Planning First, Question Second: An LLM-Guided Method for Controllable Question Generation

Kunze Li and **Yu Zhang***
Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, Harbin, China
{kzli, zhangyu}@ir.hit.edu.cn

Abstract

In the field of education, for better assessment of students' abilities, generated questions often need to meet experts' requirements, indicating the need for controllable question generation (CQG). However, current CQG methods mainly focus on difficulty control, neglecting the control of question content and assessed abilities, which are also crucial in educational QG. In this paper, we propose an LLM-guided method PFQS (for **P**lanning **F**irst, **Q**uestion **S**econd), which utilizes Llama 2 to generate an answer plan and then generates questions based on it. The plan not only includes candidate answers but also integrates LLM's understanding and multiple requirements, which make question generation simple and controllable. We evaluate our approach on the FairytaleQA dataset, a well-structured QA dataset derived from child-friendly storybooks. In the dataset, the attribute label represents content control, while the local_or_sum and ex_or_im labels denote difficulty control. Experimental results demonstrate that our approach outperforms previous state-of-the-art results and achieves better consistency with requirements compared to prompt-based method. Further application of our method to Llama 2 and Mistral also leads to improved requirement consistency in a zero-shot setting.

1 Introduction

Educational studies over the years have demonstrated that asking questions foster reading comprehension skills, critical thinking skills and writing skills (Godfrey, 2001; Etemadzadeh et al., 2013; Joseph et al., 2016). In order to maximize their educational value, questions posed in the teaching and learning activity should be reasonably designed in terms of content and diversified by the levels of questioning (Shanmugavelu et al., 2020). Recent advancements in natural language processing have spurred active exploration of question generation (QG) systems with a focus on educational applications (Xu et al., 2022; Yao et al., 2022; Zhao et al., 2022). Besides, several studies have recognized the importance of diversity (Eo et al., 2023; Yoon and Bak, 2023) and difficulty (Cheng et al., 2021; Bi et al., 2021; Uto et al., 2023) of generated QA pairs, which are key factors in educational QG.

In certain educational settings, such as exams, it is crucial that automatically generated questions derived from a given context are not only pertinent to the context, but also capable of assessing students' diverse abilities (Francis et al., 2005) and can be controlled in terms of difficulty (Bachman, 1990; Benedetto et al., 2023). However, meeting these practical requirements presents challenges for educational QG. For instance, given the sample in Table 1, existing QG methods usually overlook labels of questions which are annotated by experts,

Story Name: The Sea King Gift
But the herring were now ready, and the students ate enough for six, and gave prince some cold meat which they happened to have in the boat. Prince sat on his hind legs with delight and mewed like a pussy cat. When all was finished, the students handed matte a shining silver coin, and allowed him to fill his pipe with a special kind of tobacco. They then thanked him for his kind hospitality and went on their journey, much regretted by prince, who sat with a woeful expression and whined on the shore as long as he could see a flip of the boat's white sail in the distance. Maie had never uttered a word, but thought the more. She had good ears, and had laid to heart the story about Ahti. "How delightful," thought she to herself, "to possess a fairy cow! How delicious every morning and evening to draw milk from it, and yet have no trouble about the feeding, and to keep a shelf near the window for dishes of milk and junkets! But this will never be my luck. "
Attribute: feeling **Local_or_sum**: local **Ex_or_im**: implicit
Gold Question 1: How did the young men feel after Matte and Maie fed them?
Answer: Grateful.
Gold Question 2: How did prince feel after the young men left?
Answer: Sad.
BART-based QG(feeling ✖ local ✔ implicit ✖, *not similar* to gold questions)
Question 1: What did the students give to the prince? Answer: Some cold meat.
Question 2: What did the prince do when the herring were ready?
Answer: Sat on his hind legs with delight and mewed like a pussy cat.
Prompt-based QG(feeling ✔ local ✔ implicit ✖, *not similar* to gold questions)
Question 1: How did the prince feel when he saw the herring? Answer: Delight.
Question 2: How did the prince feel when the herring were ready? Answer: Delight.
PFQS(feeling ✔ local ✔ implicit ✔, *similar* to two gold questions respectively)
Question 1: How did the students treat the prince? Answer: Kindly.
Question 2: How did the prince feel when the students left? Answer: Sad.

Table 1: An example from FairytaleQA in which questions generated by PFQS achieve greater similarity with gold questions and better consistency with expert-annotated labels in the dataset compared to those generated by BART-based and prompt-based methods.

*Corresponding author.

Findings of the Association for Computational Linguistics ACL 2024, pages 4715–4729

typically focusing on either content or difficulty control of questions, but not both simultaneously. Furthermore, current research lacks a unified standard for defining and measuring difficulty, and a single criterion for difficulty may not be sufficiently reasonable.

To alleviate the above limitations, we propose an LLM-guided method PFQS (**P**lanning **F**irst, **Q**uestion **S**econd) that enhances the controllability of generated questions in terms of content and difficulty simultaneously. Our approach consists of two steps. Firstly, we utilize Llama 2 (Touvron et al., 2023) to generate an answer plan, a concept proposed in this paper, containing various information on candidate answers. In addition to candidate answers and answer-containing sentences extracted by Llama 2, the plan in our method contains various control information to ensure controllable question generation. The inclusion of answer-containing sentences is inspired by Back et al. (2021). In terms of control information, we prioritize content control followed by difficulty control and express control information in various forms, as content control is typically linked to candidate answer selection, while difficulty control is based on these answers. Secondly, we combine designed prompt, generated plan and given context together and feed them into QG models to generate QA pairs. Furthermore, we design several prompts to enable large language models (LLMs) to assist in evaluating whether the content and difficulty of the generated questions meet predefined requirements.

As the example shown in Table 1, questions generated by PFQS are diverse and exhibit higher consistency with gold questions and labels in the dataset. Extensive Experimental results demonstrate that our PFQS method significantly outperforms the existing state-of-the-art method, with an improvement of up to 0.254→0.413 on MAP@1 with RougeL (Lin, 2004) F1 and 0.8783→0.8965 on MAP@1 with BERTScore (Zhang et al., 2019) F1. Evaluations also indicate that PFQS ensures better requirement consistency for both small and large language models compared with common prompt-based method, implying the value of our approach in the era of LLM. The main contributions of this paper are summarized as follows.

- We propose a novel QG method (PFQS), which generates questions based on LLM-generated answer plan. The plan contains more information than usual prompt and is more beneficial for QG.
- We address both content and difficulty control in QG by incorporating various control information in a specific order and form during the planning process in PFQS to effectively utilize the power of control information.
- We conduct experiments on FairytaleQA. The results show that PFQS remarkably outperforms previous state-of-the-art results, improving requirement consistency for both small and large language models.

