KG-Adapter: Enabling Knowledge Graph Integration in Large Language Models through Parameter-Efficient Fine-Tuning

Shiyu Tian[1], Yangyang Luo[1], Tianze Xu[1], Caixia Yuan[1], Huixing Jiang[2,*], Wei Chen[2], Xiaojie Wang [1,*]
1. Beijing University of Posts and Telecommunications
2. LI Auto Inc.
{tiansy, luoyangyang, xtzorz, yuancx, xjwang}@bupt.edu.cn
{jianghuixing, chenwei10}@lixiang.com

Abstract

Although large language models (LLMs) show remarkable capabilities and generalizability across various tasks, they are criticized for lack of expertise. One promising solution is to combine knowledge graphs (KGs) with LLMs, and recent studies focus on integrating KGs into LLMs through prompt-based methods. However, these approaches fail to use the structural information of the KGs, suffer from the problem of knowledge conflict, and over-reliance on super LLMs. To address these challenges, we propose **KG-Adapter**, a parameter-level KG integration method based on parameter-efficient fine-tuning (PEFT). Specifically, we introduce a novel adapter structure designed for decoder-only LLMs, which can encode KGs from both node-centered and relation-centered perspectives, and then perform joint reasoning with LLMs to generate responses end-to-end. Experiments with diverse models on four datasets for two different tasks all demonstrate significant improvements. With only 28M parameters trained, we make the 7B-parameter LLM outperform the previous full-parameter fine-tuned state-of-the-art method and comparable to the prompt-based ChatGPT methods.

1 Introduction

Large language models (LLMs), such as Llama (Touvron et al., 2023), Mistral (Jiang et al., 2023a), ChatGPT and GPT4 (OpenAI et al., 2023), exhibit remarkable abilities in multiple natural language processing (NLP) tasks (Bang et al., 2023; Ye et al., 2023), and are even considered as "the sparks of Artificial General Intelligence" (Bubeck et al., 2023). However, LLMs are not the elixir nowadays, they are argued for lack of up-to-date knowledge (Peng et al., 2023) and domain-specific expertise (Schick et al., 2023); the problem of hallucination (Ji et al., 2023; Li et al., 2023a); and the lack of interpretability (Zhang et al., 2023).

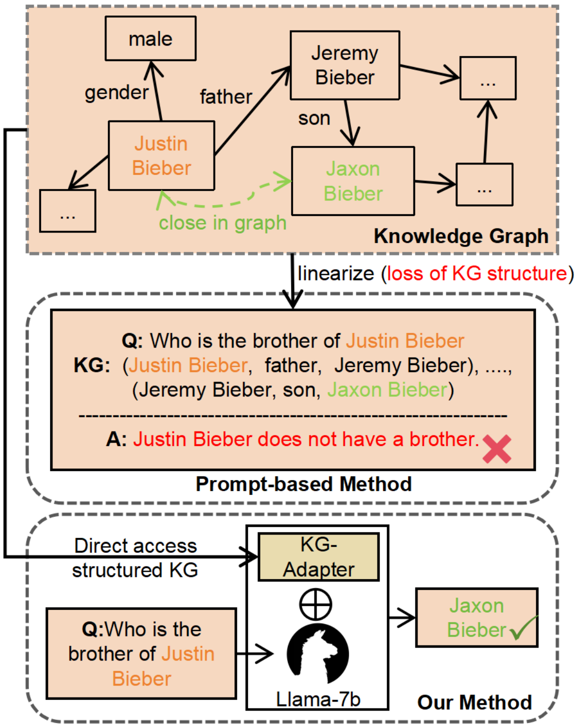

male
gender
father
Jeremy Bieber
son
...
Justin Bieber
Jaxon Bieber
...
...
close in graph
Knowledge Graph
linearize (loss of KG structure)
Q: Who is the brother of Justin Bieber
KG: (Justin Bieber, father, Jeremy Bieber), ..., (Jeremy Bieber, son, Jaxon Bieber)
A: Justin Bieber does not have a brother.
Prompt-based Method
Direct access structured KG
KG-Adapter
⊕
Q:Who is the brother of Justin Bieber
Llama-7b
Jaxon Bieber ✓
Our Method

Figure 1: An example to show the difference between the prompt-based method and our KG-Adapter.

To address the above issues, a feasible approach is to combine knowledge graphs (KGs) with LLMs (Pan et al., 2023). KGs explicitly contain massive accurate factual knowledge and domain expertise in a structured format (Ji et al., 2021; Abu-Salih, 2021). They are easy to modify, can update new knowledge readily (Mitchell et al., 2018), and provide human-readable reasoning paths (Zhang et al., 2021).

Recent works (Jiang et al., 2023b; Baek et al., 2023a; Wang et al., 2023a; Li et al., 2023b; Sun et al., 2023) focus on combining KGs with LLMs through prompt-based methods or LLMs-as-Agent (Xi et al., 2023) approach. These methods first

[*] Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2024, pages 3813–3828
August 11-16, 2024 ©2024 Association for Computational Linguistics

extract triplets from KGs by using LLMs through well-designed prompts (or other retrieval models), then linearize these triplets to textual form and concatenate them into the input. Although these methods achieve success, they have flaws (Figure 1 as an example): (1) LLMs cannot process structured KGs directly and the linearization may lose some underlying information (Li et al., 2023c; Guo et al., 2023; Chen et al., 2023); (2) Incorporating KGs via prompt-based methods suffers from the Knowledge Conflict problem (Longpre et al., 2021; Zhou et al., 2023), namely, the LLMs may completely ignore the knowledge provided in the context; (3) Over-reliance on super LLMs (e.g., ChatGPT), which makes these methods achieve limited improvement on small-scale LLMs.

Different from the above methods, we endeavor to integrate structured KGs directly into the LLMs at the parameter level to solve these problems. While some early works (Lin et al., 2019a; Yasunaga et al., 2021; Liu et al., 2020; Sun et al., 2020) have also attempted to inject KGs into pre-trained models at the parameter level, these approaches usually need pre-training or full-parameter fine-tuning, requiring inaccessible KG-text-aligned data and high training costs, or are based on encoder architectures, which are incompatible with current LLMs.

Specifically, inspired by the parameter-efficient fine-tuning (PEFT) (Lialin et al., 2023), we propose a novel adapter structure, named **KG-Adapter**, which contains a Sub-word to Entity Hybrid Initialization (SEHI) and multiple KG-Adapter layers, enabling LLMs to access structured KGs directly. The SEHI initializes the KG representations by fusing the entity-level KG representations (from pre-trained KG embeddings) with the sub-word-level KG representations (from the word embedding of LLM). Between each LLM layer, we insert a KG-Adapter layer that encodes KG from both node-centric and relation-centric perspectives and performs joint reasoning via bi-directional cross-attention to inject KG information into the LLM.

