CrossQG: Improving Difficulty-Controllable Question Generation through Consistency Enhancement

Anonymous ACL submission

Abstract

Automatically generating questions with controlled difficulty has great application value, especially in the field of education. Although large language models have the capability to generate questions of various difficulty levels, the generated questions often fail to align with the given target difficulty. To mitigate this issue, we propose CrossQG, a novel training-free question generation method that enhances difficulty consistency. Specifically, CrossQG consists of two steps: (1) contrast enhancement, which leverages questions from different difficulty levels to enhance the base models' understanding of the target difficulty, and (2) cross filtering, which compares generated questions across different difficulty levels and filters out those that do not meet the target difficulty. We evaluate CrossQG on three high-quality question answering datasets. Experimental results demonstrate that across multiple models, CrossQG significantly outperforms several mainstream methods, achieving superior consistency with target difficulty and improving question quality. Moreover, CrossQG surpasses supervised fine-tuning in various instances even without training.

Context:
A man has a bird. ... Every day the man speaks to the bird. ... "What are you doing?" says the man. "What are you doing?" says the bird. The man is not at home one day. A thief comes in. ... "What are you doing?" The thief is very afraid, so he does not take any things and runs out of the house at last.
Target Difficulty Level: *Hard*
Prompt-based QG
How does the thief react when he hears the bird? (*Too Easy* ✗)
ICL-based QG
What is the purpose of dreaming during sleep? (*Irrelevant* ✗)
Self-refine QG
What does the man's reaction to the thief's presence reveal about his character and values? (*Unanswerable* ✗)
CrossQG
Q1: How does the man's daily interaction with the bird impact the bird's behavior towards intruders? ✓
Q2: What message do you think the story is trying to convey about the relationship between humans and animals? ✓

Table 1: An example from RACE in which questions (corresponding answers are omitted) generated by CrossQG achieve better difficulty consistency compared to other methods.

1 Introduction

The task of Difficulty-Controllable Question Generation (DCQG) aims to generate questions with controlled difficulty levels. It holds significant application value in education, such as improving learning efficiency (Uto et al., 2023; Wang, 2014) and better assessing learners' abilities (Benedetto et al., 2023). The main difficulties of DCQG lie in: (1) appropriately categorizing and representing the difficulty levels of the questions, and (2) ensuring that the generated questions match the target difficulty.

In recent years, with advancements in natural language processing, several studies in DCQG have recognized the importance of question diversity (Bi et al., 2021) and the alignment of question difficulty levels with learners' abilities (Srivastava and Goodman, 2021; Uto et al., 2023). However, current research still faces a number of challenges:

Limited Difficulty Representation. Early DCQG methods (Gao et al., 2019; Bi et al., 2021) directly incorporate difficulty levels into the model framework, an approach that is not applicable for large language models (LLMs). Recent DCQG studies (Wang et al., 2023; Li and Zhang, 2024) leverage model-generated knowledge or answer plans to represent question difficulty, which may be more appropriate for LLMs. However, the difficulty representation for LLMs still remains underexplored.

Low Difficulty Consistency. Difficulty consistency in DCQG evaluate the alignment between generated questions and target difficulty levels, reflecting the model's ability to control question dif-

ficulty. Even for LLMs, ensuring such alignment remains challenging. As illustrated in Table 1, the prompt-based method struggles to ensure the difficulty consistency, while in-context learning (ICL) and self-refine methods tend to generate irrelevant or unanswerable questions.

To address these limitations, we propose CrossQG, an LLM-based DCQG method that improves the difficulty consistency of generated questions. The difficulty of a question is evaluated based on the complexity of answer acquisition. Specifically, the difficulty definition used in this paper is derived from expert-annotated labels provided in FairytaleQA (Xu et al., 2022). For question generation, our method consists of two steps: contrast enhancement and cross filtering. During contrast enhancement, unlike typical in-context learning, we incorporate additional negative examples into the prompt to guide LLMs to regenerate distinct questions. This idea is inspired by the self-refine method (Madaan et al., 2023), which allows LLMs to reflect on the questions they generate for the target difficulty. To further improve the method, we choose to use questions of other difficulties as negative examples, exposing LLMs to the information contained in questions of various difficulties. During cross filtering, we instruct LLMs to extract embeddings from generated questions, which serve as representations of question difficulty. Then, we filter out questions that exhibit high difficulty similarity with those targeted at other levels, thereby strengthening difficulty consistency.

We conduct experiments on three question answering datasets. Experimental results indicate that CrossQG significantly outperforms prompt-based, ICL-based and self-refine methods in question difficulty consistency and quality on multiple LLMs. Moreover, our training-free method even surpasses supervised fine-tuning (SFT) in various instances.

The main contributions of this paper are summarized as follows:

- We propose CrossQG, a novel training-free LLM-based method for DCQG, which enhances the difficulty consistency of generated questions by incorporating cross-difficulty information.
- We design a contrast enhancement module to enhance LLMs' understanding of target difficulty and a cross-filtering module to improve the consistency of generated questions with target difficulty levels.
- We conduct extensive experiments on several datasets. Results show that CrossQG remarkably outperforms multiple training-free methods in both question difficulty consistency and quality on various LLMs.

2 Related Work

2.1 Question Difficulty Representation

Question difficulty representation, which involves encoding difficulty information into models, plays a vital role in DCQG. In earlier years, several studies (Gao et al., 2019; Bi et al., 2021) directly integrate predefined difficulty levels into model frameworks. Besides, multi-hop QG (Cheng et al., 2021) defines difficulty as the number of inference steps required to answer a question, incorporating difficulty information during iterative generation. However, these methods of difficulty representation cannot be easily migrated to LLMs. Recent studies address this gap by utilizing model's intermediate outputs to represent difficulty. For instance, SkillQG (Wang et al., 2023) leverages the skill-specific knowledge to align questions with comprehension skills (Krathwohl, 2002). PFQS (Li and Zhang, 2024) employs the answer plans extracted by models to implicitly represent difficulty. In this paper, we innovatively introduce a difficulty-oriented embedding representation to enhance the difficulty consistency of generated questions.

2.2 Difficulty-Controllable Question Generation

Early research on DCQG mainly focus on small language models. DLPH-GDC (Gao et al., 2019) proposes an LSTM-based model to generate questions of designated difficulty levels. Multi-hop QG (Cheng et al., 2021) introduces an iterative framework that gradually increases question difficulty through step-by-step rewriting. This method is guided by an extracted reasoning chain, and uses GPT-2 (Radford et al., 2019) for question generation. CCQG (Bi et al., 2021) uses a mixture of experts (Shen et al., 2019) as the selector of soft templates. It then leverages BiLSTM (Hochreiter and Schmidhuber, 1997) as encoder to generate questions with controlled complexity. SkillQG (Wang et al., 2023) proposes a question generation pipeline. The pipeline utilizes the skill-specific knowledge extracted by GPT-2 to generate questions with BART (Lewis et al., 2020). Recently, PFQS (Li and Zhang, 2024) proposes an LLM-guided method, which generates questions based

on answer plans. However, almost all methods generate questions separately for different difficulty levels, overlooking the information implied by questions from other difficulty levels. Additionally, there is a lack of sufficient exploration of LLMs for DCQG. In this paper, we primarily focus on LLM-based methods for DCQG. When generating questions of a specific difficulty level, we leverage the questions generated at different levels to improve the performance of LLMs.