2 Related Works

2.1 Candidate Answer Selection

Candidate answer selection in question generation (QG) involves selecting potential answers for generating high-quality and relevant questions. Various QG methods (Duan et al., 2017; Subramanian et al., 2018; Yao et al., 2022) extract named entities, noun phrases or key phrases as candidate answers. However, these methods often limit the diversity of candidate answers by focusing only on explicit extractions. To tackle this issue, a recent QG method (Eo et al., 2023) trains an answer generator to produce candidate answers. Additionally, several QG methods (Tang et al., 2017; Dugan et al., 2022) simultaneously train QA and QG models to enhance the relevance between questions and answers, implicitly improving answer selection. In this paper, instead of introducing another model or employing multi-task learning, we leverage the capabilities of LLM to generate an answer plan containing diverse and relevant candidate answers.

2.2 Educational Question Generation

The task of educational question generation aims to automatically generate natural language questions for educational purposes. FairytaleQA (Xu et al., 2022) stands out as a representative dataset in educational QG. The questions in this dataset, annotated by experts, are suitable for evaluating children's reading comprehension skills. Based on this dataset, Yao et al. (2022) present a three-step pipeline that first extracts candidate answers, then generates appropriate questions, and finally ranks the generated question-answer pairs. Dugan et al. (2022) train a QG model based on T5 (Raffel et al., 2020) by multi-task learning, resulting in questions with improved relevance, interpretability, and acceptability. Zhao et al. (2022) propose

a QG framework that involves learning question type distribution and event-centric summarization to address the high-cognitive demand in question generation. Additionally, recent studies (Eo et al., 2023; Yoon and Bak, 2023) shed light on the significance of diversity in question generation. In this paper, we explore a wider array of factors that influence the educational value of generated questions, including their content and difficulty.

2.3 Controllable Question Generation

Controllable question generation (CQG) has two main categories: type controllable question generation (TCQG) and difficulty controllable question generation (DCQG). TCQG is a less-explored task (Cao and Wang, 2021). Cao and Wang (2021) and Gao et al. (2022) define different question type ontologies to generate type-aware questions. DCQG is a relatively new task (Kurdi et al., 2020). The definition of question difficulty has not been consistent over the past few years. Gao et al. (2018) use R-Net (Wang et al., 2017) and BiDAF (Seo et al., 2016) to assess question difficulty and use an LSTM-based model to generate questions. Cheng et al. (2021) defines question difficulty as the number of inference steps needed to answer a question and propose a graph-based framework for question generation. Bi et al. (2021) design five domain-independent features to measure complexity and incorporate soft templates and deep mixture of experts (Shen et al., 2019) to generate difficulty-controllable and high diversity questions. Srivastava and Goodman (2021) and Uto et al. (2023) take learner's ability into consideration and generate questions with appropriate difficulty for each learner. In this paper, we consider both content (similar to type) and difficulty control of generated questions, which are key factors in educational QG.

3 Method

Our question generation method comprises two main steps: plan generation and question generation. In the first step, we use an instruction-tuned LLM, specifically Llama 2 (Touvron et al., 2023), to generate an answer plan based on the given context and predefined requirements (*i.e.* expert-annotated labels in the dataset, where the *attribute* label represents content control, and the *local_or_sum* and *ex_or_im* labels denote difficulty control). The plan typically includes several points that serve as candidate answers or answer-containing sentences extracted by the LLM, incorporating content and difficulty control information. In the second step, the generated plan, along with the context and the designed prompt, is fed into QG models to generate question-answer pairs. The overall architecture of our method is illustrated in Figure 1.

3.1 Plan Generation

Initial Plan Generation During the initial plan generation process, we use prompts to guide the LLM in generating answer plans based on *attribute* labels of questions in the dataset. The detailed design of prompts can be found in Appendix A.1. Mathematically, given the context c, the *attribute* label att, the prompt template T^{init}, we firstly obtain LLM's input $T^{init}(c, att)$. Subsequently, the initial plan $P^{init} = (p_1, p_2, \ldots, p_{T_p^{init}})$ (T_p^{init} denotes the number of points in P^{init}) can be generated by the following expression:

$$P^{init} = \text{LLM}(T^{init}(c, att)) \quad (1)$$

Plan Fusion After discussing the initial plan generation, the next paragraphs will focus on obtaining the *key point* and fusing the two types of plans.

The process of generating the key point is similar to the initial plan generation. We use the LLM to identify answer-containing sentences by designing prompts. The detailed design of the prompts is available in Appendix A.2. With the context c, the question q, the answer a, and the prompt template T^{key}, we can get LLM's input $T^{key}(c, q, a)$. Then, the key point P^{key} can be expressed as follows:

$$P^{key} = \text{LLM}(T^{key}(c, q, a)) \quad (2)$$

After the initial plan and key points are generated, they need to be fused together. This step is necessary because some points in the initial plan are not suitable for generating questions, and these points are expected to be filtered in this step. Given the initial plan $P^{init} = (p_1, p_2, \ldots, p_{T_p^{init}})$ and the key point P^{key}, the fused plan is calculated as shown in Equation (3).

$$P^{fus} = \{p_i | \text{sim}(p_i, P^{key}) \geq t, p_i \in P^{init}\} \quad (3)$$

where $\text{sim}(\cdot, \cdot)$ is a semantic similarity function (for which we use the cosine similarity) and t denotes the similarity threshold. In this expression,

