JudgeAgent: Beyond Static Benchmarks for Knowledge-Driven and Dynamic LLM Evaluation

Zhichao Shi[1,2,3,4*], **Xuhui Jiang**[1,2*], **Chengjin Xu**[1,2], **Cangli Yao**[1,2], **Shengjia Ma**[1,2], **Yinghan Shen**[4], **Zixuan Li**[4], **Jian Guo**[2], **Yuanzhuo Wang**[4†],
[1]DataArc Tech Ltd. [2]IDEA Research, International Digital Economy Academy
[3]School of Advanced Interdisciplinary Sciences, UCAS
[4]State Key Lab of AI Safety, Institute of Computing Technology, CAS

Abstract

Current evaluation methods for large language models (LLMs) primarily rely on static benchmarks, presenting two major challenges: limited knowledge coverage and fixed difficulties that mismatch with the evaluated LLMs. These limitations lead to superficial assessments of LLM knowledge, thereby impeding the targeted model optimizations. To bridge this gap, we propose JudgeAgent, a knowledge-driven and dynamic evaluation framework for LLMs. To address the challenge of limited knowledge coverage, JudgeAgent leverages LLM agents equipped with context graphs to traverse knowledge structures systematically for question generation. Furthermore, to mitigate data contamination and difficulty mismatch, it adopts a difficulty-adaptive and multi-turn interview mechanism. Thereby, JudgeAgent can achieve comprehensive evaluations and facilitate more effective improvement of LLMs. Empirical results demonstrate that JudgeAgent enables more comprehensive evaluations and facilitates effective model iterations, highlighting the potential of this knowledge-driven and dynamic evaluation paradigm. The source code is available on https://github.com/DataArcTech/JudgeAgent.

1 Introduction

Evaluating Large Language Models (LLMs) to verify their knowledge is a crucial step for their successful application across various domains (Tang et al., 2024; Yuan et al., 2023; Shi, 2024; Wang et al., 2025). Current evaluation methods mainly rely on static benchmarks (Clark et al., 2018; Hendrycks et al., 2021; Huang et al., 2023; Lin et al., 2022; Cobbe et al., 2021; Tang and Yang, 2024), where LLMs are evaluated by their performance on answering predefined static questions. In the early stages, benefiting from the controllable question quality and rapid workflows of static benchmarking, developers can quickly grasp the core competencies of evolving LLMs, thereby accelerating iteration cycles. However, due to the static nature (Gu et al., 2024; Ko et al., 2024), these benchmarks have become saturated more and more quickly in recent years. For example, it took 3 years for MMLU (Hendrycks et al., 2020) to reach an 80% accuracy record, while GPQA (Rein et al., 2023) achieved the same record just within one year. The static nature restricts the evaluations within a predefined and limited knowledge scope (Wang et al., 2023a; Kwan et al., 2024), and increases the risk of data contamination (Schaeffer, 2023; Oren et al., 2023), resulting in LLMs being misapplied in scenarios out of their knowledge.

Consequently, researchers attempted to modify static benchmarks using LLMs (Bai et al., 2023; Shi et al., 2025). However, they still face two major challenges. **(1) Limited knowledge coverage**: Constrained by the evaluator LLM's knowledge limitations and inherent randomness, the knowledge extended by the LLM from static benchmarks remains uncontrollable and limited, making it difficult to achieve comprehensive knowledge evaluations. **(2) Mismatch between question difficulty and the evaluated LLMs**: Current mainstream methods lack adaptive adjustments to question difficulty. Thus, the question difficulty may significantly deviate from the actual knowledge mastery of the evaluated LLMs, making the evaluations harder to precisely reflect the model's capabilities. Thus, a new knowledge-driven and dynamic evaluation framework is needed.

In this paper, we propose JudgeAgent, a knowledge-driven and dynamic evaluation framework for LLMs. To address the challenge of limited knowledge coverage, JudgeAgent leverages LLM agents with context graphs to traverse knowledge structures at greater breadth and depth.

[*]Both authors contributed equally to this research.
[†]Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 13004–13030
July 2-7, 2026 ©2026 Association for Computational Linguistics

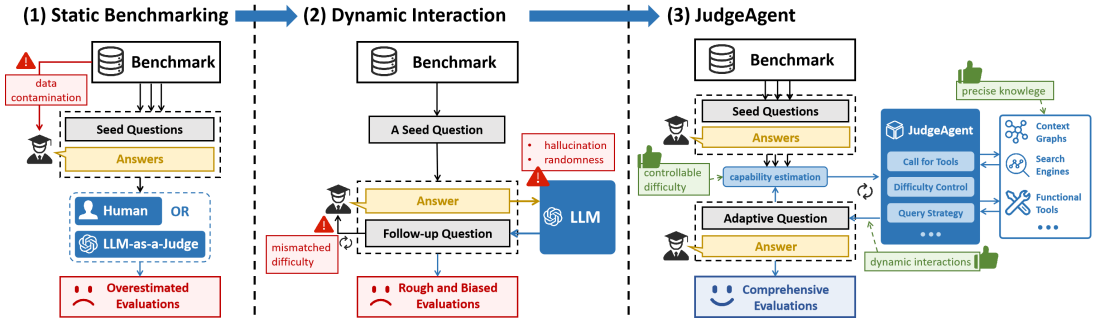


Figure 1: The difference between JudgeAgent and current evaluation paradigms.

Through graph sampling strategies and knowledge-driven data synthesis, JudgeAgent generates questions with broader and deeper knowledge coverage, reducing hallucinations with reliable knowledge sources, and further exploring potential knowledge deficiencies in evaluated LLMs. Furthermore, to mitigate data contamination and difficulty mismatch, JudgeAgent dynamically adjusts the capability estimation and generates difficulty-adaptive questions based on the target's responses in multi-turn interviews. Thereby, JudgeAgent can provide comprehensive evaluations and facilitate more effective subsequent improvement of evaluated LLMs.