Experimental results demonstrate the effectiveness of our approach on four datasets across two tasks. With only 28M trainable parameters, we enable the 7B model to outperform the full-parameter fine-tuned SOTA methods and achieve comparable performance to the prompt-based ChatGPT methods.

Our contributions are summarized below:

- For the first time, we propose a PEFT-based method that integrates KGs directly into LLMs with just 28M trainable parameters and can be trained on a single A40 GPU within a few hours.

- We propose a novel adapter structure, KG-Adapter, which is compatible with the decoder-only LLMs and can encode KG from both node-centric and relation-centric perspectives to fully utilize the structure information.

- The experimental results demonstrate our approach is applicable to LLMs of different types and sizes, achieving significant improvements across four datasets, allowing the 7B-parameter model to outperform prior SOTA methods of full-parameter fine-tuning and is comparable to the prompt-based ChatGPT methods.

2 Related Work

2.1 Integration of KG by Prompt

Recent LLM-related studies are almost based on the RAG framework, where relevant KG subgraphs are obtained through retrieval methods, then linearized and inserted into the input context directly with well-designed prompts. CoK (Li et al., 2023b) retrieves relevant knowledge from KG after dividing the original question into multiple subquestions through a question generation model and uses the retrieval results to guide the answer generation. KAPING (Baek et al., 2023a) retrieves facts relevant to the question from the KG based on semantic similarity and fills in a special prompt as the input of LLMs. RoG (Luo et al., 2023) proposes a planning-retrieval-reasoning framework, which first generates relation paths grounded by KGs and uses them to retrieve valid reasoning paths from the KGs for LLMs to conduct reasoning. StructGPT (Jiang et al., 2023b) adopts a tool-learning approach to design an iterative reading-then-reasoning framework, which contains specialized interfaces that LLMs can call to access structured data and acquire valuable knowledge to help generate answers. However, these prompt-based methods suffer from the problems of loss of structural information, knowledge conflicts, and over-reliance on super LLMs.

2.2 Integration of KG by Pre-training

Some previous studies incorporate KG knowledge through specialized pre-training tasks. For example, ERNIE (Zhang et al., 2019) designs a token-entity alignment mask task, WKLM(Xiong et al., 2019) introduces an entity replacement discrimination task, GLM (Shen et al., 2020) proposes a KG-guided entity masking scheme to conduct the entity-level masked language modeling task, and Deterministic-LLM (Li et al., 2022) focuses on deterministic knowledge thus masks only deterministic entities, and additionally introduces the clue contrastive learning and the clue classification tasks. These pre-training tasks require large amounts of KG-text-aligned data and long training time, which are prohibitively expensive and impractical for LLMs.

2.3 Integration of KG by Fine-tuning

Before the era of LLMs, there was a series of studies (Lin et al., 2019a; Sun et al., 2021; Yasunaga et al., 2021; Zhang et al., 2022b; Wang et al., 2023d; Park et al., 2023) focusing on combining pre-trained models with KGs. These methods usually design a special KG encoder to integrate the KG information and perform full-parameter fine-tuning on specific tasks. However, there are two long-standing problems of heterogeneous representation (Lin et al., 2019a; Yasunaga et al., 2021; Sun et al., 2021; Zhang et al., 2022b) and knowledge noise (Liu et al., 2020; Sun et al., 2020) for this type of methods, which limit their effectiveness. Furthermore, these methods are incompatible with current LLMs because they rely on the encoder architecture and the full-parameter fine-tuning is also costly for LLMs. For this reason, our KG-Adapter is designed for decoder-only LLMs and is based on PEFT to reduce costs.

3 Methodology: KG-Adapter

3.1 Overview

To incorporate KGs into LLM at the parameter level, we first have to address the problem of heterogeneous representations, i.e., the pre-trained KG embeddings are very different from the word embeddings of the LLM since they exist in two different vector spaces. Thus, we propose the Sub-word to Entity Hybrid Initialization (SEHI) module to bridge the gap. Then, to fully use the information in KG, we design the KG-Adapter layer to encode KG from two perspectives and perform joint reasoning. Finally, we insert the KG-Adapter layer between each of the LLM blocks, ensuring that the internal structure and parameters of the original LLM are not modified.

As shown in Figure 2, we first fuse the entity-level KG representations (from the pre-trained KG embedding) with the sub-word-level KG representations (from the word embedding of LLM) by the SEHI and obtain the final KG representations (3.2). Then, the KG representations will be input into the KG-Adapter layer, where the KG representations will be updated in a node-centric way through the graph neural network (GNN) (3.3.1). After that, we reconstruct the KG representations into a relation-centric form by MLP_{trip} (3.3.2) and fuse them with the textual representations through bi-directional cross-attention to achieve joint reasoning (3.3.3). Finally, the updated textual representations will be passed back to the LLM, while the updated KG representations will be passed to the next layer.

3.2 Sub-word to Entity Hybrid Initialization

Since the KG representations and the textual representations exist in two different vector spaces, this gap will cause the heterogeneous representation problem. To alleviate that, we propose the SEHI that fuses the two types of representations at the initialization stage.

In pre-trained KG representations, nodes and edges are all at the entity level. For example, *"knowledge graph"* is an entity (multi-word), and it consists of two words *"knowledge"* and *"graph"*. In contrast, word embedding is at the sub-word level since LLMs commonly use BPE (Shibata et al., 1999) or SentencePiece (Kudo and Richardson, 2018) encoding.