3 Method

Given an input context text c and a specific difficulty level $d \in \mathcal{D}$, the objective of the task is to generate several question-answer (QA) pairs $(\mathcal{Q}, \mathcal{A})$, where questions in $\mathcal{Q}$ align with the difficulty level d. This process can be formalized as the following function:

$$(\mathcal{Q}, \mathcal{A}) = \mathcal{F}(c, d) \tag{1}$$

where $\mathcal{F}$ is a question generation method.

Figure 1 illustrates the overall architecture of our method. Before generation, we introduce the difficulty estimation schema used in our approach. During initial question generation, based on the schema, we use LLMs to generate initial QA pairs $(\mathcal{Q}^{\text{init}}, \mathcal{A}^{\text{init}})$ with prompts tailored for different difficulty levels. During contrast enhancement, given difficulty level d, we select QA pairs from $(\mathcal{Q}^{\text{init}} \setminus \mathcal{Q}^{\text{init}}_d, \mathcal{A}^{\text{init}} \setminus \mathcal{A}^{\text{init}}_d)$ as negative examples to help LLMs avoid generating questions with non-target difficulty levels. During cross filtering, we remove questions that exhibit high similarity in difficulty with those targeted at other levels. The following paragraphs will introduce the entire generation process in detail.

3.1 Difficulty Estimation Schema

For difficulty estimation, we refer to the labels annotated by experts in FairytaleQA, a well-structured question answering dataset derived from child-friendly storybooks. These labels have been proven to be scientific and reasonable by several previous works (Eo et al., 2023; Leite and Cardoso, 2024; Li and Zhang, 2024). As shown in Table 2, we define 3 difficulty levels, with difficulty ranging from low to high. The alignment of our labels with those in FairytaleQA ("local/summary", "explicit/implicit") is as follows: (1) easy aligns with (local, explicit); (2) medium maps to both (local, implicit) and (summary, explicit); and (3) hard corresponds to (summary, implicit). To maintain a clear order of difficulty levels, we include two cases under the medium difficulty. Separating these two cases into different difficulty levels would raise ambiguity about which level is more challenging.

3.2 Initial Question Generation

During the initial question generation process, we use prompts to guide an instruction-tuned LLM in generating QA pairs based on the given context and difficulty level. Considering efficiency, we propose two methods at this stage: CrossQG and CrossQG-fast. CrossQG generates questions for varying difficulty levels with different prompts. By comparison, CrossQG-fast employs a single prompt to simultaneously generate questions across all difficulty levels, improving the efficiency of generation. The detailed design of prompts can be found in Appendix A.1.

Given the context c, the difficulty level d, and the prompt template T^{init}, we obtain the complete prompt $T^{\text{init}}(c, d)$[1]. The initial QA pairs of difficulty level d can then be generated using the following expression:

$$(\mathcal{Q}^{\text{init}}_d, \mathcal{A}^{\text{init}}_d) = \text{LLM}(T^{\text{init}}(c, d)) \tag{2}$$

where $\text{LLM}(\cdot)$ represents performing an inference by LLMs.

3.3 Contrast Enhancement

After the initial process, the generated questions might not meet the expected difficulty levels, indicating that LLMs do not fully understand the difficulty requirements. To tackle this problem, we propose a component called contrast enhancement (CE), which leverages negative examples to enhance LLMs' understanding of target difficulty.

Let $\mathcal{D}$ denote all difficulty levels in our difficulty estimation schema. To enhance LLMs' understanding of difficulty level $d (d \in \mathcal{D})$, we randomly select several QA pairs of other difficulty levels to form negative examples E_d, which can be expressed as follows:

$$E_d = \{f((\mathcal{Q}^{\text{init}}_{d'}, \mathcal{A}^{\text{init}}_{d'}), n) | d' \in \mathcal{D} \setminus \{d\}\} \tag{3}$$

where $f(\cdot, \cdot)$ represents a uniform sampling function, and n is a hyperparameter that denotes the number of QA pairs randomly selected from each difficulty level.

[1] CrossQG-fast does not require a specific difficulty level d; this formula is used here for unified expression.

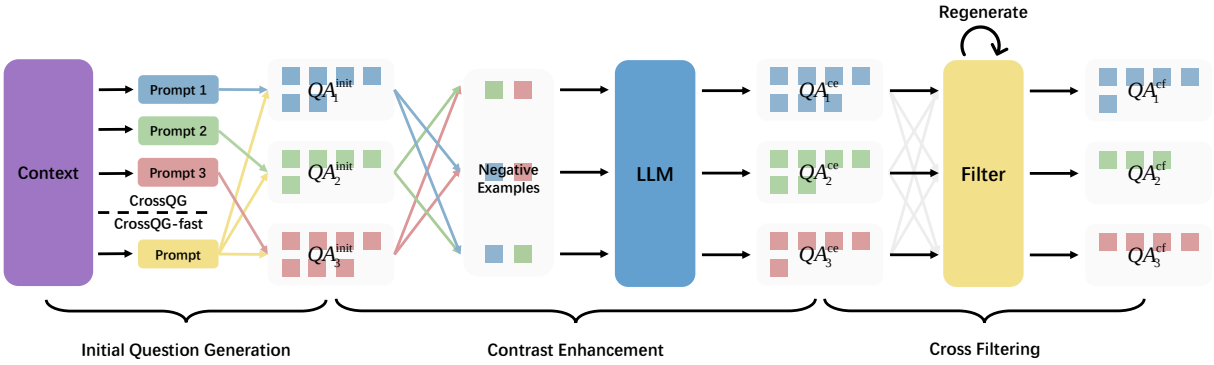

Figure 1: Overall architecture of CrossQG. The figure illustrates the optimization process with three difficulty levels. In the figure, subscript numbers indicate the difficulty of the questions, and each small square represents a QA pair.

Difficulty	Definition
Easy	Answers can be directly found in the text; getting the answer requires focusing on the local information (e.g. one single sentence) in the context.
Medium	Case 1: Answers cannot be directly found in the text; getting the answer requires focusing on the local information (e.g. one single sentence) in the context. Case 2: Answers can be found directly in the text; obtaining the answer involves synthesizing and summarizing information from multiple parts of the context.
Hard	Answers cannot be directly found in the text; obtaining the answer involves synthesizing and summarizing information from multiple parts of the context.