Extensive experiments and analysis validate that JudgeAgent can provide effective and precise evaluation results. Our in-depth analysis also reveals the significant potential of this evaluation paradigm. It offers valuable insights into how additional knowledge-driven and dynamic strategies for evaluator LLMs can be designed, which enhance the quality of dynamic evaluation results.

In summary, our contributions are as follows:

- We introduce JudgeAgent, a knowledge-driven and dynamic evaluation framework for LLMs. JudgeAgent leverages LLM agents with context graphs to traverse knowledge structures and perform dynamic multi-turn interviews, achieving comprehensive knowledge evaluations.
- We propose a difficulty-adaptive and multi-turn evaluation mechanism that simulates an interactive interview, enabling the evaluations to more precisely reflect the knowledge mastery of the evaluated LLM.
- We conduct extensive experiments to validate the effectiveness of JudgeAgent and highlight its significant potential for the dynamic evaluation paradigm.

2 Related Works

2.1 Static Benchmark-based Evaluation

These methods employ pre-constructed benchmarks to evaluate LLMs, using formats such as multiple-choice, question-answer(Q&A), or prompts for performing tasks. For multiple-choice (Clark et al., 2018; Hendrycks et al., 2021; Huang et al., 2023) or Q&A (Lin et al., 2022; Cobbe et al., 2021; Tang and Yang, 2024), the benchmark provides correct answers and assesses the target by the accuracy. For task-execution format, the benchmark measures the model by the metrics assessed by human (Chang et al., 2024) or LLM-as-a-judge (Wang et al., 2023b; Zheng et al., 2023). These methods ensure controllable question quality, but they are also susceptible to data contamination, which undermines the validity of evaluations.

2.2 Dynamic LLM-based Evaluation

With the deepening application of LLMs, in addition to mainstream static methods, recent researchers have begun to leverage LLMs to dynamically generate evaluation questions. Bai et al. (2023) employs LLMs as examiners to generate questions based on Google Trends categories. SafetyQuizzer (Shi et al., 2025) formulates questions with current events from search engines to maximize the timeliness. KIEval (Yu et al., 2024), and LLM-as-an-Interviewer (Kim et al., 2025) leverage LLMs to generate follow-up questions based on the target's responses in multi-turn interactions.

Compared to static methods, these dynamic methods address the challenge of data contamination and provide more genuine evaluations. However, the generations of these methods rely mainly on the LLM, resulting in limited knowledge coverage and pool quality control. Moreover, these methods lack adaptive adjustments of question difficulty, resulting in a deviation between the evaluations and the actual capabilities of target LLMs.

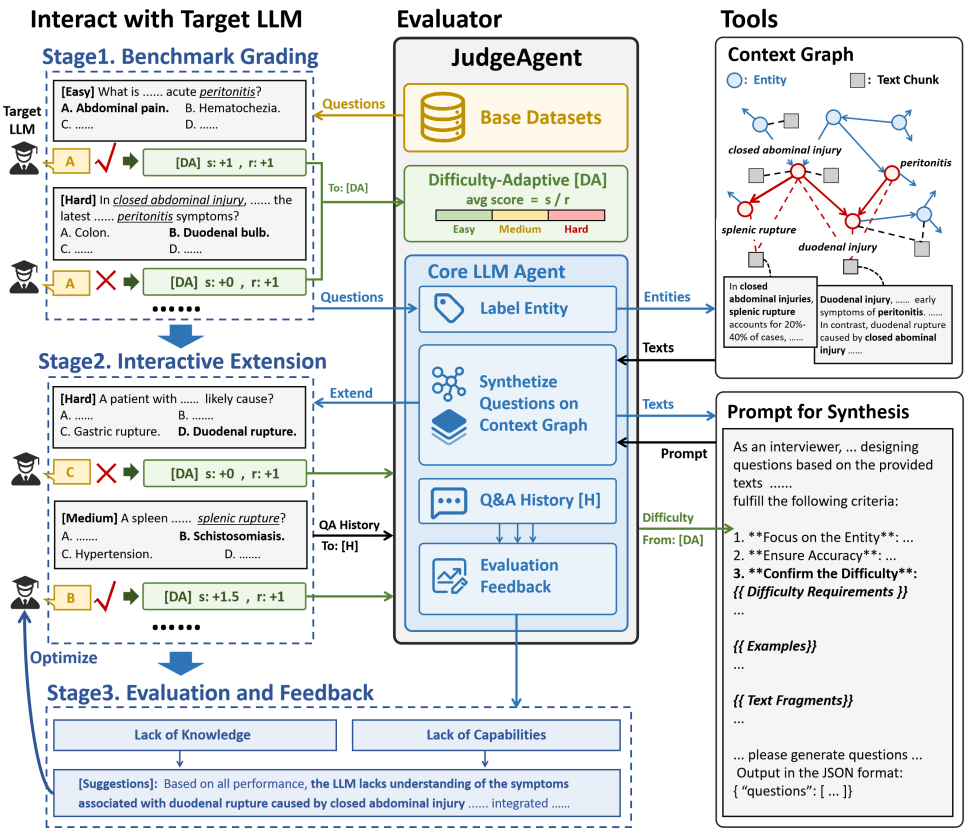


Figure 2: The framework of JudgeAgent. The left part is the interaction process. The central part is the composition of JudgeAgent. The right part presents the tools of JudgeAgent.