A multi-word (i.e., nodes and edges in KG) contains multiple words and a word contains multiple sub-words. To align them, we first input each node into the pre-trained word embedding of LLM to get their sub-word level representations:

$$\{h_n^{sub_1}, h_n^{sub_2}, ..., h_n^{sub_k}\} = Emb(n) \qquad (1)$$

where Emb is the pre-trained word embedding in LLM, n is a node in KG, $h_n^{sub_i} \in \mathbb{R}^{d'}$ is a sub-word level hidden vector of node n and d' is the hidden size of the word embedding. We then sum these hidden vectors to obtain the multi-word level representations and pass them through a MLP to downscale the hidden size to the same as the pre-

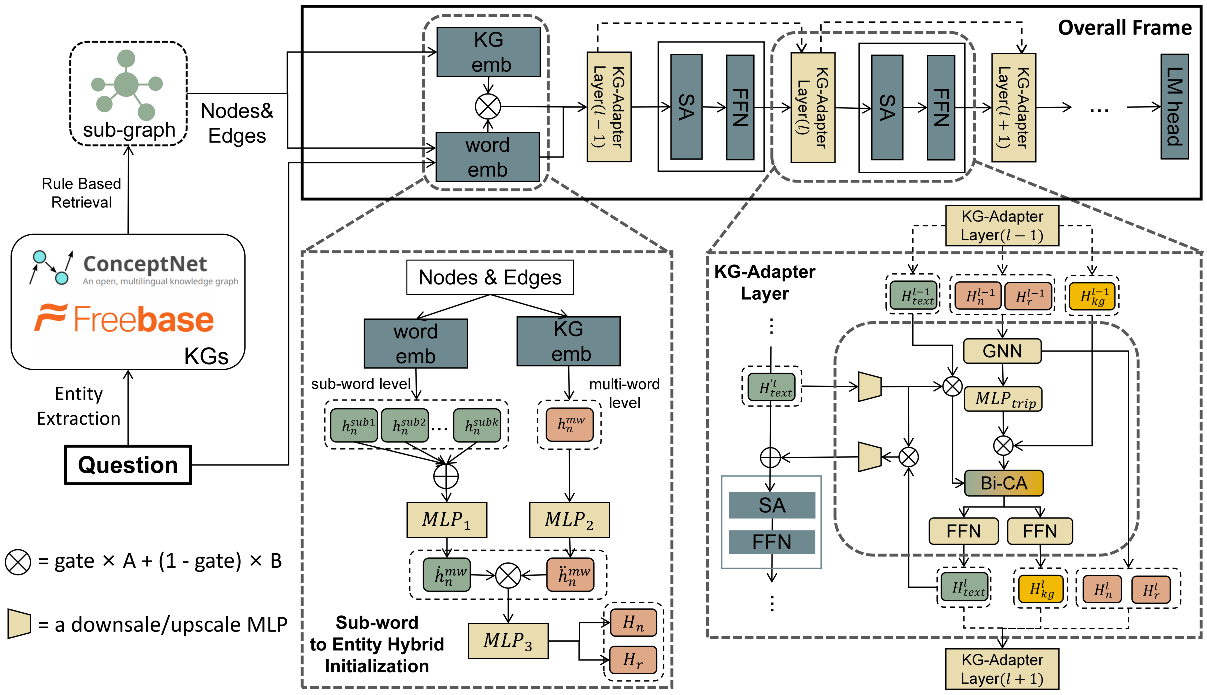


Figure 2: KG-Adapter Structure. The upper part is the overall frame, and the lower part contains the internal structure of the Sub-word to Entity Hybrid Initialization and the KG-Adapter Layer. The modules with blue color are pre-trained and will be frozen during fine-tuning, and the modules with yellow color are trainable.

trained KG embedding (from 4096 to 1024):

$$\dot{h}_n^{mw} = MLP_1\left(\sum_{i=1}^{k} h_n^{sub_i}\right) \tag{2}$$

where $\dot{h}_n^{mw} \in \mathbb{R}^{d''}$ is the multi-word level representations from LLM word embedding and d'' is the hidden size of KG embedding. Then, we update the pre-trained KG representation h_n^{mw} with another MLP:

$$\ddot{h}_n^{mw} = MLP_2(h_n^{mw}) \tag{3}$$

and fused $\dot{h}_n^{mw}$ with $\ddot{h}_n^{mw}$ by gated sum and pass through a MLP to further downscale the hidden size (from 1024 to 64):

$$h_n = MLP_3(\sigma(\dot{h}_n^{mw}) + (1-\sigma)\ddot{h}_n^{mw}) \tag{4}$$

where σ is a learnable factor, $h_n \in \mathbb{R}^d$ is the final representation of node n and d is the hidden size of KG-Adapter layer. Using the same way, we get the hybrid initialization representations for all nodes and edges in KG:

$$H_n = \{h_{n_1}, ..., h_{n_N}\} \tag{5}$$

$$H_r = \{h_{r_1}, ..., h_{r_M}\} \tag{6}$$

where N and M are the number of nodes and edges in KG, H_n is the representation of all nodes and H_r is the representation of all edges in KG.

These initial representations contain information from both LLM and KG, thus alleviating the heterogeneous representation problem and making it more understandable for LLM.

3.3 KG-Adapter Layer

3.3.1 GNN layer

GNNs learn node features through a node-centric message-passing scheme. Thus, we first use RGAT (Wang et al., 2020) to update the KG representations, where each node is updated with the information from its neighboring nodes and corresponding edges, to obtain the node-centric KG representations:

$$H_n^l = \{h_{n_1}^l, ..., h_{n_N}^l\} = RGAT(H_n^{l-1}, H_r^{l-1}) \tag{7}$$

where $H_n^{l-1} = \{h_{n_1}^{l-1}, ..., h_{n_N}^{l-1}\}$ and $H_r^{l-1} = \{h_{r_1}^{l-1}, ..., h_{r_M}^{l-1}\}$ are node and edge representations from previous layer, $h_{n_i}^l$ is a updated node representation, computed by the following equations:

$$\begin{aligned} \alpha_{ji} &= Softmax\left(\frac{(h_{n_i}^{l-1}W_q)(h_{n_j}^{l-1}W_k + h_{r_{ji}}^{l-1})^T}{\sqrt{d}}\right) \\ \hat{h}_{n_i}^{l-1} &= \sum_{j \in neighbor(i)} \alpha_{ji}(h_{n_j}^{l-1}W_v + h_{r_{ji}}^{l-1}) \\ h_{n_i}^l &= LN(h_{n_i}^{l-1} + \hat{h}_{n_i}^{l-1}W_o) \end{aligned} \tag{8}$$

where W_q, W_k, W_v, W_o are trainable linear projections, d is the hidden size of KG-Adapter layer, $neighbor(i)$ is all neighbor nodes of node i, LN is layer norm.

3.3.2 MLP_{trip}

Previous research (Park et al., 2023) found that node-centered updating using only GNNs is insufficient to encode KG, and they argued that a relation-centered representation is more suitable. Inspired by that, we introduce a triple projection MLP - MLP_{trip} - which merges the node and edge representations into relation-centered triple representations. Since each triple consists of two nodes and an edge, we first concatenate them together and fuse their representations via MLP_{trip}:

$$h_{t_{ij}} = [h_{n_i}; h_{r_{ij}}; h_{n_j}]$$
$$\hat{h}_{t_{ij}} = MLP_{trip}(h_{t_{ij}}) \tag{9}$$

where $h_{t_{ij}} \in \mathbb{R}^{3d}$ is a triple representation concatenated of nodes and edge representations, $\hat{h}_{t_{ij}} \in \mathbb{R}^d$ is the fused triple representation obtained from MLP_{trip} and ";" is the concatenation operation of vectors.