Table 2: Definitions of difficulty levels.

We design prompts to enable LLMs to regenerate questions based on previous difficulty requirements and negative examples. The details of the prompts are available in Appendix A.2.

Formally, with the context c, difficulty level d, negative examples E_d, and the prompt template T^{ce}, the whole prompt is $T^{\text{ce}}(c, d, E_d)$. The QA pairs regenerated during the contrast enhancement process can be expressed as follows:

$$(\mathcal{Q}_d^{\text{ce}}, \mathcal{A}_d^{\text{ce}}) = \text{LLM}(T^{\text{ce}}(c, d, E_d)) \quad (4)$$

where $\text{LLM}(\cdot)$ denotes performing an inference by LLMs.

3.4 Cross Filtering

In this section, we propose a component based on difficulty-oriented embeddings, called cross filtering (CF). The component is designed to filter out questions that may not align with the target difficulty levels, thereby enhancing the difficulty consistency. Specifically, if a question exhibits high similarity in difficulty to questions targeted at other levels, this suggests the question may not match the intended difficulty. In such cases, the question is removed to ensure consistency.

For difficulty-oriented embeddings, we incorporate task definitions into the prompt, inspired by MetaEOL (Lei et al., 2024), to address the complexity of difficulty assessment. For Explicit One-Word Limitation (EOL), we adopt the template from PromptSTH (Zhang et al., 2024) as follows:

This sentence: "[X]" means something

where the last layer's hidden vector for the last token "something" is extracted as the embedding. Additionally, given the length of input context and questions, we replace the placeholder [X] with a fixed-length sentence rather than long inputs to ensure LLMs focus on the task. The complete prompt is provided in Appendix A.3.

The algorithm of a single round is described in Algorithm 1. In the algorithm, P and D are formalized as $\{P_i\}_{i=1}^{n_d}$ and $\{d_i\}_{i=1}^{n_d}$, respectively. P_i represents the list of QA pairs generated by the model for difficulty d_i, and n_d is the number of different difficulty levels. The superscript q denotes the question in a QA pair, while $\text{count}(\cdot)$ represents the number of elements. The function $\text{CE}(\cdot)$ is defined in Equation 4, and $\text{sim}(c, p, q)$ calculates the difficulty similarity between questions p and q based on context c as follows:

$$\text{sim}(c, p, q) = \cos(\text{emb}(c, p), \text{emb}(c, q)) \quad (5)$$

where $\cos(\cdot, \cdot)$ is the cosine similarity function, and $\text{emb}(\cdot, \cdot)$ denotes difficulty-oriented embeddings extracted by LLMs based on the given context and question.

The algorithm consists of two main steps. Firstly, for a given question u^q, we calculate its diffi-

Algorithm 1 Cross Filtering Algorithm

Input: List of list of QA pairs P, list of difficulty levels D, context c

Parameter: Similarity threshold t, high-similarity count threshold s, filter count m

Output: Filtered QA pairs P'

Initialize $Z \leftarrow P$, $P' \leftarrow \emptyset$
for all $u \in P_i$ where $P_i \in P$ **do**
  $A_u \leftarrow \{v | \text{sim}(c, u^q, v^q) > t, v \in P \setminus P_i\}$
  if $\text{count}(A_u) \geq s$ **then**
    Remove u from P_i
  end if
end for
for all $P_i \in P$ **do**
  if $\text{count}(P_i) < m$ **then**
    $E \leftarrow Z_i \setminus P_i$ ▷ Filtered-out QA pairs
    $P' \leftarrow P' \cup \text{CE}(c, d_i, E)$ ▷ Regeneration
  else
    $P' \leftarrow P' \cup P_i$
  end if
end for
return P'

culty similarity with questions from other difficulty levels. We then count the number of questions whose similarity exceeds a threshold t, denoted as $\text{count}(A_u)$. If this count is $\geq s$, the corresponding QA pair u is removed.

Secondly, we assess whether a next iteration is necessary. We denote the remaining number of QA pairs for difficulty level d_i as n_i. If $n_i \geq m$, the filtered QA pairs for d_i are obtained. Otherwise, we regenerate QA pairs for d_i by contrast enhancement. Note that the regenerated QA pairs need to undergo the next cross-filtering process. In contrast, QA pairs of other difficulties will not be updated in the next filtering phase. They will only be used to calculate difficulty similarity with questions that require filtering. The entire process will be repeated until no further QA pairs are regenerated, or it has been repeated three times.

4 Experiments

4.1 Experimental Setup

Datasets. We conduct assessments on three question answering datasets: SQuAD (Rajpurkar et al., 2016), RACE (Lai et al., 2017), and FairytaleQA (Xu et al., 2022). These datasets, sourced from Wikipedia, English exams, and stories respectively, are well-known for their quality. We select a random sample of 1,500 articles for testing, with 500 articles from each dataset.

Metrics. To evaluate the consistency of question difficulty, we first estimate the actual difficulty of the questions, and then calculate the consistency score between the actual and target difficulties.

For difficulty estimation, we initially attempt to utilize GPT-4 (Achiam et al., 2023) in a zero-shot or few-shot manner. However, the average accuracy is below 65% (see complete results in Appendix B.1). Therefore, we train a difficulty classifier based on a manually annotated dataset.

We first construct a dataset called DiffQA. Each entry in DiffQA is a tuple (c, q, a, d), where c, q, a, d represents the context, the question, the answer, and the difficulty level respectively. $d \in \{1, 2, 3\}$ denotes {easy, medium, hard} . The QA pairs are derived from the above three datasets and human annotation. Considering the relatively lower difficulty of SQuAD, we compose some more challenging questions based on its contexts. In addition, we revise some questions from RACE due to their inconsistent format compared to the other two datasets. We employ five annotators to label d, with each annotation undergoing a cross-check by two annotators. In total, we annotate 6500 entries, with 5500 serving as the training set and 1000 as the test set. The details of DiffQA are shown in Appendix B.2.

Then, we utilize DiffQA to train a difficulty classifier based on the Roberta-base model (Liu, 2019), achieving an accuracy exceeding 81%. Detailed results are provided in Appendix B.1.

For consistency score calculation, we collect QA pairs generated by a model across all contexts and target difficulties, denoted as $\mathcal{P}$. Let $d_t(\cdot)$ and $d_a(\cdot)$ represent the target difficulty and the actual difficulty (predicted by our trained classifier) of a QA pair, respectively. We compute the Spearman correlation coefficient (Spearman, 1904) ρ between $\mathcal{D}_t = \{d_t(p) | p \in \mathcal{P}\}$ and $\mathcal{D}_a = \{d_a(p) | p \in \mathcal{P}\}$ to assess their correlation. Additionally, accuracy is used to evaluate whether the difficulty of generated QA pairs aligns with the expected difficulty. The detailed calculations are provided in Appendix B.3.