This mismatch leads to biased subsequent optimization efforts and reduced efficiency.

Thus, we propose a knowledge-driven and dynamic evaluation framework named JudgeAgent. To expand knowledge coverage, JudgeAgent leverages LLM agents with context graphs to traverse knowledge structures and generate questions. To mitigate data contamination and difficulty mismatch, JudgeAgent adopts a difficulty-adaptive multi-turn interview mechanism to conduct dynamic evaluations. Thus, JudgeAgent provides comprehensive evaluations and facilitates subsequent improvement of LLMs.

3 Methodology

We introduce JudgeAgent, a knowledge-driven dynamic evaluation framework, which utilizes LLM agents with context graphs to simulate the entire interview of target LLMs. The workflow comprises three core components: **(1) Benchmark Grading**: Fundamentally assessing target LLMs through testing on public static benchmarks. **(2) Interactive Extension**: Using knowledge-driven synthesis to dynamically generate follow-up questions and update the difficulty based on the estimations of target LLMs. **(3) Evaluation Feedback**: Evaluating the target's deficiencies in multiple dimensions and providing actionable suggestions. The overall framework is illustrated in Figure 2, and detailed as pseudo code in Algorithm 1.

3.1 Benchmark Grading

To address the challenge of difficulty mismatch in prior dynamic evaluation, JudgeAgent first evaluates the target LLM on public static benchmarks and obtains a base score through a difficulty scoring mechanism. Similar to a written test before an interview, this allows JudgeAgent to form a basic capability estimation of the target LLM.

JudgeAgent first partitions the benchmark questions into batches. Then it prompts the target LLM to answer the questions batch by batch. After assessing the responses, JudgeAgent estimates the target's capability based on its performance in each batch. Inspired by the common real-world practice of scoring student exams, we design a linear difficulty control mechanism. Within each batch, each correct answer improves the total score, and the average score determines the follow-up question's difficulty. Let $[d_1, d_2, d_3]$ respectively denote the difficulty *Easy*, *Medium*, and *Hard*, for the score

gain c_{it} for a correct answer at different difficulty levels d_i, we set $[c_{1t}, c_{2t}, c_{3t}] = [1, 1.5, 2]$. And for the threshold t_{ij} between d_i and d_j, we set $[t_{12}, t_{23}] = [0.5, 1]$. Once the dynamic average score exceeds the threshold t_{ij}, the difficulty of the next generated question will be set to d_j. The mechanism's detailed design, analysis, and proofs are available in Appendix B.

3.2 Interactive Extension

This process is an "interview" where JudgeAgent conducts a knowledge-driven and difficulty-adaptive multi-turn evaluation. Specifically, JudgeAgent dynamically adjusts the difficulty and generates questions with expanded knowledge based on the responses. Furthermore, JudgeAgent explores the potential knowledge gaps of target LLMs through a multi-turn interaction process. Each iteration in this process consists of three steps: Relevant Knowledge Retrieval, Difficulty-Adaptive Question Generation, and Capability Estimation.

Relevant Knowledge Retrieval: To achieve knowledge-driven evaluation, JudgeAgent invokes external knowledge and constructs context graphs to facilitate the association and extension of knowledge during evaluation. If only reference texts of seed questions are used for generation, the generated questions will be highly similar to the original ones. Inspired by SoG (Jiang et al., 2025), JudgeAgent employs a context-graph–based sampling approach to get background texts. The context graph is constructed from the benchmark's knowledge base. The texts in the knowledge base are sampled from domain textbooks or are just concatenated by the evidence list in the benchmark. We extract entities from these text chunks as graph nodes. Each entity node has bidirectional links with its source chunk to facilitate retrieval and tracking. The entities in the same text chunk will be linked as neighbors on the graph. Benefiting from the broad application of context graphs across domains such as medical (Nguyen et al., 2022; Xie et al., 2025), education (Yang et al., 2023), and finance (Li, 2023), JudgeAgent, equipped with context graphs, also has the potential to be widely applied to evaluation tasks in various knowledge-intensive domains. The detailed construction is illustrated in Appendix C and Algorithm 3.

During sampling, JudgeAgent first extracts entities from seed questions and finds the most similar entity e on the graph. In each hop, JudgeAgent retrieves the most relevant chunks of e to the question, and randomly selects a chunk c. e is the seed entity of the next hop, and c is added to the path. After N hops, we sample a path with breadth and depth, and the chunks on the path are concatenated to form the knowledge for generation. The context graph ensures both knowledge breadth and depth, and the greedy similarity strategy ensures relevance to seed questions. Entity extraction is performed by GPT-4.1, with prompts detailed in Appendix G.

Difficulty-Adaptive Question Generation: To mitigate data contamination and difficulty mismatch, JudgeAgent conducts multi-turn interactive interviews to explore potential deficiencies and incorporates adaptive difficulty adjustments into question generation. If a fixed difficulty level is used, it can only indicate whether the model possesses the knowledge required for that specific level, but it cannot reveal the model's precise capability range. Therefore, evaluation questions should cover multiple difficulty levels. Existing benchmarks such as LawBench (Fei et al., 2024) and SciKnowEval (Feng et al., 2024) adopt similar difficulty categorizations, including memory/easy, comprehension/medium, and reasoning/hard. However, generating questions for all difficulty levels is costly in knowledge expansion scenarios. To balance efficiency and precision, JudgeAgent dynamically adjusts the follow-up question's difficulty based on capability estimation through the difficulty control mechanism in Section 3.1.