Now, we have the relation-centered triple representations that contain information about the KG structure.

3.3.3 Joint Reasoning Module

The next step is to perform joint reasoning between the representations of KG and text to integrate these two types of information thoroughly. We first align the hidden size of the textual representations H'_{text} from LLM to the same as KG representations through a downscale MLP (from 4096 to 64):

$$H_{text} = MLP_{down}(H'_{text}) \tag{10}$$

and concatenate all triple representations into a matrix as the new KG representations:

$$H_{kg} = [\hat{h}_{t_1}; ...; \hat{h}_{t_M}] \tag{11}$$

then use a bi-directional cross-attention to conduct joint reasoning:

$$Q_{kg}, K_{kg}, V_{kg} = H_{kg}\tilde{W}_Q, H_{kg}\tilde{W}_K, H_{kg}\tilde{W}_V$$
$$Q_{text}, K_{text}, V_{text} = H_{text}\tilde{W}_Q, H_{text}\tilde{W}_K, H_{text}\tilde{W}_V$$
$$\hat{H}_{kg} = Softmax(Q_{kg}K_{text}^T/\sqrt{d})V_{text}$$
$$\hat{H}_{text} = Softmax(Q_{text}K_{kg}^T/\sqrt{d})V_{kg} \tag{12}$$

where $\tilde{W}_Q, \tilde{W}_K, \tilde{W}_V$ are three attention matrices, $\hat{H}_{kg}$ is the KG representations updated by textual information, $\hat{H}_{text}$ is the text representations updated by KG information. We then upscale the hidden size of $\hat{H}_{text}$ (from 64 to 4096) and sum it with H'_{text}:

$$\hat{H}'_{text} = MLP_{up}(\hat{H}_{text}) + H'_{text} \tag{13}$$

where $\hat{H}'_{text}$ is the final textual representation to be passed into the LLM layer. Note that we omit all residuals (the $\otimes$ in Figure 2) in equations, which fuses the two values by gated summation.

4 Experimental Setup

4.1 Datasets

Datasets used. We conduct experiments on two types of tasks: Multiple-Choice Question Answering (MCQA) and Knowledge Graph-based Question Answering (KGQA) (see A.1 for the formal definitions). For the MCQA task, we evaluate our method on two commonly used datasets, OpenBookQA (OBQA) (Mihaylov et al., 2018) and CommonsenseQA (CSQA) (Saha et al., 2018). For the KGQA task, we use WebQuestionsSP (WQSP) (Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018) datasets.

Datasets split. For CSQA, we follow Lin et al. (2019b) to split the data into in-house development (IHdev) and in-house test (IHtest) sets. For other datasets, we use their official split. Dataset statistics are in A.2.

KG used and pre-processing. For OBQA and CSQA, we use *ConceptNet* (Speer et al., 2017) as the source KG and directly use the processed KG subgraphs from Park et al. (2023), where each KG subgraph is retrieved from the entire KG based on the entities in the question sentence. For WQSP and CWQ, we use *FreeBase* (Bollacker et al., 2008) as the source KG and use the processed data by Xie et al. (2022b). See A.3 for more details about the KGs pre-processing.

4.2 Implementation Details

For training, we choose Llama2-7B-base [1] and Zephyr-7B-alpha [2] as the base models. Different instructions and templates are used for various tasks and base models (See A.4). We perform up to 10 epochs **instruction-tuning** with early-stop, use

[1] https://huggingface.co/meta-llama/Llama-2-7b

[2] https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha

the **autoregressive objective** and calculate the loss on answer tokens only. For testing, we select the best model based on the performance of the development set and use **greedy decoding** to generate responses with a **zero-shot** context. We use one NVIDIA A40 48GB GPU for training and testing. For hyperparameters, refer to A.5.

4.3 Baselines

To comprehensively evaluate our method, we select a total of 26 methods from three categories for comparison, containing (1) KG+Full-Parameter Fine-Tune LMs, which design specialized KG encoders and combine them with pre-trained language models for full-parameter fine-tuning; (2) LLM Baselines, which are original LLMs after pre-training or supervised fine-tuning (SFT); (3) KG+Prompt Enhanced LLMs, which are enhanced LLMs with retrieval and prompt-based methods.

KG+Full-Parameter Fine-Tune LM. This type of approach usually designs specialized KG encoding modules and combines them with pre-trained language models for full-parameter fine-tuning. For OBQA and CSQA datasets, we choose GreaseLM (Zhang et al., 2022b), JointLK (Sun et al., 2021), GSC (Wang et al., 2021), DHLK (Wang et al., 2023d) and QAT (Park et al., 2023), where QAT and DHLK are the recent SOTA methods. For WQSP and CWQ, we compare with the embedding-based methods, including Embed-KGQA (Saxena et al., 2020), NSM (He et al., 2021), KGT5 (Saxena et al., 2022), and retrieval-augmented methods, including PullNet (Sun et al., 2019), SR (Zhang et al., 2022a), UniKGQA (Jiang et al., 2023d).

LLM Baselines. The original LLMs or the LLMs after SFT, including Llama2 (Touvron et al., 2023), Zephyr (a SFT version of Mistral) (Jiang et al., 2023a) , Flan-T5 (Chung et al., 2022), Alpaca (Taori et al., 2023), GPT-3 (Brown et al., 2020), ChatGPT and GPT4 (OpenAI et al., 2023).

KG+Prompt Enhanced LLMs. Includes prompt engineering and RAG methods for LLMs. For OBQA and CSQA, we compare with Few-shot (Brown et al., 2020), CoT (Wei et al., 2022), Auto-CoT (Zhang et al., 2022c), KSL (Feng et al., 2023), KAPING (Baek et al., 2023b) and CoK (Wang et al., 2023b). For WQSP and CWQ, we compare with KD-CoT (Wang et al., 2023c), StructGPT (Jiang et al., 2023c), KAPING (Baek et al., 2023b).

4.4 Evaluation Metrics

For the MCQA task, we follow previous work (Park et al., 2023; Wang et al., 2023d) using *Accuracy* to assess the correctness of the prediction. For the KGQA task, we follow the same way as Jiang et al. 2023c to calculate *Hits@1* which assesses whether the top-1 predicted answer is correct or not, since we use the greedy decoding strategy that generates only one answer.