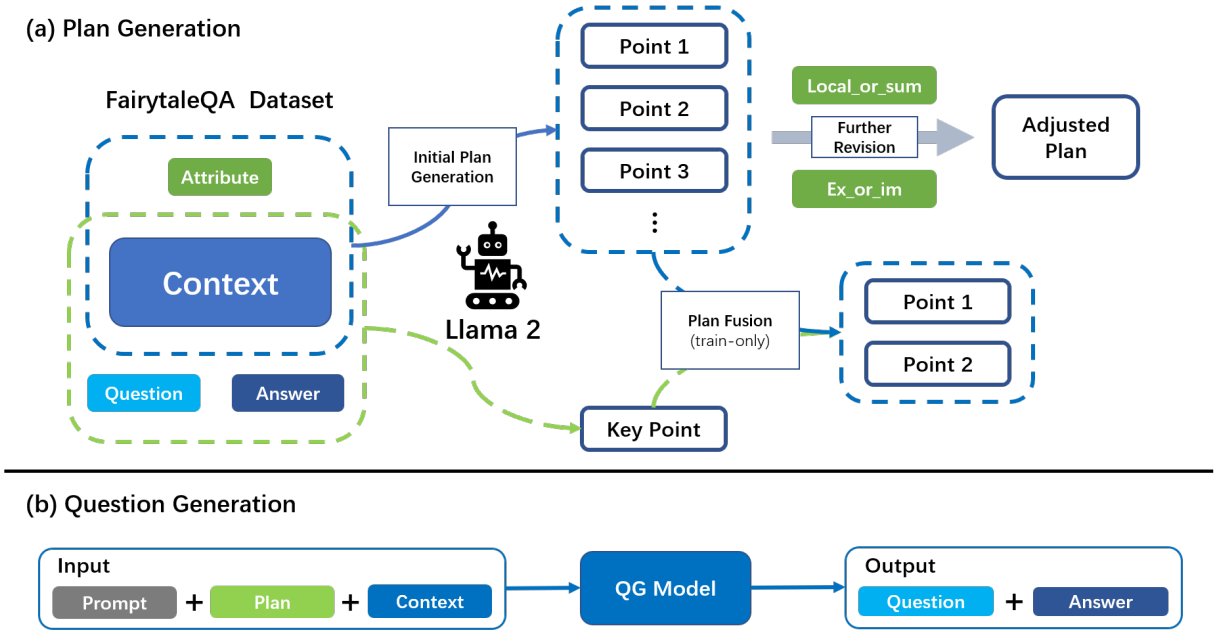

Figure 1: Overall architecture of our QG method. *Attribute*, *local_or_sum* and *ex_or_im* are three question labels in FairytaleQA. *Key point* is derived from gold question-answer pairs and is only used in the training phase.

t is not a fixed value. We set $t = \hat{t}$ which makes $|P^{fus}|_{t=\hat{t}} = \min(T_p^{init}, 5)$.

Note that plan fusion only occurs during the training phase since gold question-answer pairs are unavailable in the testing phase. Moreover, not all initial plans undergo fusion in a training epoch. About 30 percent of initial plans are fused with key points in each epoch.

Further Revision So far, either the initial plan or the fused plan only considers the *attribute* label (content control), ignoring the difficulty control of questions, which is also crucial in educational QG. In this part, we mainly focus on *local_or_sum* (answer involves one or multiple sentences) and *ex_or_im* (answer is explicit or implicit) labels in the dataset, which represent the difficulty of questions from two perspectives. The following paragraphs will introduce how to incorporate these two labels into the existing plan.

Given a plan $P = (p_1, p_2, \ldots, p_{T_p})$ and a source context $c = (s_1, s_2, \ldots, s_{T_c})$, where T_p denotes the number of points in P and T_c denotes the number of sentences in c. Let ls and ei denote *local_or_sum* and *ex_or_im* label respectively. In dataset, $ls \in \{\text{local}, \text{summary}\}$, $ei \in \{\text{explicit}, \text{implicit}\}$. For the four value combinations of ls and ei, the plan will be adjusted in different ways.

Let $\text{sim}(\cdot, \cdot)$ be a semantic similarity function. For each $p_i \in P$, we find the sentence in context which is most semantically similar to p_i. This sentence's index is denoted as $m_i = \text{argmax}_j \text{sim}(p_i, s_j)$. In addition, sentences next to s_j are defined as $\tilde{S}_j = \{s_{j-1}, s_j, s_{j+1}\}$. We will use these representations for the following instructions.

Case 1: ls = local, ei = explicit. We aim for each point of the plan to match a sentence in the context. For each $p_i \in P$, if $\text{sim}(p_i, s_{m_i}) > t_1$ (default value of t_1 is 0.6), then replace p_i by s_{m_i}.

Case 2: ls = local, ei = implicit. Unlike Case 1, it is preferable for points in the plan not to be overly similar to sentences in the context. For each $p_i \in P$, if $\text{sim}(p_i, s_{m_i}) > t_2$ (default value of t_2 is 0.5), then paraphrase p_i using LLM. The Details of paraphrasing are presented in Appendix A.3.

Case 3: ls = summary, ei = explicit. Different from Case 1 and 2, for each $p_i \in P$, we consider both s_{m_i} and the sentences next to it. Let $S_i = \{s | \text{sim}(p_i, s) > t_3, s \in \tilde{S}_{m_i}\}$ (default value of t_3 is 0.5). If $\text{sim}(p_i, s_{m_i}) > t_1$ and $|S_i| > 1$, then replace p_i with sentences in S_i in their original order in c.

Case 4: ls = summary, ei = implicit. For each $p_i \in P$, similar to the above, let $S_i' = \{s | \text{sim}(p_i, s) > t_4, s \in \tilde{S}_{m_i}\}$ (default value of t_4 is 0.45). If $\text{sim}(p_i, s_{m_i}) > t_2$ and $|S_i'| > 1$, then paraphrase p_i using LLM. The Details of paraphrasing are presented in Appendix A.3.

3.2 Question Generation

Before question generation, we design a basic prompt which includes all three labels in FairytaleQA. The detailed design of prompt is shown in Appendix A.4. We combine the prompt, the answer plan and the given context together as input for QG model.

Let att, ls and ei denote the *attribute*, *local_or_sum* and *ex_or_im* labels respectively. Given the prompt template T^{qg}, we firstly get the prompt $a = T^{qg}(att, ls, ei)$. Additionally, given the context c and the plan p, the QG task in this paper can be defined as generating a question-answer pair $\hat{y}$, such that:

$$\begin{aligned}\hat{y} &= \underset{y}{\operatorname{argmax}}\, P(y|a,p,c) \\ &= \underset{y}{\operatorname{argmax}}\, P(y|x)\end{aligned} \tag{4}$$

where $P(y|x)$ is the conditional log-likelihood of the predicted sequence y, given the input $x = (a, p, c)$.

Giving a training corpus: $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{|\mathcal{D}|}$ where $x = (a, p, c)$, the QG model's training objective is to minimize the negative log-likelihood of the training data with respect to all the parameters, denoted by θ.

$$\begin{aligned}\mathcal{L} &= -\sum_{i=1}^{|\mathcal{D}|} \log P(y^{(i)}|x^{(i)};\theta) \\ &= -\sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|y^{(i)}|} \log P(y_j^{(i)}|x^{(i)}, y_{<j}^{(i)};\theta)\end{aligned} \tag{5}$$

Once the model is trained, we do inference using beam search. If the QG model is expected to generate only one QA pair, the model's input remains the same as that during the training process. However, if the QG model is required to generate several QA pairs, the answer plan needs some modification before being fed into the QG model. The final plan consists of several points randomly selected from the original plan, and we believe that such a plan is helpful for enhancing the diversity of the generated questions.

4 Experiments

4.1 Experimental Setup

Dataset We utilize the FairytaleQA (Xu et al., 2022) dataset in our experiments. FairytaleQA is a high-quality dataset created specifically for children's reading comprehension. It comprises 10,580 questions of varying difficulty sourced from 278 children-friendly stories, covering seven types of narrative elements. The training, validation and test sets contain 8,548, 1,025 and 1,007 QA pairs respectively. To generate questions with controlled content and difficulty, we utilize narrative element labels (*i.e. attribute*) and difficulty labels (*i.e. local_or_sum* and *ex_or_im*) in the dataset, which are annotated by experts.