Based on the capability estimation, JudgeAgent adjusts the question difficulty as *Easy*, *Medium*, or *Hard*, and explicitly specifies each difficulty's requirements in the prompt. For *Easy*, the focus is on assessing information retrieval skills, primarily through cloze-style questions based on the original text, like *"Which virus primarily causes HFMD?"*. For *Medium*, the emphasis is on understanding key concepts, with questions involving basic inference, like *"Which description is correct regarding meiosis II?"* For *Hard*, the questions are designed to encourage deep thinking and complex logical analysis, like *"A patient ingested a toxic substance... The doctor employed... which type of treatment is likely lacking?"*. Detailed prompt designs are provided in Appendix G. And the validation of question quality, including the factuality, diversity, and the actual difficulty, is described and analyzed in Appendix E.4.

Capability Estimation: JudgeAgent queries the target LLM with generated questions. Based on LLM's responses, JudgeAgent computes average

difficulty scores with the strategy in Section 3.1, and determines the difficulty for the next round's generation. Through this adaptive multi-turn mechanism, JudgeAgent progressively aligns the assessment with the true capabilities of the target LLM, thereby achieving more precise evaluations.

3.3 Evaluation Feedback

Inspired by Generative Reward Model methods (Ankner et al., 2024; Ye et al., 2024; Li et al., 2025) and the evaluation report method in (Kim et al., 2025), JudgeAgent generates a text evaluation report based on all Q&A history for each batch. Benefiting from context graphs, the evaluation report can identify the knowledge deficiencies of target LLMs and provide concise and valuable feedback around key knowledge concepts.

4 Experiments

In this section, we validate the effectiveness of JudgeAgent using the method in Section 3.3, which is detailed in Algorithm 2. The following research questions guide our experiments: **RQ1**, does JudgeAgent's knowledge-driven evaluation genuinely discover the shortcomings of the target model? **RQ2**, is JudgeAgent's dynamic evaluation more effective and precise than the static benchmarking? **RQ3**, to what extent does each mechanism in JudgeAgent influence the evaluations?

Current LLM knowledge evaluation methods lack quantifiable and standardized validation approaches. Therefore, to validate the effectiveness of JudgeAgent, we use a similar approach as (Kim et al., 2025). We use the evaluation report produced by JudgeAgent as suggestions to prompt the target LLM to answer the same questions again. Then we compare the accuracy before and after the intervention to indirectly validate the effectiveness. To avoid cheating, we produce suggestions without directly providing the correct answer or background knowledge of seed questions to JudgeAgent.

Experiment Setup. We select GPT-4.1[1] to be the core LLM. In our experiments, the batch size in Benchmark Grading is 3, and the Interactive Extension is limited to a maximum of 3 rounds.

4.1 Dataset and Target Model Selection

In our experiments, we select MedQA (Jin et al., 2021), MultiHop-RAG (Tang and Yang, 2024), and QuALITY (Pang et al., 2022) as the initial benchmarks. We remove the background knowledge of questions from MedQA and MultiHop-RAG to evaluate the target's knowledge rather than its comprehension ability, and to verify whether JudgeAgent can discover the target's actual knowledge deficiencies. We use QuALITY to validate the effectiveness in guiding comprehension and reasoning. Detailed information and the preprocessing procedures are in Appendix D.

For target LLMs, we select Qwen3 (Yang et al., 2025), GLM4-Flash (GLM et al., 2024), GPT-4.1 (Achiam et al., 2023), and gemini-2.5-pro (Comanici et al., 2025)[2]. We utilize the official APIs to interact with these models.

4.2 Evaluation Metrics

The metrics used in our experiments are as follows:
(1) **Accuracy (ACC).** We use this metric to measure the performance of the target model on base datasets. To investigate the effectiveness of JudgeAgent, we compared the changes in the ACC before and after dynamic evaluation. ACC1 represents the target's ACC in answering base questions in *Benchmark Grading*, measuring its performance before receiving evaluation feedback. ACC2 denotes the ACC in answering the same questions after receiving feedback from JudgeAgent.
(2) **Correction Rate (CR).** This metric quantifies the proportion of questions that the target model initially answered incorrectly but subsequently answered correctly after receiving evaluation suggestions, which measures the effectiveness of JudgeAgent. A higher correction rate indicates better performance of the suggestions.
(3) **Correct-to-Error Rate (CtE).** This metric serves as the inverse of the Correction Rate. A lower CtE indicates greater effectiveness.

4.3 Main Results and Analysis

To address **RQ1**, we conduct experiments on three datasets to validate the effectiveness of JudgeAgent's evaluation, and the results are shown in Table 1 and Table 2.

Based on the results from MedQA and MultiHopRAG in Table 1, JudgeAgent can effectively identify potential knowledge deficiencies in target LLMs and subsequently mitigate these deficiencies by providing targeted suggestions to the target

[1]We use gpt-4.1-2025-04-14 from OpenAI's official API for all experiments

[2]We use qwen-plus-2025-04-28, gpt-4.1-2025-04-14, gemini-2.5-pro-preview-06-05, and the free version glm-4-flash-250414 as the target models.

Target Model	MedQA				MultiHopRAG			
	ACC1	ACC2	CR↑	CtE↓	ACC1	ACC2	CR↑	CtE↓
Qwen3	91.71	96.38	5.02	0.35	63.65	70.07	17.49	11.07
GLM4-Flash	80.09	92.82	13.46	0.73	51.25	65.92	24.26	9.59
GPT-4.1	84.97	92.44	7.65	0.18	68.94	75.55	12.36	5.75
Gemini-2.5-pro	91.04	94.60	4.03	0.47	62.25	71.83	15.41	5.83

Table 1: The results on MedQA and MultiHopRAG, and all values are percentages.

