SGSH: Stimulate Large Language Models with Skeleton Heuristics for Knowledge Base Question Generation

Shasha Guo[1,2], Lizi Liao[3], Jing Zhang[1,2]*, Yanling Wang[4], Cuiping Li[1,2], Hong Chen[1,2]

[1]School of Information, Renmin University of China, Beijing, China
[2]Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education
[3]Singapore Management University [4]Zhongguancun Laboratory
{guoshashaxing, zhang-jing, licuiping, chong}@ruc.edu.cn
lzliao@smu.edu.sg, wangyl@zgclab.edu.cn

Abstract

Knowledge base question generation (KBQG) aims to generate natural language questions from a set of triplet facts extracted from KB. Existing methods have significantly boosted the performance of KBQG via pre-trained language models (PLMs) thanks to the richly endowed semantic knowledge. With the advance of pre-training techniques, large language models (LLMs) (e.g., GPT-3.5) undoubtedly possess much more semantic knowledge. Therefore, how to effectively organize and exploit the abundant knowledge for KBQG becomes the focus of our study. In this work, we propose **SGSH** — a simple and effective framework to **S**timulate **G**PT-3.5 with **S**keleton **H**euristics to enhance KBQG. The framework incorporates "*skeleton heuristics*", which provides more fine-grained guidance associated with each input to stimulate LLMs to generate optimal questions, encompassing essential elements like the question phrase and the auxiliary verb. More specifically, we devise an automatic data construction strategy leveraging ChatGPT to construct a skeleton training dataset, based on which we employ a soft prompting approach to train a BART model dedicated to generating the skeleton associated with each input. Subsequently, skeleton heuristics are encoded into the prompt to incentivize GPT-3.5 to generate desired questions. Extensive experiments demonstrate that SGSH derives the new state-of-the-art performance on the KBQG tasks. The code is now available on Github[1].

1 Introduction

Knowledge Base Question Generation (KBQG) has attracted a lot of attention owing to its wide range of applications in academia and industry (Guo et al., 2024). On the one hand, KBQG can augment training data for question answering (QA) to improve the performance of QA models (Chen et al., 2023; Guo et al., 2022). On the other hand, KBQG empowers machines to actively ask questions in conversations with humans (Saeidi et al., 2018; Wang et al., 2021).

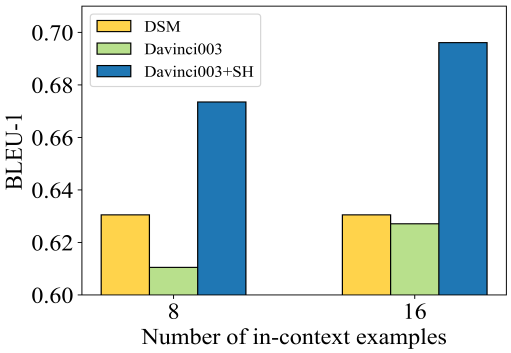

Figure 1: Performance comparison between three advanced methods for KBQG under different numbers of in-context examples on the WebQuestions dataset. The methods include the state-of-the-art PLM-based method DSM (yellow), text-davinci-003 (green), and text-davinci-003 with skeleton heuristics (blue).

The quality of KBQG has been significantly improved, largely attributable to the success of pre-trained language models (PLMs) like BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). A noteworthy example is DSM (Guo et al., 2022), which introduces a meta-learner based on the BART for KBQG, effectively capturing diverse semantic information within a KB. Moreover, AutoQGS (Xiong et al., 2022) designs an auto-prompt approach upon the BART, achieving the low-resource KBQG. Current PLMs, pre-trained on comprehensive corpora, come equipped with rich semantic knowledge, facilitating significant performance improvements in the downstream KBQG task upon fine-tuning.

Recently, large language models (LLMs) such as InstructGPT (Ouyang et al., 2022) and ChatGPT[2], have exhibited impressive capabilities in a variety

*Corresponding author. This work was done during an internship at SMU.
[1]https://github.com/RUCKBReasoning/SGSH
[2]https://openai.com/blog/chatgpt

Findings of the Association for Computational Linguistics: NAACL 2024, pages 4613–4625
June 16-21, 2024

of tasks (Liu et al., 2023; Nan et al., 2023). However, the vast amount of generalized knowledge poses a challenge in extracting pertinent information for the KBQG task, making LLMs fall short of the expected performance on KBQG. As demonstrated in Figure 1, the PLM-based cutting-edge approach DSM outperforms the direct application of the LLM, Davinci003. In view of this, our focus is on how to trigger LLMs to effectively utilize their knowledge to improve the quality of KBQG, which is an under-explored problem in the community of natural language processing.

Inspired by the way humans learn a language, which typically involves acquiring grammatical knowledge before progressing to reading and writing, we summarize the grammatical elements to guide the desired question generation, instead of directly applying an LLM. In this work, the grammatical elements include the question word phrase and the auxiliary verb, which we call "*skeleton*". Through a pilot study, we observe that prompts coupled with skeleton heuristics can boost the performance of Davinci003 on the KBQG task (Cf. the comparison between Davinci003 and Davinci003+SH in Figure 1). In effect, the skeleton heuristics can be viewed as the fine-grained guidance to excavate task-specific knowledge from the LLMs, thereby stimulating the LLMs to generate more accurate questions.

Motivated by the above insights, we propose **SGSH** — a simple and effective framework to **S**timulate **G**PT-3.5[3] with **S**keleton **H**euristics for KBQG, which contains two modules, i.e., a skeleton generator and a black-box LLM (e.g., GPT-3.5). Figure 2 illustrates the overview of SGSH. Specifically, a skeleton generator implemented by a small PLM (e.g., BART) generates the skeleton for each input, where the skeleton is a series of discrete tokens that act as a particular signal to guide the LLM toward the ground-truth question. To train the skeleton generator, we propose an automatic strategy to construct a high-quality training dataset, which leverages a rule-based method to initially extract skeletons and then utilizes the power of ChatGPT to refine these skeletons. Based on the training set, we learn the skeleton generator with a soft prompting strategy to generate the skeleton for each input. Subsequently, the black-box LLM utilizes the skeleton heuristics via *skeleton injection* and *skeleton-aware in-context learning*. Concretely, given a test input consisting of triples along with the corresponding answer, the skeleton injection step integrates the generated skeleton into the test input. Afterward, the skeleton-aware in-context learning step incorporates in-context examples with skeletons to effectively enhance the in-context learning capability for the test input, where each example shares a similar target question with the test input.