Baselines. In our experiments, we compare our CrossQG method with four baselines: (1) Prompt (Wei et al., 2022), which leverages the prompt derived from the initial question generation phase of standard CrossQG to generate questions. (2) ICL (Brown et al., 2020), which incorporates several

Method	SQuAD		RACE		FairytaleQA	
	ρ	acc	ρ	acc	ρ	acc
Llama-2-7b-chat						
Prompt	0.3799	0.4776	0.3252	0.4090	0.3672	0.4203
ICL	0.4189	0.5135	0.4117	0.5042	0.3385	0.4759
Self-refine	0.4464	0.4896	0.3619	0.4514	0.3011	0.4126
SFT (*)	0.5923	0.5832	0.6147	0.5851	0.5046	0.5353
CrossQG (ours)	**0.6216**	0.5767	**0.6660**	**0.6121**	0.5805	**0.5749**
CrossQG-fast (ours)	0.6167	**0.5943**	0.6418	0.5791	**0.5835**	0.5678
Llama-2-13b-chat						
Prompt	0.3900	0.4343	0.3704	0.4317	0.4193	0.4314
ICL	0.3512	0.4950	0.2928	0.4474	0.2542	0.4446
Self-refine	0.6211	0.5676	0.5954	0.5587	0.5927	0.5749
SFT (*)	0.6325	0.6130	0.5912	0.5866	0.5270	0.5516
CrossQG (ours)	**0.7066**	**0.6414**	**0.7144**	**0.6369**	**0.6549**	**0.6399**
CrossQG-fast (ours)	0.6834	0.6312	0.6921	0.6118	0.6441	0.6190
Mistral-7b-instruct						
Prompt	0.3946	0.4262	0.4204	0.4547	0.3482	0.4236
ICL	0.4396	0.4996	0.4592	0.5271	0.3910	0.4597
Self-refine	0.4365	0.4433	0.4491	0.4714	0.3790	0.4224
SFT (*)	0.6668	0.6377	0.6436	0.6340	0.5129	0.5483
CrossQG (ours)	**0.6142**	**0.5308**	**0.5972**	0.5652	**0.4408**	**0.4909**
CrossQG-fast (ours)	0.5785	0.5108	0.5837	**0.5692**	0.4183	0.4805

Table 3: Main experimental results on three datasets, with ρ and acc for each method. Best results are highlighted in bold. SFT (*) involves model training, so it serves as a reference, and does not directly participate in the comparison. Results surpassing SFT are marked in red.

in-context examples with target difficulty into the prompt to guide LLMs in generating questions. (3) Self-refine (Madaan et al., 2023), which lets LLMs reflect on the questions they generate based on the Prompt method and regenerate. (4) SFT, which finetunes LLMs using DiffQA.

Other Settings. We conduct experiments on three LLMs: Llama2-7b-chat, Llama2-13b-chat (Touvron et al., 2023), and Mistral-7b-instruct-v0.2 (Jiang et al., 2023). For hyperparameters, the default values are $n = 1$ for contrast enhancement and $s = 1$, $m = 2$ for cross filtering. The parameter t is computed based on the LLM embedding method, as detailed in Section 4.4. Consistent with CrossQG, both ICL and self-refine methods incorporate 2 examples in their prompts.

4.2 Main Results

The results of the baselines and CrossQG are presented in Table 3. Overall, CrossQG significantly outperforms other training-free baselines across all settings, and even surpasses SFT in many instances (marked in red). Specifically, compared with three training-free baselines, CrossQG shows an average increase of at least 0.14 in ρ and at least 0.06 in accuracy, respectively.

CrossQG-fast shows a slight performance decline compared to CrossQG but improves in some cases. However, it is more efficient than CrossQG. Specifically, CrossQG requires at least 2 LLM inferences per difficulty level, while CrossQG-fast needs only $1 + 1/n_d$ inferences, where n_d is the number of difficulty levels. In other words, CrossQG-fast significantly boosts efficiency while only undergoing a slight decrease in performance. Additionally, CrossQG often outperforms SFT, indicating that the finetuned models may struggle to learn effectively when it comes to deep semantic features of questions. Nevertheless, CrossQG still exhibits strong performance in this case.

4.3 Ablation Study

In Table 4, we investigate the impact of contrast enhancement (CE) and cross filtering (CF) on our method. The results indicate that combining CE and CF typically achieves the best performance. For smaller models (Llama2-7b, Mistral-7b), CF is more crucial for LLM performance, as removing CF causes a notable performance drop. Conversely, CE is more impactful for Llama2-13b. This is because smaller models are less effective at self-refining via negative examples, and the cross filtering component can eliminate questions misaligned with target levels. In contrast, Llama2-13b generates questions with better difficulty consistency

Method	SQuAD		RACE		FairytaleQA	
	ρ	acc	ρ	acc	ρ	acc
Llama-2-7b-chat						
CrossQG	**0.6216**	**0.5767**	**0.6660**	**0.6121**	**0.5805**	0.5749
w/o CE	0.6019	0.5741	0.6431	0.5973	0.5697	**0.5816**
w/o CF	0.5524	0.5316	0.6064	0.5797	0.5359	0.5538
Llama-2-13b-chat						
CrossQG	**0.7066**	**0.6414**	0.7144	**0.6369**	**0.6549**	**0.6399**
w/o CE	0.6290	0.5978	0.6681	0.6237	0.5416	0.5814
w/o CF	0.6998	0.6283	**0.7165**	0.6365	0.6516	0.6309
Mistral-7b-instruct						
CrossQG	**0.6142**	0.5308	**0.5972**	**0.5652**	**0.4408**	**0.4909**
w/o CE	0.6101	**0.5407**	0.5928	0.5641	0.4328	0.4722
w/o CF	0.5590	0.4979	0.5368	0.5252	0.3983	0.4624

Table 4: Ablation results for CrossQG, where *CE* denotes contrast enhancement and *CF* denotes cross filtering. We report ρ and κ for each method. The results for answer acquisition difficulty and cognitive level are displayed on the left and right side of "/", respectively.

Method	ρ	acc	NE-acc
Llama-2-7b-chat			
Self-refine	0.3691	0.4503	0.6730
CE	0.5656	0.5565	0.7682
Llama-2-13b-chat			
Self-refine	0.6026	0.5670	0.6634
CE	0.6875	0.6319	0.7699
Mistral-7b-instruct			
Self-refine	0.4176	0.4466	0.6814
CE	0.4953	0.4964	0.7614

Table 5: Results integrating three datasets, where *NE-acc* denotes the accuracy of negative examples.