Target Model	QuALITY-overall				QuALITY-easy			
	ACC1	ACC2	CR↑	CtE↓	ACC1	ACC2	CR↑	CtE↓
Qwen3	87.83	88.84	1.88	0.87	94.63	95.61	0.98	0.00
GLM4-Flash	73.48	76.38	4.78	1.88	83.26	84.65	4.19	2.79
GPT-4.1	89.13	92.75	3.91	0.29	93.66	96.10	2.44	0.00
Gemini-2.5-pro	93.77	96.38	3.19	0.58	97.55	99.02	1.47	0.00

Target Model	QuALITY-medium				QuALITY-hard			
	ACC1	ACC2	CR↑	CtE↓	ACC1	ACC2	CR↑	CtE↓
Qwen3	88.53	88.89	1.43	1.08	80.10	82.04	3.40	1.46
GLM4-Flash	74.35	79.18	5.58	0.74	62.14	64.08	4.37	2.43
GPT-4.1	91.04	94.27	3.58	0.36	82.04	87.38	5.83	0.49
Gemini-2.5-pro	93.57	95.36	2.50	0.71	90.29	95.15	5.83	0.97

Table 2: The results on QuALITY with different difficulty levels. QuALITY-X refers to the subdataset that consists of questions with specific difficulty, and -overall refers to all questions.

LLM. Furthermore, as evidenced by the overall performance on QuALITY in Table 2, JudgeAgent also contributes to providing further guidance to address potential shortcomings in the logical reasoning abilities of the target LLM, thereby assisting in refining the target's thinking steps. Additionally, by comparing CR and CtE of different LLMs before and after receiving suggestions, it can be observed that the effectiveness of suggestions is less consistent for relatively weaker LLMs (e.g., the free model GLM4-Flash), as reflected in the higher CtE. In contrast, stronger LLMs are less susceptible to misleading suggestions.

Based on the performance across different difficulties of QuALITY in Table 2, we can analyze the effectiveness of JudgeAgent for different targets on various difficulty levels. For stronger LLMs (Qwen3, GPT-4.1, and Gemini-2.5-pro), JudgeAgent provides stronger guidance, particularly on questions of greater difficulty (*Medium* and *Hard*), since these LLMs may have already mastered the basic knowledge of *Easy* questions. In contrast, for weaker LLMs, JudgeAgent leads to considerable improvement across all difficulty levels, with particularly notable gains on *Easy* level, since JudgeAgent's feedback is more effective at filling gaps in basic concepts. These results further indicate that JudgeAgent more accurately assesses the capability and provides more difficulty-adaptive guidance to the target LLMs. Furthermore, for all target LLMs, JudgeAgent's suggestions demonstrate clear guidance on *Hard* questions, indicating that JudgeAgent effectively identifies underlying deficiencies in target LLMs through dynamic interactive evaluation.

In summary, JudgeAgent can effectively and precisely identify potential knowledge deficiencies in target LLMs, offering more refined evaluation results. Furthermore, we conducted cross-validation experiments to verify that the JudgeAgent's evaluations are effective not only for seed questions, which are detailed in Appendix E.5. Moreover, we also compare with a direct method that prompts models for self-correction, as shown in Appendix E.6. The comparison confirms that JudgeAgent, guided by context graphs, can specifically identify the deficiencies of the evaluated model, rather than merely activating its self-correction ability. Additionally, experiments on the efficiency and stability of JudgeAgent are detailed in Appendix E.1 and E.2.

4.4 Analysis of JudgeAgent's Dynamic Scores

To address **RQ2**, we analyze the dynamic difficulty scores and the difficulty distribution of generated questions during JudgeAgent's Interactive Expansion stage. The results are shown in Table 3.

As the results on MedQA show, there is no signif-

Benchmark	Target Model	ACC(%)	Difficulty Score				Difficulty Distribution(%)		
			base	@1	@2	@3	easy	medium	hard
MedQA	GLM4-Flash	80.09	1.201	1.230	1.217	1.192	20.86	12.64	66.50
	GPT-4.1	84.97	1.275	1.388	1.395	1.401	14.61	9.02	76.38
	Qwen3	91.71	1.376	1.472	1.496	1.487	8.01	8.58	83.42
	Gemini-2.5-pro	91.04	1.366	1.484	1.494	1.505	9.41	7.76	82.84
MultiHopRAG	GLM4-Flash	51.25	0.769	0.587	0.535	0.486	34.06	48.13	17.81
	GPT-4.1	68.94	1.034	1.006	1.000	0.996	11.55	40.16	48.29
	Qwen3	63.65	0.955	0.801	0.767	0.746	19.62	48.91	31.47
	Gemini-2.5-pro	62.25	0.934	0.967	1.005	1.019	13.63	38.80	47.57
QuALITY	GLM4-Flash	74.35	1.065	1.143	1.232	1.304	12.09	39.30	48.60
	GPT-4.1	91.04	1.320	1.428	1.522	1.603	1.86	23.49	74.65
	Qwen3	88.53	1.296	1.388	1.466	1.520	3.26	25.58	71.76
	Gemini-2.5-pro	93.57	1.397	1.502	1.595	1.686	0.00	17.91	82.09

Table 3: The dynamic difficulty scores and difficulty distribution of questions generated by JudgeAgent. *base* means the score for base static benchmark questions, and @K means the scores for extended questions at the K-th iteration.

icant difference in the accuracy and scores between GPT-4.1 and GLM4-Flash on the static benchmark (*base*). However, through multi-round dynamic evaluation, it can be observed that the difficulty of questions queried to GPT-4.1 continuously increases, while that for GLM4-Flash decreases, indicating that GPT-4.1's actual knowledge mastery exceeds the preset difficulty of the static benchmark, further suggesting that GPT-4.1's knowledge mastery is far higher than that of GLM4-Flash.