Key Contributions. 1) The development of an automatic data-building approach with a soft prompting strategy for effective skeleton heuristic generation. 2) The creation of an enhanced prompting mechanism, with skeleton injection and skeleton-aware in-context learning, steers GPT-3.5 towards generating more precise questions. 3) Demonstrated superiority of our approach over existing methods in both automatic and human evaluations, also proving beneficial for data augmentation in question answering tasks.

2 Pilot Study

To evaluate the effectiveness of the skeleton heuristics in enhancing the performance of KBQG, we undertake a preliminary investigation to analyze.

KBQG. Given a set of triples extracted from a KB and a particular answer, the objective of KBQG is to generate a question associated with the answer. $\mathcal{D} = \{(G_i, a_i, q_i)\}_{i=1}^{N}$ denotes the dataset for training a KBQG model, where G_i represents a subgraph comprising a set of triples, a_i signifies a given answer, and q_i denotes the target question. This research explores the use of black-box LLMs like GPT-3.5 for KBQG, which can only be accessed through APIs.

Modeling. We perform a pilot study on the commonly used KBQG benchmark, WQ (Yih et al., 2016; Talmor and Berant, 2018). To reduce the cost of API usage, we randomly sample 50 test examples $\{(G_j, a_j)\}_{j=1}^{50}$ from the WQ test set for evaluation. The existing state-of-the-art PLM-based KBQG method DSM (Guo et al., 2022) is our baseline. Another baseline is directly using Davinci003 for KBQG, which takes the test example (G_j, a_j) as input and predicts the corresponding question. To investigate the potential benefits of skeleton heuristics, we use a skeleton generator to derive skeletons for the sampled 50 test examples. To train the skeleton generator, we first construct a skeleton training dataset based on $\mathcal{D} = \{(G_i, a_i, q_i)\}_{i=1}^{N}$ by a rule-based method, which extracts the skele-

[3]We use `text-davinci-003` and `gpt-3.5-turbo`.

ton elements (i.e., the question word phrase and the auxiliary verb) from q_i by searching a pre-defined vocabulary of skeleton elements. Based on the skeleton training dataset, we proceed to train a skeleton generator that produces the specific skeleton heuristics for each test example (G_j, a_j). Subsequently, these elicited skeleton heuristics are seamlessly incorporated into the test input to stimulate Davinci003 to generate the desired question. The skeleton heuristics-based approach is denoted as Davinci003+SH.

Observation — skeleton heuristics can unlock the potential of LLMs for the KBQG task. Figure 1 illustrates that directly applying the LLM (i.e., Davinci003) falls short in performance compared to the PLM-based method (i.e., DSM) in terms of BLEU-1 metric. However, Davinci003+SH, which considers the skeleton heuristics, outperforms both DSM and Davinci003. This implies that directly employing LLMs might not fully exploit useful knowledge to generate the intended questions. The incorporation of skeleton heuristics can enhance the performance of LLMs, serving as an accurate guiding signal that aids LLMs in aligning their output with the gold question. Inspired by these findings, we propose our novel approach SGSH.

3 Methodology

3.1 Model Overview

Our proposed SGSH framework comprises two pivotal modules, a PLM-based skeleton generator (e.g., BART) and a frozen GPT-3.5 model (e.g., Davinci003). Figure 2 illustrates the overall framework. The skeleton generator produces skeletons of test inputs as a precise signal to steer GPT-3.5 at a fine-grained level. More specifically, we first construct a skeleton training dataset $\mathcal{D}_\mathcal{S} = \{(G_i, a_i, s_i)\}_{i=1}^N$ based on $\mathcal{D} = \{(G_i, a_i, q_i)\}_{i=1}^N$ leveraging an automatic data construction strategy and then use a small tunable PLM to learn from the obtained training dataset $\mathcal{D}_\mathcal{S}$. Subsequently, GPT-3.5 employs skeleton heuristics through skeleton injection and skeleton-aware in-context learning. These strategies steer GPT-3.5 to skillfully leverage its internal knowledge, thus enhancing its effectiveness in advancing the KBQG task.

3.2 Skeleton Generator

We explain how to (1) perform **an automatic training data construction** strategy to acquire supervised data and (2) **fine-tune** the skeleton generator with learnable prompting to produce skeletons.

Automatic Training Data Construction. In order to effectively train a skeleton generator through supervised fine-tuning, we need to collect labeled data. To avoid costly and time-consuming human annotation, we devise an automatic approach to construct the required data. Drawing inspiration from human cognitive processes, the essential elements of a question are the question word phrase and the auxiliary verb, which are defined as the "*skeleton*". Specifically, we first derive skeletons from $\mathcal{D} = \{(G_i, a_i, q_i)\}_{i=1}^N$ using a rule-based method, which extracts the skeleton elements from q_i by searching a pre-defined vocabulary of skeleton elements. Considering the limitation of the rule-based method, such as difficulty in solving complex questions with nested clauses (Cf. Figure 3), and the powerful capabilities of ChatGPT, we utilize ChatGPT to generate skeletons with a well-designed prompt. Subsequently, we employ ChatGPT as an automatic grader to score the skeletons obtained through the aforementioned methods. We then select the skeleton with the comparatively higher score as the definitive one. By doing this, we obtain the supervised data $\mathcal{D}_\mathcal{S} = \{(G_i, a_i, s_i)\}_{i=1}^N$ consisting of input-skeleton pairs, which are used to train the skeleton generator to infer skeletons for test inputs without requiring any additional manual labeling. Figure 3 shows the overall pipeline.