Method	Precision	Recall
PromptEOL+ID+FS	0.6421	0.4090
PromptSUM+ID+FS	0.6583	0.4120
PromptSTH+DD+LI	0.5856	0.4020
PromptSTH+ID+LI	0.6316	0.3960
PromptSTH+DD+FS	0.6411	0.4170
PromptSTH+ID+FS (Ours)	0.6706	0.4170

Table 6: Precision and recall results of different LLM embedding methods.

through CE, thus cross-filtering offers limited improvement for LLM performance due to potential noise.

4.4 Validation of CE and CF Modules

To verify the quality of negative examples selected by the CE component, we conduct a comparison with the self-refine method. Experimental results are shown in Table 5. For ease of explanation, we refer to the negative examples used in CE and the questions used for reflection in self-refine collectively as negative examples. To ensure fairness, the negative examples for both CE and self-refine are sourced from questions generated by the Prompt method, and are consistent in quantity. Note that the difficulty of questions in negative examples may match the target level, making them not true negatives. Utilizing the difficulty classifier trained with DiffQA, we assess the accuracy of negative examples in both methods to measure its correlation with model performance. It is evident that that CE's negative examples exhibit significantly higher accuracy than those from self-refine. Furthermore, model performance is closely tied to negative example accuracy. Considering that the difficulty estimation schema involves deep semantic features of the questions, we speculate that LLMs demand higher-quality examples in this case.

To evaluate the quality of difficulty-oriented embeddings extracted by LLMs, we compare our method with other LLM embedding approaches. We randomly select 2,000 pairs of LLM-generated questions: half have the same difficulty, and the other half have different difficulties. The difficulty labels are annotated by our difficulty classifier. LLM embedding methods are compared across three aspects: 1) EOL method: PromptEOL (Jiang et al., 2024), PromptSTH and PromptSUM (Zhang et al., 2024). 2) task definition: direct definition (DD) and indirect definition (ID). 3) the sentence to replace the placeholder: long input (LI) and fixed-length sentence (FS). The detailed prompts are provided in Appendix A.3, and the results are presented in Table 6.

We compare the precision of various methods

when the recall is approximately 0.4 (by adjusting the threshold t). This is because: 1) if the recall is higher, the precision will be too low, making cross-filtering less effective; 2) if the recall is lower, the number of filtered-out questions per round will decrease, leading to less efficient filtering. Comparing the results with different EOL methods, it is observable that PromptSTH is the best EOL method. Moreover, methods with ID significantly outperform those with DD. This is because LLMs are not familiar with the concept of difficulty. LLMs are better at finding answers and their derivations from contexts. Additionally, since there are only three different difficulty levels, the difficulty representation space is limited. Furthermore, methods with FS achieve higher precision than those with LI. Too long inputs may distract the LLMs from the task, while the fixed-length sentence helps the LLMs to concentrate on the task, thereby enhancing their performance.

4.5 Analysis

Impact of Negative Examples Count. In contrast enhancement, we randomly select n QA pairs for each other difficulty level to form negative examples. While increasing the number of negative examples may help LLMs better understand the target difficulty, it can also introduce noise. We examine how different values of n affect model performance, with results shown on the left of Figure 2. For Llama2 series models, performance deteriorates as n increases, whereas the trend is opposite for Mistral-7b. Overall, $n = 1$ is a better parameter choice.

Impact of High-similarity Count Threshold. In cross filtering, the high-similarity count threshold s reflects a trade-off between filtering precision and efficiency. As s increases, the filtering process becomes more precise but less efficient. We investigate the impact of different values of s on model performance, with results displayed on the right of Figure 2. Overall, at $s = 1$, the two smaller models achieve their best performance, while the Llama2-13b model also exhibits competitive performance. Therefore, we select 1 as the value for parameter s.

Human Evaluation for Question Quality. In the question quality study, we first randomly select 150 articles from three datasets. Then, we perform uniform sampling on QA pairs of various difficulty levels generated by different models for human evaluation. Three dimensions are rated from 1 (worst) to 5 (best): (1) correctness—whether the question and answer match and are semantically correct; (2) relevancy—whether the question is relevant to the given context; (3) diversity—whether the QA pairs are diverse from each other. As shown in Table 7, CrossQG outperforms the Prompt and ICL methods across three quality metrics. Additionally, it also surpasses the self-refine method in terms of the relevance and diversity of the generated questions. This indicates that our method not only enhances the consistency of question difficulty but also improves the quality of questions.

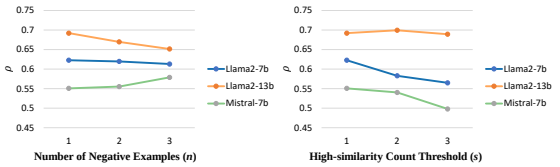

Figure 2: The ρ performance of models under different numbers of negative examples (left) and high-similarity count thresholds (right). The results integrates three datasets.

Method	Correctness	Relevancy	Diversity
Prompt	4.284	4.379	2.025
ICL	3.794	3.619	2.071
Self-refine	**4.602**	4.428	2.428
CrossQG (ours)	4.588	**4.453**	**2.664**

Table 7: Human evaluation on correctness, relevancy and diversity.

5 Conclusion

In this paper, we propose CrossQG, a novel training-free DCQG method aimed at optimizing the difficulty consistency of generated questions. CrossQG leverages information from various difficulty levels, which is often overlooked in previous research, to assist in generating questions at the target difficulty. It first employs contrast enhancement to select questions from different difficulty levels as negative examples. Then, it utilizes cross filtering to eliminate questions that exhibit high difficulty similarity with those targeted at other levels. We conduct experiments on three datasets. Results across multiple LLMs demonstrate that CrossQG achieves superior difficulty consistency and quality compared to three training-free baselines. In addition, it surpasses SFT in many instances. Future research will explore methods to enhance the robustness of CrossQG against noisy data.

Limitations

CrossQG may be affected by noise, as the precision of negative example selection and filtering can be further improved. Future work will explore the impact of noise on these components and aim to develop a more robust method. Additionally, CrossQG is a training-free method. In the future, we will explore Parameter-Efficient Fine-Tuning (PEFT) methods to enhance model performance while ensuring efficiency.

Ethics Statement

In this paper, we propose a novel DCQG method aimed at enhancing the difficulty consistency of questions generated by large language models. The three question answering datasets and all base models are publicly available. In addition, all references derived from prior works are marked with citations. During the experiments, random seeds are selected entirely at random and maintained consistently across different model configurations. In this way, we minimize bias and discrimination in our experiments. Lastly, the QA pairs are generated based on the text in datasets and do not include any harmful content. Overall, we avoid any ethical concerns in our research.

We employ 5 annotators with undergraduate degrees to perform annotations. We pay $12 USD per 100 annotations, which includes both question difficulty and quality estimation. To ensure the anonymity and privacy of the annotators, we exclude all personal identifiers and retain only the annotation results.