Similarly, on MultiHopRAG, Qwen3 and Gemini-2.5-pro show comparable performance on the static benchmark. While in dynamic evaluations, Qwen3's performance is significantly lower than that of GPT-4.1 and Gemini-2.5-pro, with the gap progressively widening over rounds. This gap is also reflected in the fact that the generated questions for Qwen3 are primarily at the *Medium* level. Meanwhile, although GPT-4.1 performs noticeably better than Gemini-2.5-pro on the static benchmark, their performance was comparable in dynamic evaluations, as can be seen from their similar difficulty distributions in the questions generated.

Benchmark	base	@1	@2	@3
MedQA	0.005	0.010	0.013	0.015
MultiHopRAG	0.009	0.027	0.038	0.047
QuALITY	0.015	0.018	0.018	0.020

Table 4: Variances of the dynamic difficulty scores for *base* and @K iterations.

Moreover, the multi-turn interview mechanism makes the differentiation in evaluation more pronounced. To more intuitively illustrate the improvement in discriminative power brought by JudgeAgent, we calculated the variance of the results between different models across benchmarks and rounds in Table 4. The results show that under the static setting (base), the variance is low, indicating that different models are difficult to distinguish. As the number of dynamic rounds increases, the variance also gradually increases. This trend is especially pronounced on MultiHopRAG, indicating that the JudgeAgent's ability to distinguish between different models is improving. In summary, compared to static benchmarking, JudgeAgent's dynamic evaluation can more precisely distinguish the actual knowledge mastery of different LLMs.

Furthermore, we validate JudgeAgent's resistance to data contamination through dynamic evaluation, with experiments and analysis detailed in Appendix E.3.

4.5 Ablation Study

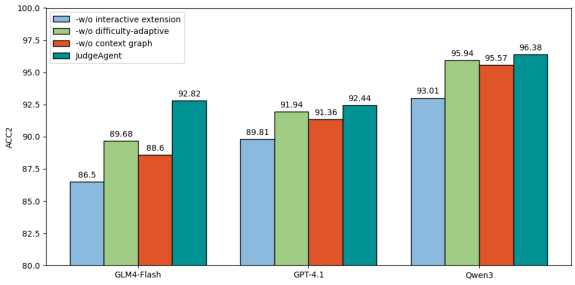


Figure 3: The results of the ablation study with MedQA as the base dataset, and all values are percentages.

To address **RQ3**, we conduct ablation studies on MedQA with GLM4-Flash as the target LLM. When maintaining the responses to seed questions unchanged, we removed different modules of JudgeAgent. By comparing the results under different ablation settings, we investigate how much

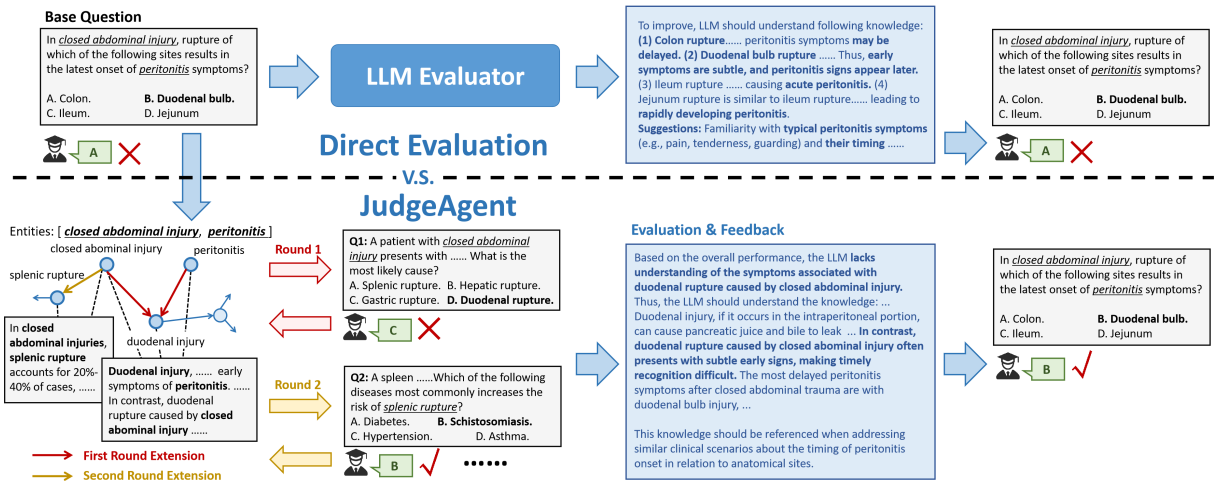


Figure 4: The brief overview of the comparative case in the Case Study.

different modules influence the evaluations The results are shown in Figure 3.