Fine-tuning a PLM-based Skeleton Generator. To train the skeleton generator, one straightforward method is to perform vanilla fine-tuning on a tunable PLM f_{PLM} (i.e., BART[4]). Motivated by the Prompt Tuning work (Lester et al., 2021), we enhance the vanilla fine-tuning by utilizing a learnable prompting training strategy to effectively train f_{PLM} for precise alignment with the target skeleton (Liang and Liao, 2023). Specifically, we linearize G_i into a triple-based sequence, with each triple separated by commas. Then we append the representations of the prompt tokens $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ to the end of the input (G_i, a_i), which will be updated during the training process. Formally, the objection function is defined as:

$$\mathcal{L}(\theta, \theta_p) = \max_{\theta;\theta_p} \sum_{i=1}^{N} \log P_{\theta;\theta_p}\left(s_i | G_i, a_i, \mathcal{P}\right), \quad (1)$$

where f_{PLM} contains two types of parameters, θ and θ_p. The former is the backbone BART parameters

[4]https://huggingface.co/facebook/bart-base

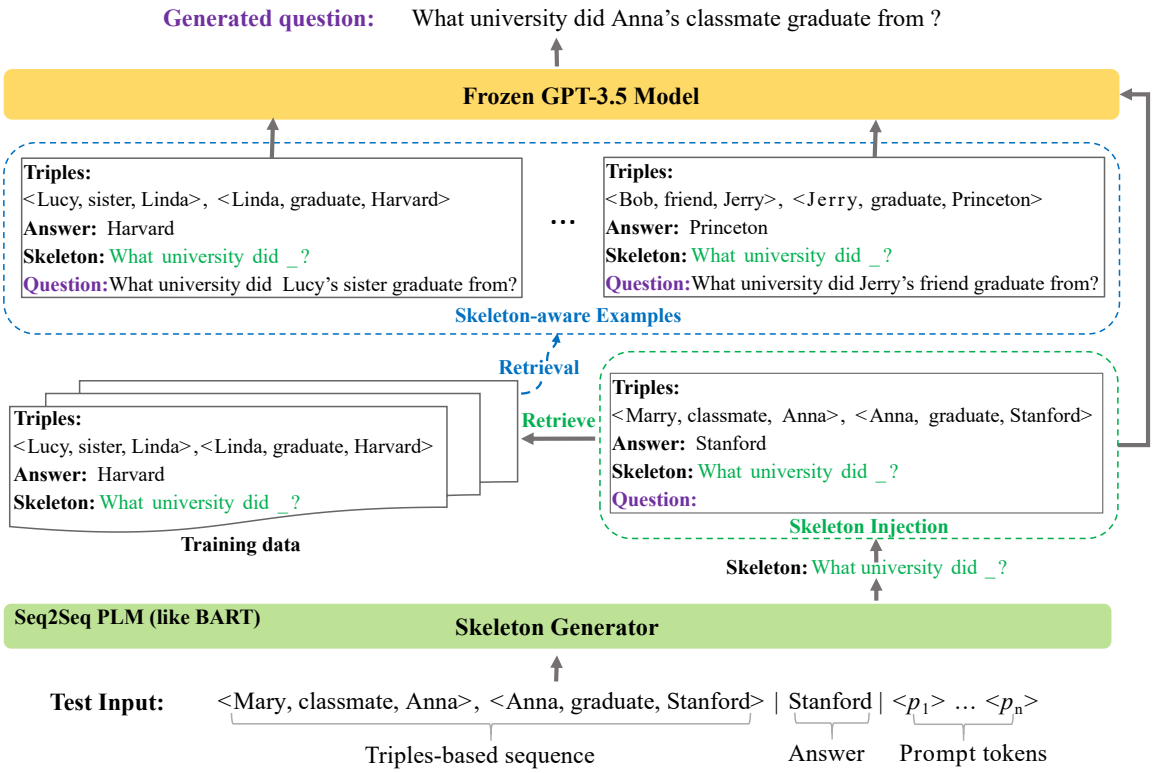

Figure 2: Overview of our SGSH framework, which consists of a PLM-based skeleton generator and a frozen GPT-3.5 model. The skeleton generator, optimized by the learnable prompting strategy, generates the skeleton for each test input. Subsequently, GPT-3.5 leverages skeleton heuristics through skeleton injection and skeleton-aware in-context learning to generate the desired question.

and the latter is the prompt specialized parameters.

Notably, training t groups[5] of learnable prompts, each with different hyperparameters, and subsequently ensembling them during the inference phase can significantly boost the performance of the model. In addition, we find an intriguing phenomenon — few supervised data (i.e., 10%) can achieve comparable performance to full supervised data. We validate the performance across different numbers of learnable prompt groups and supervised data in our experiments.

3.3 Skeleton Heuristics-Enhanced Prompting

Inspired by the observation that skeleton heuristics can stimulate GPT-3.5 for the KBQG task, we introduce skeleton heuristics into the prompt, called *skeleton heuristics-enhanced prompt*, which provides more fine-grained guidance for GPT-3.5. We elaborate on how to utilize skeleton heuristics through two distinct approaches: **skeleton injection** and **skeleton-aware in-context learning**.

Skeleton Injection. The one approach represents injecting the skeleton generated by a skeleton generator f_{PLM} into the test input (G_j, a_j). In specific, given a test input (G_j, a_j) and the corresponding skeleton $s_j = f_{\text{PLM}}(G_j, a_j, \mathcal{P})$, we can obtain the skeleton injection (G_j, a_j, s_j).

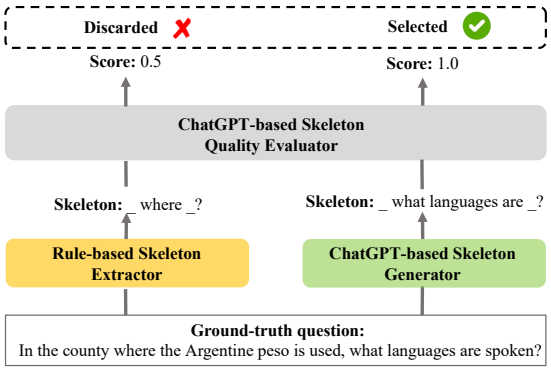

Figure 3: Illustration of the automatic training data construction strategy. We use ChatGPT as an automatic scorer to rate each skeleton generated by the rule-based and ChatGPT-based methods on a scale of 0 to 1.