References

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

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.

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

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.

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.

Yifan Gao, Lidong Bing, Wang Chen, Michael R Lyu, and Irwin King. 2019. Difficulty controllable generation of reading comprehension questions. In *IJCAI*, pages 4968–4974.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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*.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2024. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196, Miami, Florida, USA. Association for Computational Linguistics.

David R Krathwohl. 2002. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. 2024. Meta-task prompting elicits embeddings from large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10141–10157, Bangkok, Thailand. Association for Computational Linguistics.

Bernardo Leite and Henrique Lopes Cardoso. 2024. On few-shot prompting for controllable question-answer generation in narrative comprehension. In *Proceedings of the 16th International Conference on Computer Supported Education, CSEDU 2024, Angers, France, May 2-4, 2024, Volume 2*, pages 63–74. SCITEPRESS.

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.

Kunze Li and Yu Zhang. 2024. Planning first, question second: An LLM-guided method for controllable question generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4715–4729, Bangkok, Thailand. Association for Computational Linguistics.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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.

C Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

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.

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.

Tzu-Hua Wang. 2014. Developing an assessment-centered e-learning system for improving student learning effectiveness. *Computers & Education*, 73:189–203.

Xiaoqiang Wang, Bang Liu, Siliang Tang, and Lingfei Wu. 2023. SkillQG: Learning to generate question for reading comprehension assessment. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13833–13850, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. In *International Conference on Intelligent Computing*, pages 52–64. Springer.

A Design of Main Prompts

A.1 Initial Question Generation

In this section, we show prompts which are used to guide LLMs to generate initial QA pairs with given difficulty levels. For CrossQG, we design a prompt template as shown in Table 8.

Prompt Template for CrossQG
<s>[INST] Your task is to generate pairs of questions and answers according to the following context, meeting all the requirements. Context: [Context] Requirements: 1. [Difficulty Definition] 2. Answers must be clear, concrete, and well-justified based on the context. [Supplement] Output Format: - Q1: {question} - A1: {answer} - Q2: {question} - A2: {answer} - Continue as needed... [/INST]

Table 8: Prompt template used for the initial question generation in CrossQG. In the template, [Context], [Difficulty Definition] and [Supplement] need to be substituted.

In the template, we replace the [Context] token with the given context. In addition, we substitute the [Difficulty Definition] token with the corresponding difficulty definitions shown in Table 2 based on the difficulty level. Finally, we replace the [Supplement] token according to the mapping rules shown in Table 9.

Difficulty	Supplement
Easy	Ensure that the answers can be directly and unambiguously located within the text.
Medium	Make sure that each question distinctly follows either Case 1 or Case 2 as defined.
Hard	Ensure that the questions demand a deep understanding and interaction with the context, leading to comprehensive and insightful answers.

Table 9: Mapping rules for difficulty levels in initial question generation.

For CrossQG-fast, we design a different prompt template, as shown in Table 10.

A.2 Contrast Enhancement

In this section, we present prompts used to let LLMs generate QA pairs with required difficulty levels given negative examples of other levels. The prompt template is shown in Table 11.

Prompt Template for CrossQG-fast
<s>[INST] Your task is to generate pairs of questions and answers according to the following context, meeting all the requirements. Context: [Context] Requirements: 1. A question should test any difficulty level of given difficulties, and all generated questions are expected to cover all difficulty levels. 2. Answers must be clear, concrete, and well-justified based on the context. Difficulties: - EASY: Answers can be directly found in the text; getting the answer requires focusing on the local information (e.g. one single sentence) in the context. - MEDIUM: Case 1: Answers cannot be directly found in the text; getting the answer requires focusing on the local information (e.g. one single sentence) in the context. Case 2: Answers can be found directly in the text; obtaining the answer involves synthesizing and summarizing information from multiple parts of the context. - HARD: Answers cannot be directly found in the text; obtaining the answer involves synthesizing and summarizing information from multiple parts of the context. Ensure that: - For easy questions, the answers can be directly and unambiguously located within the text. - For medium questions, each question distinctly follows either Case 1 or Case 2 as defined. - For hard questions, the questions demand a deep understanding and interaction with the context, leading to comprehensive and insightful answers. Output Format: EASY: - Q1: {question} - A1: {answer} - Q2: {question} - A2: {answer} - Continue as needed... MEDIUM: - Q1: {question} - A1: {answer} - Q2: {question} - A2: {answer} - Continue as needed... HARD: - Q1: {question} - A1: {answer} - Q2: {question} - A2: {answer} - Continue as needed... [/INST]

Table 10: Prompt template used in the initial question generation phase of CrossQG-fast. In the template, [Context] needs to be substituted with the given context.

Prompt Template for Contrast Enhancement
[Initial Prompt] [Negative Examples]</s> <s>[INST] [Error Analysis] [Supplement] Output Format: - Q1: {question} - A1: {answer} - Q2: {question} - A2: {answer} - Continue as needed... [/INST]

Table 11: Prompt template applied in contrast enhancement. In the template, [Initial Prompt], [Negative Examples], [Error Analysis] and [Supplement] are special tokens and need to be substituted.

Given the input context c, target difficulty level d, we first substitute the [Initial Prompt] token with $T^{\text{init}}(c, d)$ (relative prompt template shown in Table 8). Then, we replace the [Negative Examples] token with E_d computed by Equation 3. Finally, we replace the [Error Analysis] and [Supplement] tokens according to the mapping rules shown in Table 12.

A.3 Difficulty-Oriented Embeddings

The prompt template (PromptSTH+ID+FS in Section 4.4) for guiding LLMs in embedding generation is presented in Table 13. Other prompts mentioned in Section 4.4 are shown in Table 14. In these templates, [Context] and [Question] are substituted with the specific context and question.

A.4 Others

In this section, we show prompts which are used in the main experiments. The prompt templates used for ICL and SFT are presented in Table 15 and 16 respectively.

In both templates, we replace the [Context], [Difficulty Definition] and [Supplement] tokens according to the substitution rules shown in Appendix A.1. For ICL, we need to replace the [Example i] token with "Example i:\n Context: c_e\n Q: q_e\n A: a_e\n", where c_e, q_e and a_e denote the context, question and answer of the example, respectively. For SFT, LLMs utilize the prompt as input and are fine-tuned to generate one QA pair at a time. Consequently, it is necessary to adjust the descriptions of "question" and "answer" in the prompt to the singular form.