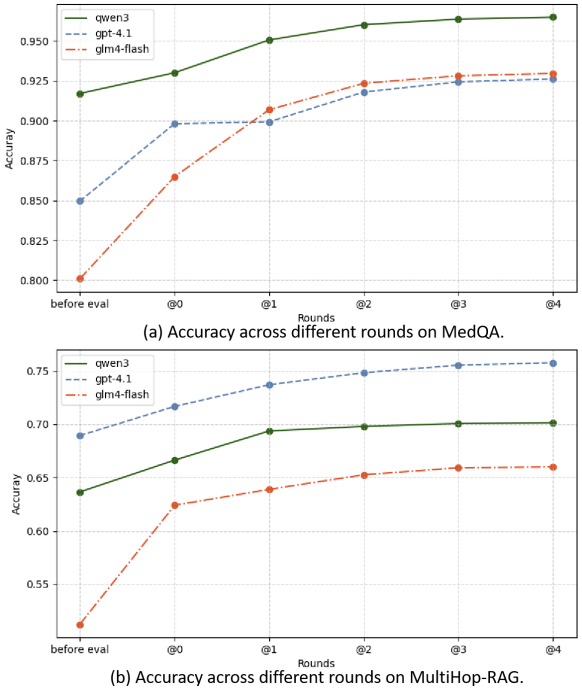
JudgeAgent (*w/o* context graph) removes the context graph and only uses chunks sampled randomly from the knowledge base. The results indicate that removing the context graph reduces the effectiveness of the JudgeAgent. The lack of context graph severs the knowledge link between extended questions and base ones, leading to feedback without accurate information, which may disrupt the thinking of target LLMs. The higher CtE compared to the setting -*w/o* difficulty-adaptive also provides supporting evidence for this potential interference.

JudgeAgent (*w/o* difficulty-adaptive) removes the difficulty-adaptive mechanism and generates questions with fixed difficulty rules in the prompt. The results show that the removal of this module diminishes the effectiveness of JudgeAgent's evaluations, demonstrating the importance of the difficulty-adaptive mechanism for providing effective evaluations.

JudgeAgent (*w/o* interactive extension) removes the Interactive Extension and only evaluates models with base benchmarks. The results show that removing this stage significantly weakens the effectiveness of JudgeAgent compared to other ablation settings, indicating that the dynamic expansion, which expands both breadth and depth of knowledge, is crucial to the evaluation's effectiveness.

4.6 Parameter Analysis

The number of extension rounds is crucial for balancing the efficiency and effectiveness of JudgeAgent. We test the JudgeAgent's effectiveness under different rounds in Interactive Extension, as shown in Figure 5. The results demonstrate



(a) Accuracy across different rounds on MedQA.

(b) Accuracy across different rounds on MultiHop-RAG.

Figure 5: The results of different expansion rounds on MedQA and MultiHop-RAG. @K indicates the ACC improvement after the K-th interaction.

that as the rounds increase, the JudgeAgent's effectiveness, which is represented by the accuracy improvement of target LLMs, also gradually improves. However, the trend gradually slows down, showing a relatively clear marginal effect.

As the rounds increase, JudgeAgent can expand more questions related to the knowledge around base questions. After reaching certain rounds, the Q&A history is sufficient to identify the target's knowledge deficiencies around base questions. Additional questions serve only as corroborative rather than critical evidence. There is a clear marginal effect in Figure 5. For example, the results of GPT-

4.1 on MultiHop-RAG, the improvement per round drops from 2.04% to 1.13%, then to 0.71%, and eventually to 0.23%, and other curves exhibit similar patterns. Therefore, JudgeAgent expands a maximum of 3 rounds in our experiments to avoid wasting resources and time.

Additionally, parameter analysis experiments regarding batch size are presented in Appendix E.7.

4.7 Case Study

To further understand JudgeAgent's mechanism, we analyze GLM4-Flash's responses to MedQA questions by receiving suggestions from direct evaluation and JudgeAgent. The seed questions where the target answers correctly are omitted in this case. A brief overview of the case is shown in Figure 4, and the detailed content is provided in Appendix H.

In this case, the target LLM answers the base question incorrectly. When directly evaluated, the evaluator LLM generically enumerated the potential consequences of closed abdominal injury. Consequently, even after receiving the feedback, the target LLM still outputs an incorrect answer.

In contrast, JudgeAgent first extracted two key entities, *closed abdominal injury* and *peritonitis*. Based on the sampled paths on the context graph, JudgeAgent generated extended questions, which help discover the lack of understanding of *duodenal injury* and *closed abdominal injury*. Ultimately, the target LLM successfully answered the original question correctly with this feedback.

5 Conclusion

In this paper, we propose JudgeAgent, a knowledge-driven and dynamic evaluation framework for LLMs. To address the challenge of limited knowledge coverage, JudgeAgent leverages LLM agents with context graphs to traverse knowledge structures and employs knowledge-driven synthesis for question generation. Furthermore, to mitigate data contamination and difficulty mismatch, JudgeAgent dynamically adjusts the capability estimation and generates difficulty-adaptive questions based on the target's responses in multi-turn interviews. Extensive experiments validate the effectiveness of JudgeAgent and highlight the significant potential of this knowledge-driven and dynamic evaluation paradigm in enhancing the evaluation results. In particular, when the benchmark is publicly available and may be incorporated into the training data of the evaluated LLMs, or when targeted iterative optimization of the models is required, JudgeAgent becomes a highly valuable option. In our future work, we will further refine this dynamic evaluation paradigm and develop more effective and reliable evaluation tools.

Limitations

In this paper, we validate JudgeAgent's effectiveness and its great potential for enhancing the dynamic evaluation paradigm of LLMs through extensive experiments. However, due to space constraints, we focus on proposing a knowledge-driven and dynamic evaluation framework, and do not delve deeply into methods for optimizing the evaluated LLMs based on JudgeAgent. In Section 4 Experiments, we only employ simple prompt injection optimization as a validation approach to verify the evaluation's effectiveness. In future work, we plan to further explore the potential of this evaluation paradigm, conducting in-depth research on how to perform targeted training and optimization of the evaluated LLMs based on JudgeAgent's evaluation results, thereby establishing an efficient and comprehensive iteration cycle for LLMs.

Ethical Considerations

In this study, all datasets used, including MedQA, MultiHop-RAG, and QuALITY, were sourced in compliance with relevant usage guidelines, ensuring no violation of privacy. No personally identifiable information was used, and no experiments were conducted that could raise privacy or security concerns. We are committed to maintaining transparency and integrity throughout the research process.