Skeleton-aware In-Context Learning. The alternative approach incorporates in-context examples with corresponding skeletons (Cf. Automatic Training Data Construction) to facilitate in-context learning for test inputs. In this method, each in-context

[5]In our experiments, we set the value of t as 8.

example shares a similar target question with the test input, thereby enhancing the quality of the question generated by the test input. Previous studies have revealed that different in-context examples may affect the performance of LLMs (Min et al., 2022a; Liu et al., 2022). Motivated by this, we devise a skeleton-aware example selection strategy called *input+skeleton emb*. Concretely, given a test input skeleton injection (G_j, a_j, s_j) and a training example skeleton injection (G_i, a_i, s_i), we apply a small PLM (i.e., BART) trained on $\mathcal{D}$ to obtain their corresponding embeddings e_j and e_i[6]. Since the embedding is used to decode the target question, it contains rich semantic information about the question for the given input-skeleton pairs. Therefore, if e_j and e_i are close in the embedding space, they probably correspond to similar target questions. Then, we calculate the cosine similarity between the test input embedding e_j and each training example embedding e_i in $\mathcal{D}$ and select the Top-k most similar training examples as the skeleton-aware examples, i.e.,

$$SE(e_j) = \underset{i \in \{1,2,...,N\}}{TopK} \frac{e_j \cdot e_i}{||e_j||_2 ||e_i||_2}. \quad (2)$$

The embeddings of the training examples can be calculated and stored in advance so that skeleton-aware examples can be efficiently selected.

Figure 4 illustrates the skeleton heuristics-enhanced prompt consisting of a prompt head, a set of skeleton-aware examples, and a test input with a skeleton. Specifically, the prompt head serves as an explanation of the KBQG task, necessitating clarity and specificity to elicit responses that meet our intended requirements. Skeleton-aware examples are derived from the Top-k skeleton-aware examples $SE(e_j)$, each containing a corresponding question similar to the target question of the test input (G_j, a_j). Notably, the number of skeleton-aware examples k affects the performance of the generated question, which will be validated in the experiments (Cf. Ablation Studies 4.3). The test input skeleton injection (G_j, a_j, s_j) follows a similar format to skeleton-aware examples. The only difference lies in that the question slot will be generated by the GPT-3.5 model.

[6]Experimentally, we utilize the last hidden state of BART encoder as the embedding.

Prompt Head

Please generate a detailed and specific complex **question** using the provided **skeleton** and the information in **triples** related to the **answer**. The question should include all relevant details from the triples while avoiding directly mentioning the answer in the question itself.

Skeleton-aware Examples

Triples: [Triples] \n **Answer:** [Answer] \n
Skeleton: [Skeleton] \n **Question:** [Question] \n
• • • • • •

Test Input

Triples: [Triples] \n **Answer:** [Answer] \n
Skeleton: [Skeleton] \n **Question:**

Figure 4: A skeleton heuristics-enhanced prompt for Davinci003 on KBQG.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate the proposed method on two widely used datasets WebQuestions (WQ) and PathQuestions (PQ) (Zhou et al., 2018). Concretely, WQ includes 22,989 instances from WebQuestionsSP (Yih et al., 2016) and ComplexWebQuestions (Talmor and Berant, 2018), which are divided into training set/dev set/test set with 18,989/2,000/2,000 instances. PQ consists of train/dev/test set with 9,793/1,000/1,000 instances.

Evaluation Metrics. For evaluation, we employ automatic evaluation metrics, human evaluation, and the downstream QA task. For automatic evaluation metrics, we use two classic metrics, namely BLEU-n (n = 1-4) (Papineni et al., 2002) and ROUGE-L (Lin and Och, 2004), which calculate the proportion of identical n-grams between the generated question and the gold question. The former can be seen as precision, while the latter focuses on recall. For downstream QA tasks, we report the F1 score as some questions have multiple answers. To measure the accuracy of the top-1 predicted answer, we use the Hits@1 metric. For human evaluation, we invite three persons to evaluate the relevance and fluency of generated questions.

Baselines. We compare with **Non-PLMs models**, in which MHQG+AE (Kumar et al., 2019) directly feeds the subgraph into Transformer (Vaswani et al., 2017) to generate the question. G2S+AE and G2S+AE+RL (Chen et al., 2023) employ a bidirec-

Model	WQ					PQ				
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
Non-PLMs models										
MHQG+AE	42.35	29.32	18.43	9.63	35.72	45.02	35.86	28.73	17.86	63.45
G2S+AE	53.48	38.67	27.35	20.54	55.61	78.21	69.62	63.35	54.21	82.32
G2S+AE+RL	54.69	39.77	27.35	20.80	55.73	76.05	67.75	61.64	52.19	81.94
PLMs-based models										
T5	50.14	37.01	28.24	21.88	50.20	75.46	67.99	63.01	57.79	75.69
BART	56.39	41.05	29.59	21.46	56.51	79.59	70.63	64.30	55.73	84.54
JointGT(T5)	55.55	39.71	29.61	22.57	56.23	77.87	69.38	63.49	56.17	81.98
JointGT(BART)	56.80	41.27	31.23	24.01	57.29	81.67	72.80	66.97	59.88	83.61
DSM	62.94	48.20	37.50	28.62	64.25	82.44	74.20	68.60	61.03	86.06
B+S	64.44	52.83	44.20	36.70	67.41	86.57	79.03	73.28	65.63	89.45
GPT-3.5-based models										
ChatGPT	56.46	41.76	31.92	24.36	58.23	78.45	70.88	64.88	57.52	84.72
Davinci003	61.68	47.85	38.05	30.00	61.80	80.53	73.55	68.14	61.67	86.48
Our proposed approach										
SGSH(ChatGPT)	63.30	50.34	40.89	32.78	65.46	83.81	77.28	72.04	65.13	87.78
SGSH(Davinci003)	**68.16**	**56.32**	**47.30**	**39.12**	**69.59**	**88.87**	**83.76**	**79.52**	**74.13**	**92.47**

Table 1: Overall evaluation on WQ and PQ (%).

tional gated GNN to encode the subgraph and use the LSTM model to decode the question, whereas the latter adds the reinforcement loss to reward the model for generating better questions. In addition, we also compare with **PLMs-based models**, where BART (Lewis et al., 2020) and T5[7] (Raffel et al., 2020) are directly fine-tuned to solve the KBQG task. JointGT(BART) and JointGT(T5) (Ke et al., 2021) inject the structure-aware semantic aggregation module into the vanilla PLMs to preserve the graph structure and devises three pre-training tasks to learn graph-text alignment. DSM (Guo et al., 2022) focuses on the diversity of subgraphs and models the diverse subgraph via meta-learner (Finn et al., 2017). BART+Skeleton (abbreviated as B+S) represents our developed baseline, trained on the raw input and its corresponding skeleton. Finally, we compare with **GPT-3.5-based models**, where Davinci003 and ChatGPT (i.e., `gpt-3.5-turbo`) are directly used for the task.