Metrics In line with previous studies, we adopt the MAP@N score as the primary metric in our main experiment. We use MAP@N with Rouge-L (Lin, 2004) and MAP@N with BertScore (Zhang et al., 2019) to evaluate token-level similarity and semantic similarity between generated and ground-truth QA pairs. Additionally, to evaluate the effectiveness of our control over question generation, we assess the consistency between the generated questions and the annotated labels (the *attribute* label for content control, and the *local_or_sum* and *ex_or_im* labels for difficulty control) by Llama 2 (Touvron et al., 2023) and designed rules. Related details are provided in Appendix B.

Baselines In our experiments, we compare our PFQS method with four baselines. 1) FQAG (Yao et al., 2022), which generate questions through a three-step pipeline; 2) SQG (Dugan et al., 2022), which utilizes summaries of given context to generate questions; 3) DQAG (Eo et al., 2023), which enhances the diversity of generated questions by producing different interrogative sentences and implicit/explicit answers; 4) BART-large (Lewis et al., 2020), which is trained on FairytaleQA without any expert-annotated labels. It is important to note that all these baselines are designed for *answer-agnostic* question generation and are expected to generate QA pairs instead of solely generating questions based on answers. Consequently, several QG methods that generate questions alone or generate questions based on given answers are not included in baselines.

4.2 Main Results

The performance of baselines and our method are presented in Table 2 with the following main insights.

Result on MAP@N with Rouge-L As shown in Table 2, in terms of MAP@N with Rouge-L,

Method	MAP@N (Rouge-L F1)				MAP@N (BERTScore F1)			
	Top 10	Top 5	Top 3	Top 1	Top 10	Top 5	Top 3	Top 1
FQAG (Yao et al., 2022)	0.440/0.435	0.375/0.374	0.333/0.324	0.238/0.228	0.9077/0.9077	0.8990/0.8997	0.8929/0.8922	0.8768/0.8776
SQG (Dugan et al., 2022)	0.460/0.455	0.392/0.388	0.344/0.337	0.234/0.242	0.9056/0.9062	0.8953/0.8955	0.8876/0.8878	0.8707/0.8723
DQAG (Eo et al., 2023)	0.500/0.503	0.426/0.429	0.369/0.372	0.247/0.254	0.9156/**0.9178**	0.9046/0.9068	0.8956/0.8977	0.8752/0.8783
BART-large (Lewis et al., 2020)	0.375/0.353	0.354/0.332	0.337/0.314	0.298/0.276	0.8911/0.8900	0.8878/0.8866	0.8851/0.8839	0.8794/0.8784
+Prompt,Plan(PFQS,*ours*)	**0.569/0.547**	**0.535/0.510**	**0.506/0.487**	**0.431/0.413**	**0.9198**/0.9173	**0.9144/0.9121**	**0.9099/0.9082**	**0.8988/0.8965**

Table 2: The main experimental results for our PFQS method. We report Map@N score with Rouge-L F1 and BERTScore F1 for each model. The results for the validation split are displayed on the left side, while those for the test split are shown on the right side. Results of FQAG, SQG and DQAG are sourced from Eo et al. (2023).

Model	MAP@N (Rouge-L F1)					MAP@N (BERTScore F1)				
	SLMQ	Top 10	Top 5	Top 3	Top 1	SLMQ	Top 10	Top 5	Top 3	Top 1
BART-large	0.306	0.353	0.332	0.314	0.276	0.8794	0.8900	0.8866	0.8839	0.8784
+Prompt	0.385	0.446	0.431	0.420	0.394	0.8908	0.9020	0.8996	0.8980	0.8942
+Plan	0.408	0.538	0.496	0.456	0.381	0.8945	0.9157	0.9096	0.9037	0.8920
+Prompt,Plan	0.418	0.542	0.507	0.477	0.401	0.8963	0.9166	0.9113	0.9069	0.8951
+Plan(fused)	0.419	0.545	0.506	0.466	0.389	0.8968	0.9165	0.9111	0.9050	0.8932
+Prompt,Plan(fused)	**0.443**	**0.547**	**0.510**	**0.487**	**0.413**	**0.9001**	**0.9173**	**0.9121**	**0.9082**	**0.8965**

Table 3: Ablation results for our PFQS method on the test set. We report Map@N score with Rouge-L F1 and BERTScore F1 for each model. *SLMQ* denotes **S**ame **L**abel, **M**ultiple **Q**uestions, in which situation there are multiple gold questions with the same annotated labels for given context. SLMQ poses a challenge to the model's ability to generate diverse questions. Prompt, plan and fusion operation all functions in our PFQS method.

our PFQS method demonstrates a significant performance improvement over four baseline models across all splits and top-N results. Especially in the test set, PFQS outperforms DQAG by +0.044 in the MAP@10, +0.081 in the MAP@5, +0.115 in the MAP@3, and +0.159 in the MAP@1, representing a noteworthy advancement.

Result on MAP@N with BERTScore In terms of MAP@N with BERTScore, our method also achieves higher performance in all settings except for the MAP@10 test result. In the best case of MAP@10, DQAG outperforms FQAG and SQG by +0.0079/+0.0101 and +0.0100/+0.0116 respectively in valid/test split, indicating generated QA pairs of DQAG are semantically better than those of FQAG and SQG. By comparison, our method records 0.9198/0.9173 in valid/test split (MAP@10), achieving results comparable to DQAG. This means that questions generated by our method are also high-quality semantically. Additionally, our method outperforms DQAG by +0.0053 in the MAP@5, +0.0105 in the MAP@3, and +0.0182 in the MAP@1 in the test result of BERTScore, the tendency of which is consistent with the MAP@N with Rouge-L F1 result.

4.3 Ablation Study

In Table 3, we investigate the effects of prompt and plan on the model's performance. Specifically, by comparing the results of BART-large + Prompt and BART-large + Plan, we observe that prompt has a greater impact on improving MAP@1 scores than plan for BART-large, indicating that the prompt enhances QG model's accuracy in question generation. By comparison, after adding the plan, BART-large's MAP@3, MAP@5 and MAP@10 scores significantly increase, suggesting that the plan enables QG model to generate a wider variety of questions. Furthermore, BART-large performs better when both prompt and plan are added simultaneously than when either of them is added separately. Therefore, in our PFQS method, we input both prompt and plan into QG model for better performance. Finally, by comparing the results of BART-large + Plan with and without fusion, it is evident that fusion operation positively impacts the improvement of MAP@N scores, regardless of whether prompt is added or not.