5 Results and Analysis

5.1 Main Results

MCQA task. In Table 1, we report our KG-Adapter and baseline performances in the MCQA task, including OBQA and CSQA datasets. Our KG-Adapter exhibits an average improvement of 40.8 and 47.5 points over the base models Llama2-7B and Zephyr-7B on two datasets. With only 27.9M parameters trained, our KG-Adapter outperforms all LLM baselines and KG+Full-Finetune methods, including previous SOTA method QAT, demonstrating the effectiveness of our method and the potential of LLM with well-designed PEFT architectures for KG-related tasks. In comparison to KG+Prompt-enhanced LLMs, KG-Adapter outperforms all methods even though their base model (ChatGPT) is much stronger than ours (Zephyr-7B). Furthermore, these methods over-rely on super LLMs and achieve limited enhancements on small-scale LLMs, whereas our method is effective on the 7B models as well.

KGQA task. Unlike the MCQA task, the KGQA task is open-ended QA with no options, so the KG subgraphs may contain extensive invalid information unrelated to the correct answers, to address that, many previous methods use a two-stage retrieval-then-generation pipeline. However, retrieval is not the focus of our study, the KG subgraphs we used contain more noise than the work focus on retrieval (marked with † in Table 2), putting our approach at a disadvantage in comparisons.

We present the results of both WQSP and CWQ datasets in Table 2. Compared to our base models Llama2-7B and Zephyr-7B, our method achieves an average of 32.8 and 16.7 points improvement over the two datasets. Compared with KG+Full-Finetune methods, KG-Adapter outperforms all methods that don't use retrieval and achieves comparable results to methods using retrieval. Except for UniKGQA, which first pre-trains a model and

Categories	Methods	OBQA(acc)	CSQA(acc)		#base model	#FTP
		test	dev	ih_test		
KG+Full-Finetune LMs	GreaseLM (2022b)	84.8	78.5	74.2	AristoRoBERTa(355M)	>355M
	JointLK (2021)	84.9	77.9	74.4	AristoRoBERTa(355M)	>355M
	GSC (2021)	86.7	79.1	74.5	AristoRoBERTa(355M)	>355M
	DHLK (2023d)	86	79.4	74.7	AristoRoBERTa(355M)	>355M
	QAT (2023)	86.9	79.5	75.4	AristoRoBERTa(355M)	>355M
LLM Baselines	Llama2-7B* (2023)	55.6	29.6	28.5	Llama2(7B)	0
	Zephyr-7B* (2023a)	41.2	39.2	36.3	Zephyr(7B)	0
	Llama2-70B (2023)	60.2	—	—	Llama2(70B)	0
	GPT-3 (2020)	48.2	53.9	52	GPT3(175B)	0
	ChatGPT (2023)	74.8	73.5	71	ChatGPT(>100B)	0
	GPT4 (2023)	91.0	77.6	79	GPT4(>100B)	0
KG+Prompt Enhanced LLMs	Few-Shot (2020)	76.6	79.5	—	GPT3(175B)	0
	CoT (2022)	73	73.5	—	GPT3(175B)	0
	Auto-CoT (2022c)	—	74.4	—	GPT3(175B)	0
	KSL+GPT3.5 (2023)	81.6	79.6	—	ChatGPT(>100B)	0
	KSL+Lama2-7B (2023)	32.2	26.3	—	Llama2(7B)	0
	KAPING (2023b)	60	—	—	FLAN-T5 xxlarge(11B)	0
	CoK (2023b)	74.8	77.3	—	GPT3(175B)	0
	GNP (2024)	87.2	—	—	FLAN-T5 xxlarge(11B)	>0
	GNP+LoRa (2024)	89.6	—	—	FLAN-T5 xxlarge(11B)	>0
KG-Adapter (Our)	Our+Llama2-7B	89.2	78.1	76.6	Llama2(7B)	27.9M
	Our+Zephyr-7B	**93.2**	**79.6**	**79.3**	Zephyr(7B)	27.9M

Table 1: Main results of MCQA task. #FTP is the number of trainable parameters of fine-tuning. $*$ denotes results reproduced with our prompt in zero-shot, all other results are from the original paper or other paper.

Categories	Methods	WQSP(hit@1)	CWQ(hit@1)	#base model	#FTP
		test	test		
KG+Full-Finetune LMs	EmbedKGQA (2020)	66.6	44.7	RoBERTa-base(125M)	>125M
	KGT5 (2022)	56.1	36.5	T5-small(60M)	>60M
	NSM (2021)	68.7	47.6	—	>0
	GraftNet†(2022)	66.4	36.8	—	>0
	PullNet†(2019)	68.1	45.9	—	>0
	Subgraph Retrieval†(2022*a*)	69.5	49.3	RoBERTa-base(125M)	>125M
	UniKGQA†(2023*d*)	**77.2**	51.2	RoBERTa-base(125M)	>375M
LLM Baselines	Llama2-7B* (2023)	31.4	8.3	Llama2(7B)	0
	Zephyr-7B* (2023a)	54.9	31.4	Zephyr(7B)	0
	Flan-T5-xl (2022)	31	14.7	Flan-T5-xl(3B)	0
	Alpaca-7B (2023)	51.8	27.4	Alpaca(7B)	0
	ChatGPT (2023)	66.8	39.9	ChatGPT(>100B)	0
KG+Prompt Enhanced LLMs	KD-CoT†(2023*c*)	68.6	**55.7**	ChatGPT(>100B)	0
	StructGPT†(2023*c*)	72.6	—	ChatGPT(>100B)	0
	KAPING-175B†(2023*b*)	73.9	—	GPT3(175B)	0
	KAPING-7B†(2023*b*)	60.4	—	GPT3(7B)	0
KG-Adapter (Our)	Our+Llama2-7B	65.9	47.8	Llama2(7B)	25.7M
	Our+Zephyr-7B	**68.7**	**51**	Zephyr(7B)	25.7M

Table 2: Main results of KGQA task. #FTP is the number of trainable parameters of fine-tuning. † denotes using retrieval. $*$ denotes results reproduced with our prompt in zero-shot, all other results are from the original paper or other paper.

then fine-tunes it to get both a retrieval model and a generation model for retrieval-then-generation, whereas our method only performs end-to-end training with 25.7M learnable parameters and no pre-training or retrieval is used. For the LLM-based approaches, our KG-Adapter outperforms all LLM baselines and achieves a close performance to the KG+Prompt-enhanced LLMs, even though all of them have stronger base models (e.g., ChatGPT) than ours and all use retrieval.