4.2 Overall Evaluation

In Table 1, we present the comprehensive assessment findings for WQ and PQ. Based on these findings, the following conclusions can be made: **(1) Directly applying GPT-3.5 to KBQG fails to achieve good performance.** Compared to existing state-of-the-art (SOTA) PLMs-based method (i.e., DSM), we notice that ChatGPT reduces 6.48% BLEU-1 and 6.02% ROUGE-L, while Davinci003 reduces 1.26% BLEU-1 and 2.45% ROUGE-L on WQ. This aforementioned performance does not match the remarkable capabilities of GPT-3.5, which can be explained that employing GPT-3.5 directly with a vanilla prompt only provides coarse-grained guidance but cannot offer specific and accurate guidance direction, resulting in poor quality of the generated questions. **(2) Our proposed framework SGSH can motivate GPT-3.5 to produce high-quality questions, which demonstrates the effectiveness of introducing skeleton heuristics.** We observe that our approaches (i.e., SGSH(ChatGPT) and SGSH(Davinci003)) significantly outperform ChatGPT and Davinci003, because our method incorporates a novel part, i.e., skeleton heuristics, into the vanilla prompt to form a skeleton heuristics-enhanced prompt. This prompt provides more fine-grained guidance for GPT-3.5, which can effectively guide GPT-3.5 to generate questions that are closely related to the ground-truth question. Furthermore, our approach (i.e., SGSH(Davinci003)) surpasses the existing baselines (i.e., Non-PLMs models and PLMs-based models), which indicates the strong capabilities of GPT-3.5 on KBQG. **(3) Injecting skeletons into PLMs can also enhance the performance of KBQG.** B+S derives 8.05% BLEU-1 gain and 10.9% ROUGE-L gain over its corresponding vanilla model BART on WQ and obtains 6.98% BLEU-1 gain and 4.91% ROUGE-L gain on PQ. This indicates the skeleton combined with the raw input can play a very significant role in guiding the question generation.

[7]https://huggingface.co/t5-base

4.3 Ablation Studies

Example Selection Strategy. To investigate the effectiveness of our proposed skeleton-aware example selection strategy, namely "*input+skeleton emb*", we compare it with other example selection strategies, namely "*random*" and "*input emb*". Specifically, the *random* signifies the random selection of examples; the *input emb* denotes the selection of examples based on cosine similarity using the input embedding, which is derived from the last hidden state of the BART encoder; the *input+skeleton emb* introduces our innovative selection strategy that identifies examples based on the combined similarity of both the input and its corresponding skeleton. Table 2 shows the evaluation results. Compared with other strategies, our proposed strategy (i.e., *input+skeleton emb*) achieves the best performance as our strategy takes into account the proximity of the input as well as the consistency of the skeleton in the latent space, which significantly contributes to retrieving examples that are similar to the test input. For instance, *input+skeleton emb* achieves 10.15% gain over *random* and 1.09% gain over *input emb* regarding BLEU-4.

Number of In-Context Examples. To explore the effect of different numbers of in-context examples (k), we set $k \in \{8, 16\}$ for each test input. As shown in Table 2, the performance of SGSH(Davinci003) improves with the increase of k. For example, SGSH(Davinci003) with k = 16 demonstrates better performance compared to k = 8 in terms of BLEU-4 (**74.13%** vs. 72.96%) and ROUGE-L (**92.47%** vs. 91.99%). This suggests that providing GPT-3.5 with few-shot in-context examples is key for enabling its capability on KBQG.

Number of Learnable Prompt Groups. We study the effect of different numbers of learnable prompt groups (t) and set $t \in \{1, 8\}$. As t increases, the performance can be significantly boosted in Table 2. For instance, when t is set to 8, the performance in BLEU-4 obtains 2.83% gain and the performance in ROUGE-L gets 1.63% gain compared to t = 1. It is worth noting that various groups of learnable prompts possess distinct hyperparameters, which are ensembled during the inference stage. This can be explained by the fact that various prompts focus on distinct aspects, thus integrating them together facilitates the KBQG.

Proportion of Training Data for Skeleton Generator. As indicated in Table 2 and Table 1, we find an interesting phenomenon that our SGSH can significantly outperform the existing SOTA model DSM using only 10% of the training data to optimize the skeleton generator (**72.72%** vs. 61.03% in BLEU-4 and **91.66%** vs. 86.06% in ROUGE-L). This shows the effectiveness of the skeleton generator we developed for steering GPT-3.5 toward the target question for the KBQG task with only a small amount of training data.

	BLEU-1	BLEU-4	ROUGE-L
Example selection strategy			
Random	83.50	63.98	87.81
Input emb	88.33	73.04	91.89
Input+skeleton emb	**88.87**	**74.13**	**92.47**
Number of examples (k)			
8	88.16	72.96	91.99
16	**88.87**	**74.13**	**92.47**
Number of learnable prompt groups (t)			
1	87.36	71.30	90.84
8	**88.87**	**74.13**	**92.47**
Proportion of training data for skeleton generator			
10%	88.13	72.72	91.66
100%	**88.87**	**74.13**	**92.47**

Table 2: SGSH(Davinci003) ablation studies on PQ.

Model	GRAFT-Net		NSM	
	F1	Hits@1	F1	Hits@1
Real	**0.622**	**0.681**	**0.666**	**0.727**
-o	0.493	0.575	0.524	0.594
+DSM	0.604	0.664	0.663	0.721
+B+S	0.606	0.676	0.664	0.724
+SGSH(Davinci003)	0.618	0.677	**0.666**	0.726

Table 3: QA performance of GRAFT-Net and NSM.