The purpose of conducting experiment of SLMQ is to visually demonstrate the distinction between prompt and plan. In Table 3, only BART-large + Prompt performs worse on SLMQ than MAP@1 in both Rouge-L and BERTScore (0.394→0.385, 0.8942→0.8908 respectively), implying the following insights: 1) BART-large + Prompt struggles to generate diverse questions from a fixed input, resulting in lower SLMQ and MAP@3, MAP@5 and MAP@10 scores; 2) BART-large + Prompt achieves a high MAP@1 score on non-SLMQ

datasets, suggesting it excels at question generation without the need for diversity; 3) BART-large + Plan exhibits a completely different tendency in all top-N outcomes compared with BART-large + Prompt and is better at question generation requiring diversity.

4.4 Label Consistency Evaluation

Previous studies have overlooked the importance of expert-annotated labels of questions in FairytaleQA. These labels, however, are vital for a comprehensive assessment of students' abilities and for controlling question content and difficulty. Specifically, *attribute* label covers seven types of narrative elements and questions about different narrative elements typically reflect different question content and assess students' various abilities. For instance, questions about settings typically assess students' ability to extract information, while those on causal relationships often require a deeper understanding of context. Besides, *local_or_sum* and *ex_or_im* labels indicate question difficulty from two dimensions. A method which generates questions more consistent with these two labels is considered to be better at controlling question difficulty.

BART-large In Table 4, we mainly evaluate the impact of our PFQS method on label consistency after it is applied to BART-large. It is observable that BART-large + Plan generates questions which are better consistent with *attribute* label while BART-large + Plan (adjusted) generates questions with better consistency regarding *local_or_sum* and *ex_or_im* labels. This result is easily understood, as the initial plan only contains information on narrative elements (corresponding to *attribute* label) and the adjusted plan takes all three labels into consideration. Additionally, by comparing results of BART-large with and without prompt, it is indicated that prompt performs differently when the plan is adjusted or not. Because the initial plan only contains information on narrative elements and the prompt contains all three labels, BART-large with prompt and plan achieves obviously better label consistency on *local_or_sum* and *ex_or_im* than BART-large with plan only. By comparison, adjusted plan contains information on all three labels itself, as a result, prompt cannot bring much help in this case.

Llama 2 and Mistral In Table 5, we apply our PFQS method to Llama 2 and Mistral in a zero-shot manner, evaluating its impact on label consistency.

Model	Labels		
	Attribute	Local_or_sum	Ex_or_im
BART-large	0.9551/0.9505	0.8894/0.8945	0.5979/0.5698
+Prompt	0.9538/0.9552	0.8959/0.8967	0.6524/0.6279
+Plan	**0.9573/0.9568**	0.8907/0.8951	0.6094/0.6101
+Prompt,Plan	0.9570/0.9560	0.8956/0.9014	0.6613/0.6355
+Plan(adjusted)	0.9534/0.9553	0.8959/**0.9050**	0.6876/0.6439
+Prompt,Plan(adjusted)	0.9551/0.9534	**0.8961**/0.9022	**0.6883/0.6539**

Table 4: Label consistency evaluation results of our PFQS method on BART-large. The result for the validation split is on the left side, and the right side is for the test split.

Model	Labels		
	Attribute	Local_or_sum	Ex_or_im
Llama-2-7B-chat	0.9480/0.9355	0.8858/0.8669	0.5444/0.5869
+Prompt	**0.9614**/0.9507	0.8898/0.8987	0.5980/0.6018
+Plan	0.9561/0.9579	0.8849/0.8997	0.5629/0.6197
+Prompt,Plan	0.9575/0.9529	0.8956/0.9007	0.6205/0.6539
+Plan(adjusted)	0.9510/**0.9631**	**0.8956/0.9017**	**0.6898**/0.7269
+Prompt,Plan(adjusted)	0.9534/0.9579	0.8937/0.8967	0.6829/**0.7319**
Mistral-7B-instruct	0.9530/0.9392	0.8693/0.8868	0.6224/0.5938
+Prompt	0.9487/0.9444	0.8878/0.8957	0.6000/0.6395
+Plan	0.9561/0.9503	0.8810/0.9017	0.5502/0.5740
+Prompt,Plan	**0.9577/0.9605**	**0.8927/0.9027**	0.6176/0.6326
+Plan(adjusted)	0.9538/0.9601	**0.8927/0.9027**	0.6790/**0.7259**
+Prompt,Plan(adjusted)	0.9495/0.9579	0.8917/0.8928	**0.6927**/0.7080

Table 5: Label consistency evaluation results of our PFQS method on LLMs (Llama 2 and Mistral) in a zero-shot setting. The result for the validation split is on the left side, and the right side is for the test split.

Overall, the addition of prompt and plan, especially plan, notably enhances LLM's performance on label consistency, such as 0.8669→0.9017 on *local_or_sum* label for Llama 2 and 0.5938→0.7259 on *ex_or_im* label for Mistral. For both Llama 2 and Mistral, models with adjusted plan exhibit the best performance on control difficulty and are comparable to other models in terms of consistency with *attribute*. Furthermore, by comparing results of two LLMs with and without prompt and referring to results in Table 4, we observe that the impact of prompt on LLMs resembles that of prompt on BART-large. On the one hand, the prompt indeed enhances label consistency when added to Llama 2/Mistral + Plan, due to the fact that the prompt can compensate for *local_or_sum* and *ex_or_im* information not contained in initial plan. On the other hand, the addition of prompt cannot improve the performance of Llama 2/Mistral + Plan (adjusted), and may even lead to a decrease in performance. This is because adjusted plan contains information about all three labels and the addition of prompt does not provide any new information.

4.5 Other Analysis

Performance of Multiple QG Models We investigate our PFQS method's impact on multiple QG models. We choose BART-large (Lewis et al., 2020), T5-base (Raffel et al., 2020) and flan-T5-base (Chung et al., 2022) respectively as QG models for experiments and explore the performance of these QG models in four different situations: 1) trained directly on FairytaleQA; 2) with prompt added; 3) with plan added; 4) with both prompt and plan added.

Model	MAP@N (Rouge-L F1)			
	Top 10	Top 5	Top 3	Top 1
BART-large	0.353	0.332	0.314	0.276
+Prompt	0.446	0.431	0.420	0.394
+Plan	0.538	0.496	0.456	0.381
+Prompt,Plan	**0.542**	**0.507**	**0.477**	**0.401**
T5-base	0.336	0.316	0.300	0.257
+Prompt	0.423	0.407	0.394	0.358
+Plan	0.509	0.473	0.434	0.350
+Prompt,Plan	**0.516**	**0.480**	**0.448**	**0.365**
flan-T5-base	0.341	0.318	0.300	0.254
+Prompt	0.435	0.420	0.408	0.373
+Plan	0.511	0.476	0.435	0.367
+Prompt,Plan	**0.521**	**0.488**	**0.457**	**0.389**

Table 6: FairytaleQA test set evaluation results based on different QG models.