5.2 Ablation Studies

Impact of KG-Adapter components. We perform ablation experiments to analyze the effec-

Method	Base Model	
	Zephyr-7B	Llama2-7B
KG-Adapter	**93.2**	**89.2**
w/o SEHI	92.2	88.8
w/o GNN	91.2	88.2
w/o MLP_{trip}	90.6	88.4
w/o KG	90.8	88.8
w/ LoRa	91.6	88.4

Table 3: Ablation study on OBQA(acc) test set.

tiveness of each module, including (1) **w/o SEHI**, where we remove the SEHI and use the LLM's word embedding as the KG representations; (2) **w/o GNN**, where we remove the GNN which updates the KG representations from node-centered; (3) **w/o** MLP_{trip}, where we remove the MLP_{trip} which is used to get the relation-centered KG representations.

As shown in Table 3, removing any module from KG-Adapter hurts performance, illustrating the positive effects of each module. KG-Adapter w/o MLP_{trip} and w/o GNN cause the greatest decrease in performance on Zephyr-7B and Llama2-7B, respectively, suggesting that both node-centered and relation-centered KG encoding helps the models to utilize KG information better. When w/o SEHI, the decrease in performance is minimal, suggesting that the word embedding of LLM contains most of the information in the KG embedding, with the size and pre-trained corpus of LLM growing, training a KG embedding may be unnecessary. In addition, we observe that the effectiveness of different modules varies across different base models, suggesting that the kind of base model affects the effectiveness of our approach because they have different capabilities and parameter knowledge and therefore may have different mechanisms for understanding KG.

Impact of using KG. To test the effect of integrating KG, we also experiment with (1) **w/o KG**, where we input an empty KG containing two PAD nodes and a PAD edge; (2) **w/ LoRa**, where we train a LoRa (Hu et al., 2021) using the same number of learnable parameters and data as KG-Adapter.

The results are shown in Table 3. Without using KG, our performance is similar to the w/ LoRa, suggesting that our method can also serve as a general PEFT approach beyond injecting KG. Compared to KG-Adapter (w/ KG), the use of KG leads to 1.6% ↑ on average across two models, indicating the effectiveness of injecting KG. Surprisingly, a simple LoRa can also yield impressive improvement, showing the potential of LLM with well-designed PEFT methods, since the LLMs may have acquired most of the knowledge in KG as parameterized knowledge during pre-training, and may generate correct answers without external knowledge due to their strong reasoning ability.

Impact of the number of KG-Adapter layers. We verify the effect of the number of KG-Adapter layers. As shown in Figure 3 (a), we find that increasing the number of KG-Adapter layers results in a consistent improvement. This can be attributed to the increase of trainable parameters and more stacked layers can help encode the KG better.

Impact of the hidden size of KG-Adapter. In Figure 3 (b), we examine different hidden sizes of KG-Adapter. We observe a notable improvement when the hidden size increases from 16 to 64. Instead, the effect decreases slightly as the hide size continues to increase to 128. This indicates that a hidden size of 64 is sufficient to represent the main information of KG and a larger hidden size leads to slower convergence during training and may contain more noise.

5.3 Analysis

Applicability to LLMs with different sizes. We evaluate the generalizability of our KG-Adapter by applying it to LLMs with different numbers of parameters. As shown in Figure 4 (a), our KG-Adapter improves the performance of models ranging from 3B to 13B in size, proving our KG-Adapter can apply to LLMs with different sizes and can inherit the capabilities of the base models that achieve better performance on a larger base model.

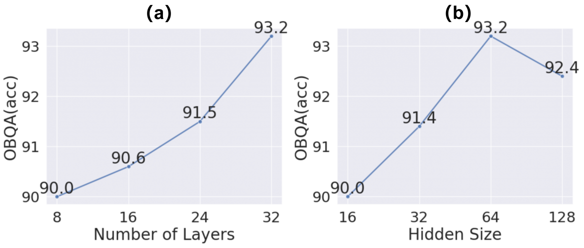


Figure 3: Impact of the number of KG-Adapter layers (a), and the hidden size of KG-Adapter (b).

Applicability to LLMs at different stages. To further assess the generalizability of KG-Adapter, we select Llama-base, Llama-chat, Mistral-base, and Mistral-inst as the base models to test whether

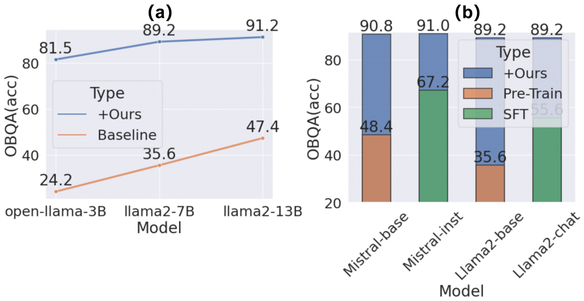
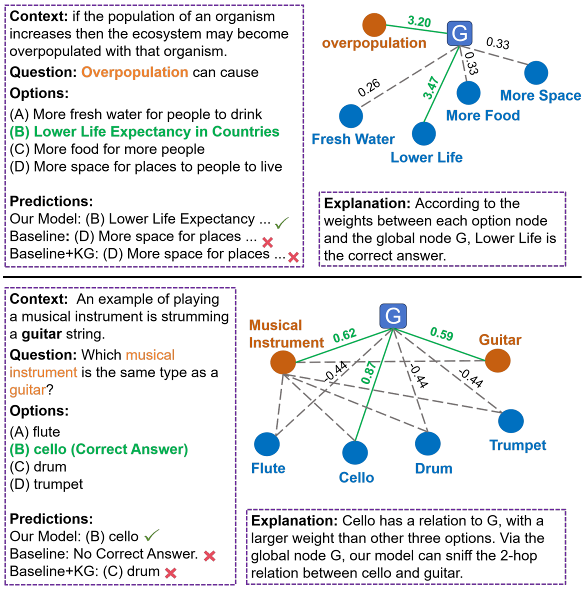


Figure 4: Applicability to LLMs with different sizes (a), and at different stages (b).

our approach can be applied to LLMs at different stages (after SFT or not). As illustrated in Figure 4 (b), our method demonstrates applicability to different types of LLMs, regardless of whether they have undergone SFT or not. Furthermore, experimental results show that our approach does not lead to catastrophic forgetting even when the SFT models are trained three times (i.e., pre-training, SFT, fine-tuning).



Context: if the population of an organism increases then the ecosystem may become overpopulated with that organism.
Question: Overpopulation can cause
Options:
(A) More fresh water for people to drink
(B) Lower Life Expectancy in Countries
(C) More food for more people
(D) More space for places to people to live

Predictions:
Our Model: (B) Lower Life Expectancy ... ✓
Baseline: (D) More space for places ... ✗
Baseline+KG: (D) More space for places ... ✗

Explanation: According to the weights between each option node and the global node G, Lower Life is the correct answer.