4.4 Effect on QA Performance

We explore whether our SGSH can contribute to QA tasks. GRAFT-Net (Sun et al., 2018) and NSM (He et al., 2021) are two popular KBQA models utilized for experiments on WebQSP (Yih et al., 2016), a widely adopted KBQA dataset with 2,848 (question, answer) training instances. There are 1,409 (question, answer) pairs in WebQSP overlapping with WQ. Then we can quickly obtain their corresponding subgraphs from WQ, so we conduct experiments by replacing some of the (question, answer) pairs in WebQSP with questions generated by KBQG models on WQ. Specifically, we train GRAFT-Net and NSM on the datasets partially replaced by the pseudo questions generated by DSM, B+S, and SGSH(Davinci003), denoting them as

"+DSM", "+B+S", and "+SGSH(Davinci003)" respectively. We also train GRAFT-Net and NSM on the original WebQSP, denoted as "Real", and a modified version where overlapping instances are eliminated, indicated as "-o". Finally, we compare their performances with Real.

Table 3 reports F1 and Hits@1 of GRAFT-Net and NSM on various datasets. From the results, we can draw the following conclusions. **(1) The generated questions and the corresponding answers form (question, answer) pairs which can be seen as a data augmentation method for KBQA,** because GRAFT-Net and NSM perform better than -o on +DSM, +B+S, and +SGSH(Davinci003). **(2) SGSH(Davinci003) generates better questions than other baselines (i.e., DSM, B+S),** because +SGSH(Davinci003) outperforms all other baselines. **(3) The questions generated by SGSH(Davinci003) closely resemble the actual questions,** because +SGSH(Davinci003) and Real have comparable results.

4.5 Human Evaluation

To further explore the effectiveness of SGSH, we randomly select 50 test examples $\mathcal{S}_{50} = \{(G_j, a_j, q_j)\}_{j=1}^{50}$ from WQ. Then we assess the generated questions from three perspectives: fluency, relevance, and diversity. Fluency aims to evaluate whether the generated questions are human-readable. Relevance measures how relevant the generated question is to the input. Meanwhile, diversity focuses on assessing the extent to which the generated questions differ from the ground truth. We use the five-point Likert scale to score fluency, relevance, and diversity, where 1 is a poor score and 5 is a perfect score. We invite three persons to score all questions generated by our SGSH(Davinci003) and two baselines (i.e., DSM and B+S), and then average their scores as the final result. As shown in Table 4, our proposed SGSH consistently outperforms other baselines in fluency, relevance, and diversity. Besides, our method is comparable to the ground-truth question in fluency and relevance.

Model	Fluency	Relevance	Diversity
DSM	4.17	4.16	3.62
B+S	4.21	4.18	3.56
SGSH(Davinci003)	4.25	4.21	3.81
Ground-truth	**4.39**	**4.25**	-

Table 4: Human evaluation results on WQ.

5 Related Work

Knowledge Base Question Generation (KBQG). KBQG has evolved significantly over recent years, primarily driven by advancements in sequence-to-sequence (Seq2Seq) modeling approaches (Bi et al., 2020; Chen et al., 2023; Kumar et al., 2019; Liu et al., 2019). Early models focused on encoding serialized subgraphs and specific answers into intermediate representations, which were then decoded into questions. These initial methods, while effective, were limited by the scope of their training data. This limitation paved the way for pre-trained language models like BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), which brought a paradigm shift in KBQG (Guo et al., 2023, 2022; Ke et al., 2021). Additionally, LLMs exhibit considerable potential as they possess a substantial quantity of parameters and demonstrate impressive performance on a wide range of downstream tasks such as KBQA (Baek et al., 2023) and fact-checking (Li et al., 2023). However, despite these advancements, a gap remained in harnessing the full potential of LLMs like InstructGPT (Ouyang et al., 2022) and ChatGPT for KBQG tasks. These LLMs, with their extensive parameterization, encode a wealth of generalized knowledge but have been underutilized in the specific domain of KBQG. Concurrently to our work, KQG-COT (Liang et al., 2023) uses unlabeled data to craft prompts to generate questions.

In-Context Learning. The emergence of LLMs introduced a novel capability—In-Context Learning (ICL). ICL enables models like GPT-3.5 to adapt to new tasks through carefully designed prompts, incorporating task descriptions and relevant examples, without necessitating further parameter tuning (Brown et al., 2020). Research in ICL has unraveled intriguing insights, such as its dependency on example selection strategies and prompt templates Zhao et al. (2021), its insensitivity to ground-truth labels Min et al. (2022b), and its unique modalities of Task Recognition and Task Learning Pan et al. (2023). Yet, the application of ICL in KBQG, especially in the context of utilizing large language models for nuanced and accurate question generation from knowledge bases, remains an underexplored area.

6 Conclusion

We explore how to steer GPT-3.5 toward the gold question on KBQG. In this paper, we pro-

pose a simple but effective framework SGSH to **S**timulate **G**PT-3.5 with **S**keleton **H**euristics for KBQG, which provides fine-grained guidance for GPT-3.5 to generate high-quality questions. Specifically, we employ a BART-based skeleton generator that is trained on our constructed training dataset using a learnable prompting strategy to obtain skeleton heuristics. Toward these skeleton heuristics, we then devise a skeleton heuristics-enhanced prompt to trigger GPT-3.5 to align with the ground-truth question. Extensive experiments on widely used datasets demonstrate the advanced performance of our proposed SGSH. In addition, optimizing the skeleton generator with only a small amount of training data (i.e., 10%) can outperform existing SOTA (i.e., DSM). This fine-grained guiding framework could be inspiring for other NLP tasks.

Acknowledgments

This work is supported by National Key Research & Develop Plan (2023YFF0725100) and the National Natural Science Foundation of China (62322214, U23A20299, 62076245, 62072460, 62172424, 62276270). This work is supported by Public Computing Cloud, Renmin University of China. We also acknowledge the support from the China Scholarship Council Scholarship Fund. We are sincerely grateful to all reviewers for their insightful feedback.