According to the experimental results in Table 6, both prompt and plan contribute to the enhancement of QG model performance, but through different ways. It is indicated that prompt has a stronger effect on improving MAP@1 scores, while plan facilitates the generation of a wider variety of questions and leads to improved performance in MAP@3, MAP@5 and MAP@10. Furthermore, all these three QG models perform better when both prompt and plan are added. Finally, note that above conclusions are applicable to all three models, implying the robustness of our PFQS method.

Performance of Plan Generation Methods We investigate various methods to generate planning. These plans are fed into the BART-based QG model along with given context, and question-answer pairs will be generated. Descriptions of plans in the table are as follows. 1) Initial: corresponding to initial plan P^{init} in the method section and generated by Equation (1); 2) Fused: corresponding to fused plan P^{fus} in the method section and generated by Equation (3) (utilized only in training phase); 3) Filtered: same as fused plan, but used in both training and testing phase; 4) Random: consisting of several points randomly selected from initial plan; 5) Gold: corresponding to key point P^{key} in the method section and generated by Equation (2); 6) Initial+Gold: derived from the connection of initial plan and key point.

Plans	MAP@N (Rouge-L F1)			
	Top 10	Top 5	Top 3	Top 1
Initial	0.538	0.496	0.456	0.381
Fused	0.545	0.506	0.466	0.389
Filtered*	0.515	0.486	0.465	0.406
Random	0.515	0.479	0.450	0.382
Gold*	0.509	0.497	0.484	0.452
Initial+Gold*	**0.610**	**0.587**	**0.568**	**0.511**

Table 7: FairytaleQA test set evaluation results of BART-large after adding multiple plans. The plan marked with * cannot be applied realistically, because gold question-answer pairs are needed to generate it.

In Table 7, as plans have fewer points and become more precise (Initial→Filtered→Gold), the MAP@1 score of the models increases, but MAP@3, MAP@5 and MAP@10 scores decrease. Due to this trend, we ultimately opt for a fused plan for question generation, which retains most of the content of the initial plan and incorporates some information from the filtered and gold plans. In this way, the fused plan leads to improvements in all MAP@N scores compared to the initial plan. Additionally, it is observable that incorporating both initial and gold plans yields significant improvements in all outcomes. However, realistically, gold plan cannot be added. Finding a way to incorporate this type of information would be a meaningful area for future work.

5 Conclusion

In this paper, we propose PFQS, a novel QG method with LLM-guided answer planning at its core. Unlike existing QG methods, PFQS utilizes LLM to generate an answer plan for candidate answer selection, rather than training another model. The generated plan extends the standard prompt, typically including candidate answers, LLM's understanding, and control information, all of which contribute to question generation. Experimental results indicate that PFQS achieves outstanding performance and integrating the plan enhances the diversity and label consistency of the QG model, in comparison to using a standard prompt. Our approach opens up new possibilities for incorporating

more information into prompts with the assistance of LLM, instead of training another model when faced with detailed or complex requirements in controllable question generation tasks.

In the future, with the emergence of more and more educational question generation datasets with expert-annotated labels, the *planning first, question second* idea in our QG method may gradually show its application value.

Limitations

We only evaluate our method and other compared methods on FairytaleQA, as our method requires suitable annotated labels in the dataset for generating answer plans. If there are more datasets with expert-annotated labels, we believe that our method will be more comprehensively evaluated. In addition, we attempted several methods to build a ranker model for helping QG model achieve higher MAP@1 score. However, these ranker models did not bring any stable improvement in either Rouge-L or BERTScore. We believe that a robust ranker model or a dataset including positive and negative samples will be good future works in educational question generation.

Ethics Statement

In this paper, we propose an LLM-guided method to improve the ability of QG model to generate diverse and label-consistent questions. The dataset (FairytaleQA) and models (BART-large, T5-base, flan-T5-base, Llama-2-7B-chat, Mistral-7B-instruct-v0.1, all-MiniLM-L6-v2 and RoBERTa-large) we use are all public and all the references drawn from the work of others are marked with citations. During experiments, random seeds are chosen completely randomly and remain fixed across different configurations of model, so that few biases or discriminations are introduced in experiments. Finally, the plan, prompt and questions are all generated based on the text or labels in FairytaleQA and do not contain harmful information, so there are no ethical issues in our work.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback and helpful comments. This work was supported by the National Natural Science Foundation of China (No.62277002).

References

Lyle F Bachman. 1990. *Fundamental considerations in language testing*. Oxford university press.

Seohyun Back, Akhil Kedia, Sai Chetan Chinthakindi, Haejun Lee, and Jaegul Choo. 2021. Learning to generate questions by learning to recover answer-containing sentences. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1516–1529, Online. Association for Computational Linguistics.

Luca Benedetto, Paolo Cremonesi, Andrew Caines, Paula Buttery, Andrea Cappelli, Andrea Giussani, and Roberto Turrin. 2023. A survey on recent approaches to question difficulty estimation from text. *ACM Computing Surveys*, 55(9):1–37.

Sheng Bi, Xiya Cheng, Yuan-Fang Li, Lizhen Qu, Shirong Shen, Guilin Qi, Lu Pan, and Yinlin Jiang. 2021. Simple or complex? complexity-controllable question generation with soft templates and deep mixture of experts model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4645–4654, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shuyang Cao and Lu Wang. 2021. Controllable open-ended question generation with a new question type ontology. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6424–6439, Online. Association for Computational Linguistics.

Yi Cheng, Siyao Li, Bang Liu, Ruihui Zhao, Sujian Li, Chenghua Lin, and Yefeng Zheng. 2021. Guiding the growth: Difficulty-controllable question generation through step-by-step rewriting. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5968–5978, Online. Association for Computational Linguistics.

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*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Liam Dugan, Eleni Miltsakaki, Shriyash Upadhyay, Etan Ginsberg, Hannah Gonzalez, DaHyeon Choi, Chuning Yuan, and Chris Callison-Burch. 2022. A feasibility study of answer-agnostic question generation for education. In *Findings of the Association for*

Computational Linguistics: ACL 2022, pages 1919–1926, Dublin, Ireland. Association for Computational Linguistics.

Sugyeong Eo, Hyeonseok Moon, Jinsung Kim, Yuna Hur, Jeongwook Kim, SongEun Lee, Changwoo Chun, Sungsoo Park, and Heuiseok Lim. 2023. Towards diverse and effective question-answer pair generation from children storybooks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6100–6115, Toronto, Canada. Association for Computational Linguistics.

Atika Etemadzadeh, Samira Seifi, and Hamid Roohbakhsh Far. 2013. The role of questioning technique in developing thinking skills: The ongoing effect on writing skill. *Procedia-Social and Behavioral Sciences*, 70:1024–1031.