Context: An example of playing a musical instrument is strumming a **guitar** string.
Question: Which musical instrument is the same type as a guitar?
Options:
(A) flute
(B) cello (Correct Answer)
(C) drum
(D) trumpet

Predictions:
Our Model: (B) cello ✓
Baseline: No Correct Answer. ✗
Baseline+KG: (C) drum ✗

Explanation: Cello has a relation to G, with a larger weight than other three options. Via the global node G, our model can sniff the 2-hop relation between cello and guitar.

Figure 5: Case Study.

Case Study. We conduct a case study to analyze how our KG-Adapter leverages the structure information in KG and mitigates the knowledge conflict problem. We select some representative nodes and their corresponding edges from the whole KG subgraph for visualization, and the weights on edges are obtained by summing the attention weights in each KG adapter layer and performing normalization. We compare our KG-Adapter with two types of vanilla approaches: (1) baseline, which is the base model of our KG-Adapter, and (2) baseline+KG, in which the linearized KG is added in the context as model input.

As shown in the upper of Figure 5, our approach starts from the key entity *"Overpopulation"* in the question, and through 2-hop reasoning, successfully finds the correct entity *"lower life"*, which has the highest weight on edge. Conversely, the baseline methods generate an incorrect answer, regardless of the addition of KG, which illustrates incorporating KG through prompts may lead to knowledge conflicts and the lack of KG structure leads to underutilization of KG information. Similarly, in the figure below, our approach reaches the correct answer entity *"Cello"* through the path with the highest weight from the two question entities, showing the interpretability of our method. However, the baseline method response is "No Correct Answer" due to the lack of additional knowledge, and the baseline+KG gives an answer but is wrong, suggesting that simply inputting a linearized KG is difficult to understand by the model and loses structural information in the KG.

6 Conclusion

To enable LLMs to directly access structured KGs and overcome the limitations of current prompt-based approaches, i.e., the inability to utilize the structural information of KGs, the problem of knowledge conflicts, and the over-reliance on super LLMs. We propose a novel PEFT-based adapter structure, named KG-Adapter, which is a plug-and-play adapter module applicable to various LLMs and requires only 28M parameters to train. Benefiting from the internal GNN layer and MLP_{trip}, KG-Adapter can encode KGs from both node-centered and relation-centered perspectives to fully utilize the structure information in KGs. Experiments on four datasets across two tasks show that our KG-Adapter achieves superior performance over previous full-parameter fine-tuned SOTA methods and is comparable to the recent prompt-based ChatGPT methods.

Limitations

Due to the limitation of computational resources, our main experiments were performed only on the models with 7B parameters, and based on the experiments in Section 5.3, we believe that our method can achieve better performance on larger models (e.g., 70B). For the same reason, we do not compare with the full-parameter fine-tuned LLMs.

Many studies have indicated that a good KG retrieval method can effectively improve the quality of the final generated responses (as shown in Table 2, the methods with retrieval usually get better results). However, retrieval is not the focus of our work, so we do not use complex retrieval methods, which means that our approach can be further improved by combining with good retrieval methods.

We find that existing KGQA datasets are not entirely suitable for LLMs; they are either too easy for LLMs, which can generate answers well without relying on KG knowledge, or too dependent on retrieval, which makes the input KG contain much noise, making it hard to independently test the ability of LLMs to comprehend and utilize structured knowledge.

Ethical Considerations

Our study focuses on integrating KGs into LLMs through parameter-efficient fine-tuning, aiming to alleviate the issues of losing KG structure information in prompt-based methods. All models, datasets and KGs used in this paper are open-source and publicly available. Based on this, we believe that our research does not compromise data security or personal privacy. Furthermore, our study is beneficial in reducing the generation of false information by LLMs. We are confident that our research not only avoids causing harm to society but also contributes to societal well-being.

Acknowledgements

We would like to thank anonymous reviewers for their suggestions and comments sincerely. The work was partially supported by the National Natural Science Foundation of China (NSFC62076032).

References

Bilal Abu-Salih. 2021. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023a. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023b. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106, Toronto, Canada. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Chao Feng, Xinyu Zhang, and Zichu Fei. 2023. Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM* *international conference on web search and data mining*, pages 553–561.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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. 2023a. Mistral 7b.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023c. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023d. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023a. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Cheng-Jie Sun, Bingquan Liu, Zhenzhou Ji, Xin Jiang, and Qun Liu. 2022. Pre-training language models with deterministic factual knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11118–11131.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023b. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023c. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019a. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019b. Kagnet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839.

Biyang Liu, Huimin Yu, and Guodong Qi. 2022. Graftnet: Towards domain generalized stereo matching with a broad-spectrum and task-oriented feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13012–13021.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Jinyoung Park, Hyeong Kyu Choi, Juyeon Ko, Hyeonjin Park, Ji-Hoon Kim, Jisu Jeong, Kyungmin Kim, and Hyunwoo Kim. 2023. Relation-aware language-graph transformer for question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13457–13464.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Amrita Saha, Vardaan Pahuja, Mitesh Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Tao Shen, Yi Mao, Pengcheng He, Guodong Long, Adam Trischler, and Weizhu Chen. 2020. Exploiting structured knowledge in text via graph-guided representation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8980–8994.

Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. 1999. Byte pair encoding: A text compression scheme that accelerates pattern matching.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390.

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

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. *arXiv preprint arXiv:2010.00309*.

Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. 2021. Jointlk: Joint reasoning with language models and knowledge graphs for commonsense question answering. *arXiv preprint arXiv:2112.02732*.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jianing Wang, Qiushi Sun, Nuo Chen, Xiang Li, and Ming Gao. 2023a. Boosting language models reasoning with chain-of-knowledge prompting. *arXiv preprint arXiv:2306.06427*.

Jianing Wang, Qiushi Sun, Nuo Chen, Xiang Li, and Ming Gao. 2023b. Boosting language models reasoning with chain-of-knowledge prompting.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020. Relational graph attention network for aspect-based sentiment analysis. *arXiv preprint arXiv:2004.12362*.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023c. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Kuan Wang, Yuyu Zhang, Diyi Yang, Le Song, and Tao Qin. 2021. Gnn is a counter? revisiting gnn for question answering. In *International Conference on Learning Representations*.

Yujie Wang, Hu Zhang, Jiye Liang, and Ru Li. 2023d. Dynamic heterogeneous-graph reasoning with language models and knowledge representation learning for commonsense question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14048–14063.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022a. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022b. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, et al. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *arXiv preprint arXiv:2303.10420*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022a. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022b. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022c. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. Context-faithful prompting for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14544–14556, Singapore. Association for Computational Linguistics.