Limitations

In this section, we discuss the limitations of this work from two aspects. Firstly, the effectiveness of our method is influenced by the accuracy of the skeleton heuristics. The scarcity of labeled data for training the skeleton generator has motivated us to explore automatic training data construction, utilizing both rule-based and ChatGPT-based approaches. However, the quality of this synthetically produced training data is inherently constrained by the capabilities of ChatGPT. Secondly, a more diverse range of datasets for evaluating the generalizability of KBQG models is under-explored. In future work, we plan to establish a comprehensive benchmark dataset encompassing a broad spectrum of domains. This benchmark will enable a more detailed evaluation of our approach and contribute significantly to the ongoing development within the KBQG community.

References

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

Sheng Bi, Xiya Cheng, Yuan-Fang Li, Yongzhen Wang, and Guilin Qi. 2020. Knowledge-enriched, type-constrained and grammar-guided question generation over knowledge bases. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020*, pages 2776–2786.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th Conference on Neural Information Processing Systems, NeurIPS 2020*, pages 1877–1901.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2023. Toward subgraph-guided knowledge graph question generation with graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 1126–1135.

Shasha Guo, Lizi Liao, Cuiping Li, and Tat-Seng Chua. 2024. A survey on neural question generation: Methods, applications, and prospects. *arXiv preprint arXiv:2402.18267*.

Shasha Guo, Jing Zhang, Xirui Ke, Cuiping Li, and Hong Chen. 2023. Diversifying question generation over knowledge base via external natural questions. *arXiv preprint arXiv:2309.14362*.

Shasha Guo, Jing Zhang, Yanling Wang, Qianyi Zhang, Cuiping Li, and Hong Chen. 2022. Dsm: Question generation over knowledge base via modeling diverse subgraphs with meta-learner. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 4194–4207.

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, WSDM 21*, pages 553–561.

Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu, and Minlie Huang. 2021. Jointgt: Graph-text joint representation learning for text generation from knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, pages 2526–2538.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *Proceddings of the 18th International Semantic Web Conference, ISWC 2019*, pages 382–398.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 3045–3059.

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, ACL 2020*, pages 7871–7880.

Miaoran Li, Baolin Peng, and Zhu Zhang. 2023. Self-checker: Plug-and-play modules for fact-checking with large language models. *arXiv preprint arXiv:2305.14623*.

Jinggui Liang and Lizi Liao. 2023. Clusterprompt: Cluster semantic enhanced prompt learning for new intent discovery. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10468–10481.

Yuanyuan Liang, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. 2023. Prompting large language models with chain-of-thought for few-shot knowledge base question generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pages 4329–4343.

Chin-Yew Lin and FJ Och. 2004. Looking for a few good metrics: Rouge and its evaluation. In *Ntcir workshop*, pages 1–8.

Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie, and Jun Zhao. 2019. Generating questions for knowledge bases via incorporating diversified contexts and answer-aware loss. 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 2019*, pages 2431–2441.

Chao Liu, Xuanlin Bao, Hongyu Zhang, Neng Zhang, Haibo Hu, Xiaohong Zhang, and Meng Yan. 2023. Improving chatgpt prompt for code generation. *arXiv preprint arXiv:2305.08360*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022*, pages 100–114.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 11048–11064.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 11048–11064.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. *arXiv preprint arXiv:2305.12586*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th Conference on Neural Information Processing Systems, NeurIPS 2022*, pages 27730–27744.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319.

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, ACL 2002*, pages 311–318.

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

Marzieh Saeidi, Max Bartolo, Patrick S. H. Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational

machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 2087–2097.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 4231–4242.

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, NAACL-HLT 2018*, pages 641–651.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems, NeurIPS 2017*, pages 5998–6008.

Zichao Wang, Andrew S. Lan, and Richard G. Baraniuk. 2021. Math word problem generation with mathematical consistency and problem context constraints. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 5986–5999.

Guanming Xiong, Junwei Bao, Wen Zhao, Youzheng Wu, and Xiaodong He. 2022. Autoqgs: Auto-prompt for low-resource knowledge-based question generation from SPARQL. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2250–2259.

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, ACL 2016*, pages 201–206.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, pages 12697–12706.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018*, pages 2010–2022.

A Appendix

A.1 Skeleton Generator

Training Process. For our proposed SGSH, the skeleton generator f_{PLM} is a crucial module to obtain skeleton heuristics. Algorithm 1 details the training process of our devised skeleton generator. We present the automatic training data construction strategy to construct labeled data $\mathcal{D}_{\mathcal{S}} = \{(G_i, a_i, s_i)\}_{i=1}^{N}$ for training skeleton generator f_{PLM} in Lines 1-9. More specifically, we utilize a rule-based approach to extract the skeleton s_i' from the target question q_i on $\mathcal{D}$ by searching a pre-defined vocabulary of skeleton elements (Line 3). We employ the powerful potential of ChatGPT to generate the skeleton s_i'' for the target question q_i on $\mathcal{D}$ (Line 4). Subsequently, we use ChatGPT as an automatic grader to score s_i' and s_i'' (Lines 5-6). We choose the higher score one as the refined skeleton s_i to put into the skeleton set S (Line 7). Based on obtained labeled data $\mathcal{D}_{\mathcal{S}} = \{(G_i, a_i, s_i)\}_{i=1}^{N}$, we apply a learnable prompting strategy to train skeleton generator f_{PLM} (Lines 10-13).

Algorithm 1: Skeleton generator training

Input: $\mathcal{D} = \{(G_i, a_i, q_i)\}_{i=1}^{N}$, Training epochs T.
Output: Parameters of skeleton generator f_{PLM}.

1: Initialize the skeleton set $S = \varnothing$;
2: **for** each $q_i \in \mathcal{D}$ **do**
3: Extract the skeleton s_i' using rule-based skeleton extractor;
4: Generate the skeleton s_i'' using ChatGPT-based skeleton generator;
5: Score s_i' and s_i'' using ChatGPT-based skeleton quality evaluator;
6: Obtain refined skeleton s_i = MaxScore (s_i', s_i'');
7: $S = S \cup \{s_i\}$;
8: **end for**
9: Acquire supervised data $\mathcal{D}_{\mathcal{S}} = \{(G_i, a_i, s_i)\}_{i=1}^{N}$ based on $\mathcal{D}$ and S to train f_{PLM};
10: Initialize parameters of learnable prompts θ_p and parameters of backbone BART θ;
11: **for** $epoch \leftarrow 1$ **to** T **do**
12: Calculate $\mathcal{L}(\theta, \theta_p)$ via Eq.1;
13: $\theta, \theta_p \leftarrow \text{AdamW}(\theta, \theta_p, \mathcal{L})$;
14: **end for**
15: Return θ and θ_p

Prompt of ChatGPT-based Skeleton Generator. The rule-based method retrieves the skeleton (i.e., the question word phrase and the auxiliary verb) from the target question through a search within a pre-defined vocabulary of skeleton elements. Obviously, its challenge is in addressing complex ques-

System Prompt

You are a powerful syntax analyzer.