David J Francis, Jack M Fletcher, Hugh W Catts, and J Bruce Tomblin. 2005. Dimensions affecting the assessment of reading comprehension. *Children's reading comprehension and assessment*, pages 369–394.

Lingyu Gao, Debanjan Ghosh, and Kevin Gimpel. 2022. "what makes a question inquisitive?" a study on type-controlled inquisitive question generation. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 240–257, Seattle, Washington. Association for Computational Linguistics.

Yifan Gao, Lidong Bing, Wang Chen, Michael R Lyu, and Irwin King. 2018. Difficulty controllable generation of reading comprehension questions. *arXiv preprint arXiv:1807.03586*.

Kathleen A Godfrey. 2001. Teacher questioning techniques, student responses and critical thinking.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Laurice M Joseph, Sheila Alber-Morgan, Jennifer Cullen, and Christina Rouse. 2016. The effects of self-questioning on reading comprehension: A literature review. *Reading & Writing Quarterly*, 32(2):152–173.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. 2020. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30:121–204.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Ganesan Shanmugavelu, Khairi Ariffin, Manimaran Vadivelu, Zulkufli Mahayudin, and Malar Arasi RK Sundaram. 2020. Questioning techniques and teachers' role in the classroom. *Shanlax International Journal of Education*, 8(4):45–49.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR.

Megha Srivastava and Noah Goodman. 2021. Question generation for adaptive education. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 692–701, Online. Association for Computational Linguistics.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 78–88, Melbourne, Australia. Association for Computational Linguistics.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

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*.

Masaki Uto, Yuto Tomikawa, and Ayaka Suzuki. 2023. Difficulty-controllable neural question generation for reading comprehension using item response theory. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 119–129, Toronto, Canada. Association for Computational Linguistics.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ying Xu, Dakuo Wang, Mo Yu, Daniel Ritchie, Bingsheng Yao, Tongshuang Wu, Zheng Zhang, Toby Li, Nora Bradford, Branda Sun, Tran Hoang, Yisi Sang, Yufang Hou, Xiaojuan Ma, Diyi Yang, Nanyun Peng, Zhou Yu, and Mark Warschauer. 2022. Fantastic questions and where to find them: FairytaleQA – an authentic dataset for narrative comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 447–460, Dublin, Ireland. Association for Computational Linguistics.

Bingsheng Yao, Dakuo Wang, Tongshuang Wu, Zheng Zhang, Toby Li, Mo Yu, and Ying Xu. 2022. It is AI's turn to ask humans a question: Question-answer pair generation for children's story books. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–744, Dublin, Ireland. Association for Computational Linguistics.

Hokeun Yoon and JinYeong Bak. 2023. Diversity enhanced narrative question generation for storybooks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 465–482, Singapore. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhenjie Zhao, Yufang Hou, Dakuo Wang, Mo Yu, Chengzhong Liu, and Xiaojuan Ma. 2022. Educational question generation of children storybooks via question type distribution learning and event-centric summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5073–5085, Dublin, Ireland. Association for Computational Linguistics.

A Design of Main Prompts

A.1 Initial Plan Generation

In this section, we show prompts which are used to guide Llama 2 to generate initial plan. In FairytaleQA, there are totally seven *attribute* labels of questions annotated by experts. We first design a prompt template as shown in Table 8.

Prompt Template for Initial Plan Generation
<s>[INST] «SYS» You are a reading helper. Please help me extract some information and DO NOT do other things. «/SYS» Please help me extract some valuable [ATTRIBUTE] from the following passage. Just tell complete sentences from the original text. Passage: [CONTEXT] [/INST]

Table 8: Prompt template used for initial plan generation. In the template, [ATTRIBUTE] and [CONTEXT] need to be substituted.

To fill the template, we replace [CONTEXT] token by given context and replace [ATTRIBUTE] token using mapping rules shown in Table 9.

Attribute	**Mapped Text**
prediction	predictable information
action	events
setting	places or times
causal relationship	causal relationships
outcome resolution	outcome resolutions
feeling	feelings
character	characters

Table 9: Mapping rules for *attribute* labels.

A.2 Key Point Generation

In this section, we present prompts used to let Llama 2 generate answer-containing sentences which is so-called key point. The prompt template is shown in Table 10.

To fill the template, we replace [CONTEXT], [QUESTION] and [ANSWER] token by given context, gold question and gold answer respectively.

A.3 Paraphrasing

In this section, we present prompts used to let Llama 2 paraphrase given sentences. The prompt template is shown in Table 11.

When use the template, we replace [SENTENCE] token by given sentence and feed the whole prompt into Llama 2.

Prompt Template for Key Point Generation
<s>[INST] «SYS» You are a reading helper. «/SYS» The following are a passage and a pair of question and answer generated from it. Please tell me where to find the answer by using the original text in the passage. Passage: [CONTEXT] Question: [QUESTION] Answer: [ANSWER] [/INST]

Table 10: Prompt template used for key point generation. In the template, [CONTEXT], [QUESTION] and [ANSWER] are special tokens and need to be substituted.

Prompt Template for Paraphrasing
<s>[INST] Paraphrase the following sentence. Don't change the meaning of the sentence too much but use words different from the original sentence as much as possible. Sentence: [SENTENCE] Paraphrased Sentence: [/INST]

Table 11: Prompt template used for paraphrasing. In the template, [SENTENCE] is a special token and needs to be substituted.

A.4 Question Generation

In this section, we present a prompt template used for QG models. The prompt template includes three labels in dataset and is shown in Table 12.

Prompt Template for Question Generation
Focus on the [ATTRIBUTE] part of the text and generate a pair of question and answer. The question is from [LOCA_OR_SUM] of the text, the answer is [EX_OR_IM].

Table 12: Prompt template used for QG models. In the template, [ATTRIBUTE], [LOCAL_OR_SUM] and [EX_OR_IM] are special tokens and need to be substituted.

To use the prompt template, we replace [ATTRIBUTE] and [EX_OR_IM] token by *attribute* and *ex_or_im* label respectively and modify [LOCAL_OR_SUM] token using mapping rules shown in Table 13.

B Label Consistency Evaluation Details

In this section, we introduce how to evaluate the consistency between generated questions and annotated labels (*i.e.* *attribute*, *local_or_sum* and *ex_or_im*). For *attribute* and *local_or_sum* label, prompt templates are shown in Table 14.

When use the template, we replace [CONTEXT],

Local_or_sum	Mapped Text
local	one single sentence
summary	several different parts

Table 13: Mapping rules for *local_or_sum* labels.

[Q&A] and [ATTRIBUTE] token by given context, pair of question and answer and *attribute* respectively, then feed the whole prompt into Llama 2.