A Appendix

A.1 Preliminary

In this section, we introduce the definition of the knowledge graph and formally define the problem of multiple-choice question answering (MCQA) and knowledge graph-based question answering (KGQA).

Definition 1. Knowledge Graph. A knowledge graph is defined as a set of triples $\mathcal{G} = \{T_1, T_2, ..., T_n\}$, each T is a triple $T = (h, r, t)$, where h and t are the head entity and the tail entity, and r is a directed edge from h to t.

Problem 1. Multiple-Choice Question Answering. Given a question Q, a set of answer options $\mathcal{A} = \{a_1, a_2, ..., a_k\}$, and an optional context C depending on open-book or close-book, the task is to select the best answer. We additionally use knowledge graphs to provide external knowledge, therefore the task can be defined as $a^* = \arg\max_\theta f_\theta(Q, C, \mathcal{G}, \mathcal{A})$, where $a^* \in \mathcal{A}$ is the correct option and f_θ is the model with parameter θ.

Problem 2. Knowledge Graph-Based Question Answering. Given a question Q and a knowledge graph $\mathcal{G}$, the target is to generate the expected result to answer the question Q based on the knowledge from $\mathcal{G}$, which is defined as $a^* = \arg\max_\theta f_\theta(Q, \mathcal{G})$, where a^* is an open answer.

A.2 Statistics of Datasets

A.3 Detail of KG Processing

Subgraph Retrieval. In the MCQA and KGQA tasks, the general process is first to retrieve a sub-

Statistics	OBQA	CSQA	WQSP	CWQ
nodes	18.08±7.14	18.45±5.19	48.99±17.69	71.87±19.89
edges	58.33±31.89	70.57±26.74	39.89±10.65	57.50±6.79
train/dev/test	4957/500/500	8500/1221/1241	2673/309/1639	27639/703/2816

Table 4: Statistics of Datasets. Displays the mean and standard deviation of the number of nodes and edges, and the number of training/validation/test sets in each dataset.

graph related to the question from the full KG, and then input that subgraph to the model with the question. Specifically, the retrieval process is to (1) find all paths from each question entity to each option entity (if have) within n-hop; (2) then calculate the relevant scores of each node (entity) and question by a pre-trained LM, such as Sentence-BERT; (3) only retain the top-k scoring nodes and build a subgraph.

In our work, we do not focus on the retrieval process, so we directly use the retrieved subgraphs from previous works. Specifically, for the MCQA task, we use the processed KG data by Park et al. (2023), which provides a sub-graph of KG for each sample. It merges the edge types in *ConceptNet* from 31 to 17, and introduces two additional edges of global nodes "G → q" and "G → a", and finally, we add the reverse edge to each edge to obtain 38 edge types (relations). For the KGQA task, we use the processed KG-QA pair data by Xie et al. (2022a) that can be downloaded here [3], where contain 772 and 801 relation types for WQSP and CWQ datasets, respectively.

KG Embeddings. KG embeddings are the same as word embeddings which provide a vector representation for each node/entity in the KG. A simple way to get KG embeddings is first using templates to convert the triples into sentences and feed them into some LMs (such as BERT-Large), obtaining a sequence of token embeddings from the last layer and performing mean pooling to get the embedding for an entity. Another approach is using special algorithms (e.g., TransE) to train KG embeddings on KGs.

For the MCQA task, we use the KG embeddings from Park et al. (2023). For the KGQA task, we do not use pre-trained KG embeddings since the *FreeBase* is a super large KG (over 100GB), so there are no open-source KG embeddings.

A.4 Detail of Training

We use the same way to initialize the parameters in KG-Adapter as Sung et al. (2022), which first calculates the importance score of each weight vector in LLMs and chooses the n rows with top-n importance scores as the initial parameters of new modules.

The inputs and outputs examples during training are shown in Table 6. For different LLMs, we use different templates. For different tasks, we design two types of system instructions (upper for MCQA task and below for KGQA task). We only calculate the loss on answer tokens, such as "(B) fecal matter</s>" in Table 6.

Since the dataset for the KGQA task (WQSP and CWQ) does not have pre-trained KG embedding, the implementation of the Sub-word to Entity Hybrid Initialization on the KGQA task is different in which it only contains the KG representations initialized from the word embedding of LLM.

A.5 Detail of Hyperparameter

We list all hyperparameters of different datasets in Table 5. We do not change any hyperparameters of the LLM.

Hyperparameter	OBQA	CSQA	WQSP	CWQ
learning_rate	5.00E-04	1.00E-04	7.00E-04	7.00E-04
lr_scheduler	polynomial_decay_schedule_with_warmup			
warm_up_epoch	0.1	0.1	0.1	0.1
micro_batch_size	2	2	2	2
marco_batch_size	64	64	64	64
weight_decay	0.02	0.02	0.02	0.02
max_seq_length	1024	1024	800	2048
max_train_epochs	10	10	10	10
early_stop_patience	5	5	3	3

Table 5: Hyperparameters of different datasets.

[3] https://drive.google.com/drive/folders/1GXigUv3MU-Sh4XiY6Wz3xVeNT_s0SuON

<table>
<tr><td rowspan="4">Input
and
Output</td><td>Zephyr for MCQA task</td></tr>
<tr><td><|system|>You are an honest and helpful AI assistant.
Now you're going to do a multiple choice task, you will be given a question and options,
and you need to use the knowledge graph to select the correct option(s).
First output the correct answer(s). If the question does not make any sense, or is not factually coherent,
please answer "I have no comment". If you don't know the answer to the question,
answer "I don't know" instead of sharing false information.</s>
<|user|>
Q: processes sometimes produce waste products Eating and digesting a large meal is guaranteed to produce
(A) disease
(B) fecal matter
(C) fuel
(D) fertilizer
A:</s>
<|assistant|>
(B) fecal matter</s></td></tr>
<tr><td>Llama2 for KGQA task</td></tr>
<tr><td><s><|system|>You are an honest and helpful AI assistant.
Now you're going to do a QA task, you will be given a question, and you need to generate all correct answers and split them by ";".
First output all correct answers. If the question does not make any sense, or is not factually coherent,
please answer "I have no comment". If you don't know the answer to the question,
answer "I don't know" instead of sharing false information.</s>
<|user|>
Q: where is the capital city of assyrians?
A:</s>
<|assistant|>
Assur; Nineveh</s></td></tr>
</table>

Table 6: Examples of our used input and output for different LLMs and different tasks in training. upper is the input and output we used for Zephyr on MCQA task and below is for Llama2 on KGQA task.