User Prompt

Please generate the question word phrase and the auxiliary verb in the **question** (if any).
Question: **[Question]**

Figure 5: A ChatGPT prompt for generating skeletons.

System Prompt

You are a scoring system used to evaluate the alignment of the **skeleton** to the **question**, where the skeleton is extracted from the question and just contains the question word and the auxiliary verb (if any).

User Prompt

Please score the following two skeletons according to the alignment of the skeleton to the **question**. Scores range from 0 to 1, where a higer score indicates higher accuracy and completeness of the skeleton. Please fairly give the scores for **skeleton 1** and **skeleton 2**, do not output other content.
Question: **[Question]**
Skeleton 1: **[Skeleton 1]**
Skeleton 2: **[Skeleton 2]**

Figure 6: A ChatGPT prompt for scoring and selecting high-quality skeletons.

tions with nested clauses. Hence, we leverage the capabilities of ChatGPT as an enhanced skeleton generator to generate skeletons for target questions, especially those that are inherently complex. Figure 5 demonstrates the system and user prompts of the ChatGPT-based skeleton generator.

Prompt of ChatGPT-based Skeleton Quality Evaluator. To circumvent the costly and time-consuming human selection, we exploit the potential of ChatGPT as an automatic scorer. This capability empowers us to effectively filter out low-scoring skeletons generated by the rule-based skeleton extractor and the ChatGPT-based skeleton generator. Figure 6 illustrates the system and user prompts of the ChatGPT-based skeleton quality evaluator. "Question" denotes the specific target question, "Skeleton 1" corresponds to the skeleton extracted through the rule-based approach, and "Skeleton 2" represents the skeleton generated by the ChatGPT-based generator.

A.2 Experimental Implementation Details

Code Implementation. We implement our method in Pytorch, and run all experiments on

Test Input

Triples:
<Sweden, national_anthem, du gamla, du fria>
<Swedish krona, currency_used, Swedish krona>
Answer: Swedish krona

Ground-truth Question

What kind of money does the country with the national anthem du gamla, du fria use ?

Davinci003

Generated question:
What is the monetary unit of the nation with the national anthem du gamla , du fria ?

SGSH(Davinci003)

Generated skeleton: **What kind of money does** _ ?
Generated question:
What kind of money does the country with the anthem du gamla , du fria use ?

Figure 7: Illustration of an example from WQ dataset, which shows the question generated by Davinci003 and our method SGSH(Davinci003).

Test Input

Triples:
<Savang Vadhana, parents, Mongkut>
<Mongkut, children, Chulalongkorn>
<Chulalongkorn, children, Paripatra>
Answer: Paripatra

Ground-truth Question

Who is the child of kid of Savang Vadhana 's father ?

Davinci003

Generated question:
What is the child of offspring of Savang Vadhana 's dad ?

SGSH(Davinci003)

Generated skeleton: **Who is** _ ?
Generated question:
Who is the child of kid of Savang Vadhana 's father ?

Figure 8: Illustration of an example from PQ dataset, which presents the question generated by Davinci003 and our approach SGSH(Davinci003).

a server with a single Nvidia RTX A6000 (48G) GPU card, an Intel(R) Xeon(R) Gold 5218R CPU, 256GB memory, and the Ubuntu 20.04.2 LTS operating system.

Skeleton Generator f_{PLM}. We employ BART-base[8] as the backbone of the skeleton generator and fine-tune it with the AdamW optimizer. We set

[8]https://huggingface.co/facebook/bart-base

the learning rate as 5e-5, batch size as 16, training epochs as 20. We initialize the learnable prompts from word embeddings in the vocabulary. We compare different lengths of the prompt such as [2, 4, 8, 16, 32], and set it to 16. We train 8 groups of learnable prompts using different learning rates (abbreviated as lr) and batch sizes (abbreviated as bs) including f_{PLM} (lr = 2e-5, bs = 8), f_{PLM} (lr = 2e-5, bs = 16), f_{PLM} (lr = 3e-5, bs = 8), f_{PLM} (lr = 3e-5, bs = 16), f_{PLM} (lr = 4e-5, bs = 8), f_{PLM} (lr = 4e-5, bs = 16), f_{PLM} (lr = 5e-5, bs = 8), and f_{PLM} (lr = 5e-5, bs = 16).

Frozen GPT-3.5 Model. We utilize two versions of the GPT-3.5 series models including `text-davinci-003` (i.e., Davinci003) and `gpt-3.5-turbo` (i.e., ChatGPT) in our experiments. We use our proposed skeleton-aware example selection strategy (i.e., "*input+skeleton emb*") to choose in-context examples and set the number of these examples as 16. We set n as 10 and employ a majority voting approach across the n questions to determine the final question. We set temperature as 0.7, top_p as 1, frequency_penalty as 0, and presence_penalty as 0.

A.3 Running Examples

We provide two illustrative examples for the WQ and PQ datasets in Figure 7 and Figure 8, respectively. For each example, we present the generated questions by Davinci003 and our approach SGSH(Davinci003). we observe that: (1) The questions generated by SGSH(Davinci003) are more closely related to the ground-truth questions compared to Davinci003, which shows the superiority of our devised framework. (2) The skeletons produced by the skeleton generator are similar to the actual skeletons of the ground-truth question, which demonstrates the effectiveness of our devised skeleton generator. (3) The questions generated by Davinci003 express similar semantics to ground-truth questions but differ in surface form, which verifies that Davinci003 contains rich semantic knowledge, but requires more fine-grained guidance to stimulate it toward the ground-truth question. Nevertheless, our proposed SGSH framework effectively addresses this challenge.