MacGyver (1985 TV series)", "tv", "American Broadcasting Company"], (2)["American Broadcasting Company", "founded", "1943"], (3)["American Broadcasting Company", "headquarters", "New York City"], (4)["New York City", "nickname", "The Big Apple"]]. Text: The distinguished 1980s television series "MacGyver" was transmitted by the American Broadcasting Company (1), known as a primary network established in The Big Apple (4), aka New York City (3), since its inception in 1943 (2).
#Given Prompt#:
knowledge triples: [["Association football", "olympics", "Iker Muniain"], ["Athletic Bilbao", "player", "Iker Muniain"], ["Athletic Bilbao", "sports_team", "Association football"], ["Spain national football team", "player", "Iker Muniain"], ["Spain national football team", "sport", "Association football"]]. Text: is a Spanish professional Association football who plays for Athletic Bilbao, where Iker Muniain is Captain (association football), and the Spain national football team.
#Rewritten Prompt#:
Knowledge triples: (1)["Association football", "olympics", "Iker Muniain"], (2)["Athletic Bilbao", "player", "Iker Muniain"], (3)["Athletic Bilbao", "sports_team", "Association football"], (4)["Spain national football team", "player", "Iker Muniain"], (5)["Spain national football team", "sport", "Association football"]. Text: Iker Muniain is a Spanish professional Association football player who plays for Athletic Bilbao (1)(2)(3), where he is the Captain (association football), and the Spain national football team (4)(5).
Follow these guidelines to enhance the trustworthiness and accuracy of AI-generated content by making the sources of information transparent and precise.
User prompt: Think carefully and rewrite the prompt.
#Given Prompt#:
{input}
#Rewritten Prompt#:

Figure 8: Prompt for the parallel attribution annotation

System prompt: You are professional in knowledge graphs. Your task is to simplify and clarify text to improve AI understanding. Focus on clearly expressing the relationships in knowledge triples without adding extra information or concepts into the optimized text
Instructions:
Look at the #Knowledge Triple# and #Given Text#. Optimized the text by (1) deleting all concepts that are in the #Given Text# but not in the #Knowledge Triple# (2) making sure that all entities and relations in the #Knowledge Triple# are included in the text. It should clearly represent the entities and their relationship as stated in the triple.
Example:
#Knowledge Triple#:
['Doug Fenske', 'institution', 'Rich South High School']
#Given Text#:
Doug Fenske enrolled at Rich South High School in Richton Park, IL, and was quickly promoted to the upperclass jazz ensemble, playing second tenor sax, all while performing with Rich South High School.
#Optimized Text#:
Doug Fenske is affiliated with Rich South High School.
User prompt: #Knowledge Triple#:
{input triple}
#Given Text#:
{input text}
#Optimized Text#:

Figure 9: Prompt for the rewrite and annotation

System Prompt: As professional in knowledge graph and prompt engineering, your task is to increase the complexity of prompts for AI models like GPT, while keeping them understandable for humans. This involves adding a new knowledge triple with the prefix $$ to the existing knowledge triples list in the #Given Prompt#, using an entity from the #Entity Candidates# list. The new triple must logically connect with the existing ones and the revised text should include this new triple without introducing unrelated concepts with the triples in the knowledge triple lists. Here's how to proceed:
1. **Select from #Entity Candidates#**: Choose an entity from the list like "James Pain" or "1700" for the new triple.
2. **Review the #Given Prompt#**: Understand the existing knowledge triples. For instance, the sample prompt relates to "Adare Manor".
3. **Form a New Knowledge Triple**: In the knowledge triple list, create a triple that includes an entity from the #Entity Candidates# and integrates well with the existing ones. Add "$$" before the created triple (e.g., "$$["James Pain", "birth year", "1779"]").
4. **Integrate the New Triple into the Prompt**: Seamlessly add this new triple into the existing text, keeping the narrative coherent. Ensure that all elements of the given knowledge triples are included without introducing unrelated entities with the knowledge triple lists.
5. **Limit Word Addition**: Add only 10 to 20 words to the original prompt.
6. **Keep It Coherent and Clear**: Despite increased complexity, the prompt should be understandable.
Example following these guidelines:
#Entity Candidates#
{"James Pain", "1700", "1862"}
#Given Prompt# Knowledge triples: [["Adare Manor", "architect", "James Pain"], ["Adare Manor", "building start date", "1700"], ["Adare Manor", "completion date", "1862"]]. Text: James Pain was the architect of Adare Manor, which started construction in 1700 and was completed in 1862.
#Rewritten Prompt#
Knowledge triples: [["Adare Manor", "architect", "James Pain"], ["Adare Manor", "building start date", "1700"], ["Adare Manor", "completion date", "1862"], $$["James Pain", "birth year", "1779"]]. Text: James Pain, born in 1779, architected Adare Manor, initiated in 1700 and completed in 1862.
User Prompt: Rewrite and Do NOT forget to add "$$" token before your added triple in the list of Knowledge triples.
#Entity Candidates#
{entities}
#Given Prompt#:
{inputs}
#Rewritten Prompt#:

Figure 10: Prompt for the sequential generation, where the {entities} and {inputs} are the given input

System prompt:
Your task is to enhance the clarity and coherence of prompts for AI systems like ChatGPT and GPT-4. As a Prompt Enhancer with a focus on knowledge organization, your role is to resequence knowledge triples in a list to improve their transformation into comprehensible text. This reordering should be based on the context of the provided reference text. You MUST NOT change or rewrite any provided knowledge triples.
Guidelines for Prompt Enhancement:
You are required to provide an optimized sequence for the given prompt. Essentially, you need to rearrange the knowledge triples to align more effectively with the reference text.
llustrative Example:
#Reference Text#:
The iconic 1980s television series "MacGyver" was broadcast by the American Broadcasting Compan a major network rooted in New York City, known as The Big Apple, since its founding in 1943.
#Original Prompt#:
Knowledge triples: [['MacGyver (1985 TV series)', 'tv', 'American Broadcasting Company'], ['American Broadcasting Company', 'founded', '1943'], ['American Broadcasting Company', 'headquarters', 'New York City'], ['New York City', 'nickname', 'The Big Apple']].
#Enhanced Prompt#:
Knowledge triples: [['MacGyver (1985 TV series)', 'tv', 'American Broadcasting Company'], ['American Broadcasting Company', 'headquarters', 'New York City'], ['New York City', 'nickname', 'The Big Apple'], ['American Broadcasting Company', 'founded', '1943']].
User prompt: Think carefully and rewrite the prompt.
#Reference Text#:
{ref_text}
#Given Prompt#:
Knowledge triples: {triple}
#Enhanced Prompt#:

Figure 11: Prompt for the linearization annotation