Difficulty	Error Analysis	Supplement
Easy	The above questions are too hard to answer. Answers were not adequately justified with direct text references or focused on multiple text areas instead of local information. Please generate easier questions which meet the requirements.	Ensure that the questions only require simple information extraction and superficial understanding with the context, leading to easy and direct answers.
Medium	The above questions are either too easy or too hard to answer. Answers were justified with direct text references (too easy) or focused on multiple text areas and required summarization (too hard). Please generate questions which meet the requirements.	Ensure that the questions require a moderate (not too deep, not too simple) understanding and interaction with the context.
Hard	The above questions are too easy to answer. Answers were justified with direct text references or focused on local information instead of multiple text areas. Please generate questions which meet the requirements.	Ensure that the questions demand a deep understanding and interaction with the context, leading to comprehensive and insightful answers.

Table 12: Mapping rules for difficulty levels in contrast enhancement.

Prompt Template for Embedding Generation
In this task, you're given a context and a question. Your task is to find the answer and describe how to get the answer. Context: [Context] Question: [Question] For this task, this sentence: "Given the question, the answer and its derivation are" means something

Table 13: Prompt template applied in difficulty-oriented embedding generation. In the template, [Context] and [Question] are special tokens and need to be substituted.

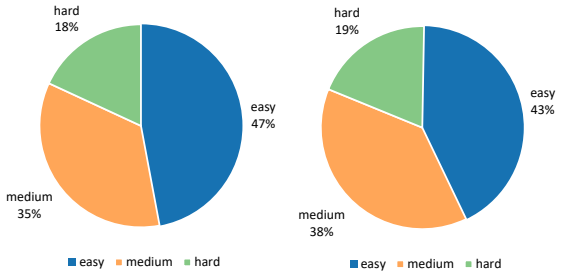

Figure 3: The proportion distribution of d in train split (left) and test split (right) of DiffQA.

B Details of Metrics

B.1 Experiments of Difficulty Estimation

For difficulty estimation, we compare the accuracy of GPT-4 and our classifier on the test split of DiffQA, as shown in Table 17. Our classifier utilizes the tuple (c, q, a) as input and is trained to predict the difficulty level d of the given questions. It is evident that our classifier outperforms GPT-4 under the difficulty estimation schema, with an accuracy exceeding 81%.

B.2 Details of DiffQA

The proportion distribution of d in train split and test split of DiffQA is visualized in Figure 3.

B.3 Details of Calculation

Consistent with the symbols used in the main text, $\mathcal{P} = \{p_i\}_{i=1}^{N}$ represents the QA pairs generated by the LLM. Let $d_t(\cdot)$ and $d_a(\cdot)$ represent the target difficulty and the actual difficulty (predicted by trained classifiers) of a QA pair, respectively.

Spearman correlation coefficient ρ can be calculated by the following formula:

$$\rho = 1 - \frac{6\sum_{i=1}^{N}(d_a(p_i) - d_t(p_i))^2}{N(N^2-1)} \tag{6}$$

We calculate ρ using the spearmanr function from the SciPy library.

Accuracy is computed as follows:

$$acc = \frac{\sum_{i=1}^{N} \mathbb{1}(d_a(p_i) = d_t(p_i))}{N} \tag{7}$$

where $\mathbb{1}(\cdot)$ is an indicator function.

C Experiments on GPT-4

In this section, we apply CrossQG to GPT-4 (Achiam et al., 2023), with results presented in Table 18. Although GPT-4 achieves superior performance in difficulty consistency compared to other evaluated LLMs, there remains significant room for improvement. CrossQG significantly enhances GPT-4's performance, indicating its potential to complement and elevate advanced LLMs in DCQG tasks.

D Implementation Details

In cross filtering, we utilize the commonly used Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024) model for difficulty-oriented embedding. For difficulty estimation, we fine-tune Roberta with the following parameter settings: learning rate = 1e-5;

Other Prompt Templates for Embedding Generation
PromptEOL+ID+FS: In this task, you're given a context and a question. Your task is to find the answer and describe how to get the answer. Context: [Context] Question: [Question] For this task, this sentence: "Given the question, the answer and its derivation are" means in one word:"
PromptSUM+ID+FS: In this task, you're given a context and a question. Your task is to find the answer and describe how to get the answer. Context: [Context] Question: [Question] For this task, this sentence: "Given the question, the answer and its derivation are" can be summarized as
PromptSTH+DD+LI: In this task, you're given a context and a question. Your task is to assess the difficulty of the question based on the following difficulty definitions. Difficulty Definition: 1. Easy: Answers must be directly found in the text; getting the answer should require focusing on the local information (e.g. one single sentence) in the context. 2. Medium: Case 1: Answers cannot be directly found in the text; getting the answer requires focusing on the local information (e.g. one single sentence) in the context. Case 2: Answers can be found directly in the text; obtaining the answer should involve synthesizing and summarizing information from multiple parts of the context. 3. Hard: Answers cannot be directly found in the text; obtaining the answer should involve synthesizing and summarizing information from multiple parts of the context. For this task, this sentence: "Context: [context] Question: [question]" means something
PromptSTH+ID+LI: In this task, you're given a context and a question. Your task is to find the answer and describe how to get the answer. For this task, this sentence: "Context: [context] Question: [question]" means something
PromptSTH+DD+FS: In this task, you're given a context and a question. Your task is to assess the difficulty of the question based on the following difficulty definitions. Difficulty Definition: 1. Easy: Answers must be directly found in the text; getting the answer should require focusing on the local information (e.g. one single sentence) in the context. 2. Medium: Case 1: Answers cannot be directly found in the text; getting the answer requires focusing on the local information (e.g. one single sentence) in the context. Case 2: Answers can be found directly in the text; obtaining the answer should involve synthesizing and summarizing information from multiple parts of the context. 3. Hard: Answers cannot be directly found in the text; obtaining the answer should involve synthesizing and summarizing information from multiple parts of the context. Context: [Context] Question: [Question] For this task, this sentence: "Given the question, the answer and its derivation are" means something

Table 14: Other prompt templates applied in difficulty-oriented embedding generation. In the template, [Context] and [Question] are special tokens and need to be substituted.

Prompt Template for ICL
<s>[INST] Your task is to generate pairs of questions and answers according to the following context, meeting all the requirements. Context: [Context] Requirements: 1. [Difficulty Definition] 2. Answers must be clear, concrete, and well-justified based on the context. [Supplement] Here are several examples. [Example 1] [Example 2] ... Output Format: - Q1: {question} - A1: {answer} - Q2: {question} - A2: {answer} - Continue as needed... [/INST]

Table 15: Prompt template used for ICL. In the template, [Context], [Difficulty Definition], [Supplement] and [Example i] tokens need to be substituted.

Prompt Template for SFT
<s>[INST] Your task is to generate a pair of question and answer according to the following context, meeting all the requirements. Context: [Context] Requirements: 1. [Difficulty Definition] 2. Answer must be clear, concrete, and well-justified based on the context. [Supplement] Output Format: - Q: {question} - A: {answer} [/INST]

Table 16: Prompt template used for SFT. In the template, [Context], [Difficulty Definition] and [Supplement] tokens need to be substituted.