For *ex_or_im* label, we find that Llama 2 cannot distinguish explicit and implicit answers very well. Moreover, whether an answer is explicit or implicit mainly involves the comparison and judgment on token level. As a result, we propose the following rules to judge given answer is explicit or implicit:

$A =$ #word appearing in both answer and context
$B =$ #word in answer
if$A \leq B/2$, then the answer is $implicit$;
if$A > B/2$, then the answer is $explicit$.

where #word denote the number of words without stopwords.

C Implementation Details

In our PFQS method, we use Llama-2-7B-chat (Touvron et al., 2023) model to generate answer plans. In the main experiment, following previous studies, we initialize the QG model with pretrained BART-large (Lewis et al., 2020). Hyperparameters are follow: learning rate = 2e-5; batch size = 24; epoch = 10. Besides, we select the commonly used sentence-transformer model, all-MiniLM-L6-v2 (Reimers and Gurevych, 2019) to encode texts and compute cosine similarity between texts. Our code is implemented on Huggingface (Wolf et al., 2020), whereas Adam (Kingma and Ba, 2014) is used for optimization. All models are trained on 1 Nvidia-A100-40G GPU, and most models can be trained within an hour. In some experiments, to evaluate impacts of our PFQS method on LLMs, we use Llama-2-7B-chat (Touvron et al., 2023) and Mistral-7B-instruct-v0.1 (Jiang et al., 2023) model for question generation without training. Additionally, We use RoBERTa-large (Liu et al., 2019) model for BERTScore. For each configuration of our method and all compared methods, we conduct 5 independent runs and report the average score.

D Case Study

Table 15 and Table 16 present two complete examples generated by applying our method to the BART-large model. These two examples show a typical situation of plan generation, where some points are helpful for generating questions, while others are less suitable. Specifically, multiple questions are generated based on some points (point 3 in both case 1 and 2), but there is no question generated based on certain points. This phenomenon indicates that the QG model has the ability to filter plan instead of generating one question for each point of plan. Besides, it is observable that generated QA pairs in two cases have high quality and diversity.

E List of Software and Data Licences Used in this Work

Main dependencies for our method are as follows. They are all public and free for research use.

- FairytaleQA: https://github.com/uci-soe/FairytaleQAData, under an Apache License 2.0.
- Huggingface Transformers: https://github.com/huggingface/transformers/blob/master/LICENSE, under an Apache License 2.0.
- Huggingface Datasets: https://github.com/huggingface/datasets/blob/master/LICENSE, under an Apache License 2.0.
- Huggingface Evaluate: https://github.com/huggingface/evaluate/blob/main/LICENSE, under an Apache License 2.0.
- Pytorch: https://github.com/pytorch/pytorch/blob/main/LICENSE, Misc.
- NLTK: https://github.com/nltk/nltk/blob/develop/LICENSE.txt, under an Apache License 2.0.
- Llama 2: https://github.com/facebookresearch/llama/blob/main/LICENSE, under the LLAMA 2 Community License.
- Mistral: https://github.com/mistralai/mistral-src/blob/main/LICENSE, under an Apache License 2.0.

Prompt Template for Consistency Evaluation
Attribute: <s>[INST] The following are a passage and a pair of question and answer generated from it. Score the following question and answer given the corresponding passage and attribute with respect to relevance with one to five stars, where one star means "irrelevance" and five stars means "perfect relevance". Note that relevance measures how well the question and answer are related to the attribute. Passage: [CONTEXT] Question and Answer: [Q&A] Attribute: [ATTRIBUTE] Stars: [/INST]
local_or_sum: <s>[INST] The following are a passage and a pair of question and answer generated from it. Please tell whether the answer can be found from a single continuous part or several different parts of the passage, namely "local" versus "summary" questions. In general, local questions revolve a single sentence or part of the text, and summary questions require summarizing information from different parts of the passage. Say "local" if the answer can be found just from a single part and say "summary" if it is summarized from several parts. Passage: [CONTEXT] Question and Answer: [Q&A] Local or Summary: [/INST]

Table 14: Prompt template used for consistency evaluation on *attribute* and *local_or_sum* label. In the template, [CONTEXT], [Q&A] and [ATTRIBUTE] are special tokens and need to be substituted.

Context: the boy did as he was bid , for he was a willing enough little fellow ; but when he entered the barn his brothers , in revenge for his having run away from them in the afternoon , set on him and pulled him down , and piled so much straw on top of him that , had his father not come from the house to see what they were all waiting for , he would , of a surety , have been smothered . but when , at supper - time , his mother was quarrelling with the other lads for what they had done , and saying to them that it was only cowards who set on bairns littler and younger than themselves , assipattle looked up from the bicker of porridge which he was supping .
Prompt: Focus on the action part of the text and generate a pair of question and answer. The question is from one single sentence of the text, the answer is explicit.
Answer Plan: 1. The boy did as he was bid, but his brothers set on him in revenge for running away from them in the afternoon. 2. The brothers piled so much straw on top of the boy that he would have been smothered if his father had not come from the house to see what they were doing. 3. The mother scolded the other boys for setting on the smaller and younger boy, and the boy named Assipattle looked up from his porridge during the scolding.
Generated QA Pairs: Question 1: what did assipattle do when his mother was quarrelling with the other lads for what they had done? Answer : looked up from the bicker of porridge which he was supping. Question 2: what did assipattle's brothers do when he entered the barn? Answer : set on him and pulled him down, and piled so much straw on top of him that he would, of a surety, have been smothered. Question 3: what did assipattle look up from? Answer : the bicker of porridge which he was supping.

Table 15: Example 1 on FairytaleQA dataset.

Context:
then the courtiers of the king , who had also set off in pursuit , but whose steeds were less fleet of foot than go - swift , came up , and seized the bridle of the queen 's horse , and led it and its rider back to the palace . she was brought before the council , and judged , and condemned to be shut up in a high tower for the remainder of her life . which thing surely came to pass . as for assipattle , when the proper time came he was married to the princess gemdelovely , with great feasting and rejoicing . and when the old king died they ruled the kingdom for many a long year .

Prompt:
Focus on the outcome resolution part of the text and generate a pair of question and answer. The question is from one single sentence of the text, the answer is explicit.

Answer Plan:
1. The queen was condemned to be shut up in a high tower for the remainder of her life.
2. Assipattle was married to the princess Gemedlovely with great feasting and rejoicing.
3. When the old king died, Assipattle and Gemdelovely ruled the kingdom for many a long year.

Generated QA Pairs:
Question 1: what happened when the old king died?
Answer : they ruled the kingdom for many a long year.
Question 2: what happened when the proper time came for assipattle to marry princess gemdelovely?
Answer : he was married to the princess.
Question 3: what happened when the old king died?
Answer : assipattle and princess gemdelovely ruled the kingdom for many a long year.

Table 16: Example 2 on FairytaleQA dataset.