Model	acc
GPT-4 (zero-shot)	0.548
GPT-4 (few-shot)	0.619
Roberta (Ours)	**0.817**

Table 17: Accuracy results on the test split of DiffQA.

Method	SQuAD		RACE		FairytaleQA	
	ρ	acc	ρ	acc	ρ	acc
Prompt	0.6046	0.5357	0.5857	0.5676	0.5587	0.5688
ICL	0.6304	0.5631	0.6542	0.5935	0.5644	0.5701
Self-refine	0.7494	0.6077	0.7022	0.6333	0.6005	0.5973
CrossQG (ours)	0.7672	0.6255	**0.7345**	**0.6651**	0.6371	**0.6257**
CrossQG-fast (ours)	**0.7888**	**0.6335**	0.7311	0.6583	**0.6454**	0.6197

Table 18: Experimental results on GPT-4, with ρ and acc for each method. Best results are highlighted in bold.

batch size = 32; and epoch = 5. In the main experiments, the prompt templates used for ICL and SFT are presented in Appendix A.4. The prompt template employed in the self-refine method is identical to that in the CE component, but the approach to selecting negative examples differs. When fine-tuning the LLM, the hyperparameters are as follows: learning rate = 1e-5; batch size per device = 8; and epoch = 3.

Our code is implemented based on Huggingface (Wolf et al., 2020), whereas AdamW (Loshchilov and Hutter, 2019) is used for optimization. All LLMs are loaded and used for inference on 1 Nvidia-A100-40G GPU and trained on 8 Nvidia-A100-40G GPUs. For each configuration of our method and all compared methods, we conduct 5 independent runs and report the average score.

E Case Study

Table 19 illustrates a complete example of question generation using our CrossQG method. In the initial question generation, the difficulty consistency of the generated questions is poor. The questions expected to be of medium difficulty all turn out to be easy. After applying contrast enhancement (CE), the difficulty consistency of the regenerated questions is significantly improved. Specifically, questions targeted at an easy difficulty level consistently match expectations. In addition, generated questions at other difficulties also achieve better difficulty consistency than before. Then, following cross filtering (CF), questions highlighted in yellow are removed due to high difficulty similarity. The operation results in an insufficient number of medium difficulty questions, making it necessary for LLM to regenerate them. Finally, it is evident that the final set of generated questions aligns well with the expected difficulty levels.

F List of Software and Data Licences Used in this Work

Main dependencies in this paper are as follows. They are all public and free for research use.

- 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.
- Pytorch: https://github.com/pytorch/pytorch/blob/main/LICENSE, Misc.
- 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.
- Llama 3: https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/LICENSE, under the LLAMA 3.1 Community License.
- SciPy: https://github.com/scipy/scipy, under the BSD-3-Clause license.

Context: The investors showed little interest in Tesla's ideas for new types of motors and electrical transmission equipment and also seemed to think it was better to develop an electrical utility than invent new systems. They eventually forced Tesla out leaving him penniless. He even lost control of the patents he had generated since he had assigned them to the company in lieu of stock. He had to work at various electrical repair jobs and even as a ditch digger for $2 per day. Tesla considered the winter of 1886/1887 as a time of "terrible headaches and bitter tears." During this time, he questioned the value of his education.		
Initial Question Generation: Q1: Why did the investors think it was better to develop an electrical utility than invent new systems? (*Medium*) Q2: What did Tesla consider the winter of 1886/1887 as in relation to his education? (*Easy*) Q3: What happened to Tesla after he was forced out of the company? (*Easy*) Q4: How did Tesla feel about his situation during this time? (*Hard*)	Q1: What was the reason why the investors showed little interest in Tesla's ideas for new types of motors and electrical transmission equipment? (*Easy*) Q2: What did Tesla have to do to survive after he was forced out of the company? (*Easy*) Q3: What was the outcome of Tesla's decision to assign his patents to the company in lieu of stock? (*Easy*) Q4: What did Tesla consider the winter of 1886/1887 as in relation to his education? (*Easy*)	Q1: How did Tesla feel about the value of his education during the winter of 1886/1887? (*Medium*) Q2: What was Tesla's job at the time of the winter of 1886/1887? (*Easy*) Q3: What did Tesla learn from his experiences during the winter of 1886/1887? (*Hard*) Q4: Why did Tesla assign his patents to the company in lieu of stock? (*Medium*)
Negative Examples: Q1: What did Tesla have to do to survive after he was forced out of the company? Q2: What was Tesla's job at the time of the winter of 1886/1887?	Q1: Why did the investors think it was better to develop an electrical utility than invent new systems? Q2: What did Tesla learn from his experiences during the winter of 1886/1887?	Q1: What did Tesla consider the winter of 1886/1887 as in relation to his education? Q2: How did Tesla feel about the value of his education during the winter of 1886/1887?
Contrast Enhancement: Q1: What did the investors not show interest in? (*Easy*) Q2: Why did Tesla lose control of his patents? (*Easy*) Q3: What was Tesla's job during the winter of 1886/1887? (*Easy*) Q4: How much did Tesla earn from his job as an electrical repairman? (*Easy*)	Q1: What were some of the challenges that Tesla faced in trying to convince investors to fund his ideas for new types of motors and electrical transmission equipment? (*Hard*) Q2: How did Tesla's experiences during the winter of 1886/1887 affect his view of the value of his education? (*Medium*)	Q1: What were the investors' reasons for not showing interest in Tesla's new motor and electrical transmission ideas, and how did this impact Tesla's decision-making? (*Medium*) Q2: How did Tesla's experience working as a ditch digger impact his perspective on the value of his education? (*Medium*) Q3: How did Tesla's relationships with his colleagues and superiors at the company impact his experience and decision-making during this time, and what lessons did he learn from these interactions? (*Hard*)
Cross Filtering: Q1: What did the investors not show interest in? (*Easy*) Q2: Why did Tesla lose control of his patents? (*Easy*) Q3: What was Tesla's job during the winter of 1886/1887? (*Easy*) Q4: How much did Tesla earn from his job as an electrical repairman? (*Easy*)	Q1: How did Tesla feel about the situation he was in after being forced out of his company? (*Medium*) Q2: What were some of the personal and financial challenges that Tesla faced in the winter of 1886-1887? (*Medium*)	Q1: What were the investors' reasons for not showing interest in Tesla's new motor and electrical transmission ideas, and how did this impact Tesla's decision-making? (*Medium*) Q2: How did Tesla's relationships with his colleagues and superiors at the company impact his experience and decision-making during this time, and what lessons did he learn from these interactions? (*Hard*)

Table 19: A complete example of question generation using CrossQG, with corresponding answers omitted. In the table, the target difficulty levels of the questions in the three columns from left to right are easy, medium, and hard, respectively. The difficulty similarity between the two questions highlighted in yellow is relatively high.