Acknowledgments

This research is funded by the Key Research and Development Project of Henan Province (No.241111211900).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. 2024. Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, and 1 others. 2023. Benchmarking foundation models with language-model-as-an-examiner. *Advances in Neural Information Processing Systems*, 36:78142–78167.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, and 1 others. 2024. Lawbench: Benchmarking legal knowledge of large language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 7933–7962.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. 2024. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010.

Xuhui Jiang, Shengjie Ma, Chengjin Xu, Cehao Yang, Liyu Zhang, and Jian Guo. 2025. Synthesize-on-graph: Knowledgeable synthetic data generation for continue pre-training of large language models. *arXiv preprint arXiv:2505.00979*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Eunsu Kim, Juyoung Suk, Seungone Kim, Niklas Muennighoff, Dongkwan Kim, and Alice Oh. 2025. LLM-as-an-interviewer: Beyond static testing through dynamic LLM evaluation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26456–26493, Vienna, Austria. Association for Computational Linguistics.

Miyoung Ko, Sue Park, Joonsuk Park, and Minjoon Seo. 2024. Hierarchical deconstruction of llm reasoning: A graph-based framework for analyzing knowledge utilization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4995–5027.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20153–20177.

Victor Xiaohui Li. 2023. Findkg: Dynamic knowledge graph with large language models for global finance. *Available at SSRN 4608445*.

Yafu Li, Xuyang Hu, Xiaoye Qu, Linjie Li, and Yu Cheng. 2025. Test-time preference optimization: On-the-fly alignment via iterative textual feedback. *arXiv preprint arXiv:2501.12895*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.

Vinh Nguyen, Hong Yung Yip, Goonmeet Bajaj, Thilini Wijesiriwardene, Vishesh Javangula, Srinivasan Parthasarathy, Amit Sheth, and Olivier Bodenreider. 2022. Context-enriched learning models for aligning biomedical vocabularies at scale in the umls metathesaurus. In *Proceedings of the ACM Web Conference 2022*, pages 1037–1046.

Yonatan Oren, Nicole Meister, Niladri S Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. 2023. Proving test set contamination in black-box language models. In *The Twelfth International Conference on Learning Representations*.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and 1 others. 2022. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Rylan Schaeffer. 2023. Pretraining on the test set is all you need. *arXiv preprint arXiv:2309.08632*.

Yi Shi. 2024. Drug development in the ai era: Alphafold 3 is coming! *The Innovation*, 5(5).

Zhichao Shi, Shaoling Jing, Yi Cheng, Hao Zhang, Yuanzhuo Wang, Jie Zhang, Huawei Shen, and Xueqi Cheng. 2025. Safetyquizzer: Timely and dynamic evaluation on the safety of llms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1733–1747.

Yi-Da Tang, Er-Dan Dong, and Wen Gao. 2024. Llms in medicine: The need for advanced evaluation systems for disruptive technologies. *The Innovation*, 5(3).

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023a. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, and 1 others. 2023b. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*.

Yilei Wang, Jiabao Zhao, Deniz S Ones, Liang He, and Xin Xu. 2025. Evaluating the ability of large language models to emulate personality. *Scientific reports*, 15(1):519.

Yuzhang Xie, Xu Han, Ran Xu, Xiao Hu, Jiaying Lu, and Carl Yang. 2025. Hypkg: Hypergraph-based knowledge graph contextualization for precision healthcare. In *International Semantic Web Conference*, pages 328–348. Springer.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang Yang, Chubing Zhang, Xin Song, Zheng Dong, Hengshu Zhu, and Wenjie Li. 2023. Contextualized knowledge graph embedding for explainable talent training course recommendation. *ACM Transactions on Information Systems*, 42(2):1–27.

Ziyi Ye, Xiangsheng Li, Qiuchi Li, Qingyao Ai, Yujia Zhou, Wei Shen, Dong Yan, and Yiqun Liu. 2024. Beyond scalar reward model: Learning generative judge from preference data. *arXiv preprint arXiv:2410.03742*.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and Shikun Zhang. 2024. Kieval: A knowledge-grounded interactive evaluation framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5967–5985.

J Yuan, P Bao, Z Chen, M Yuan, J Zhao, J Pan, Y Xie, Y Cao, Y Wang, Z Wang, and 1 others. 2023. Advanced prompting as a catalyst: Empowering large language models in the management of gastrointestinal cancers. *The Innovation*, 521.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A The Use of Large Language Models

In this paper, we only used LLMs, including DeepSeek-R1 and ChatGPT, for polishing the writing. Specifically, we used LLMs to assist in refining the language, improving readability, and grammar checking. The authors take full responsibility for the content of the paper.

B Analysis and Proofs of Adaptive Difficulty-Control Mechanisms

In Section 3.1, we have formalized the difficulty control mechanisms as follows:

Let $[d_1, d_2, d_3]$ respectively denote the difficulty *Easy*, *Medium*, and *Hard*, n_{it}, n_{if} respectively represent the number of correct and incorrect answers for d_i-level questions, c_{it} be the score gain for a correct answer at d_i, and t_{ij} be the score threshold between d_i and d_j, we can get the average score avg_s and the difficulty of the next round question generation d_{next}:

$$avg_s = \frac{\sum_{i=1}^{3} c_{it} n_{it}}{\sum_{i=1}^{3}(n_{it} + n_{if})}$$

$$d_{next} = \begin{cases} d_1, & avg_s \leq t_{12} \\ d_2, & avg_s \leq t_{23} \\ d_3, & avg_s > t_{23} \end{cases}$$

where $[c_{1t}, c_{2t}, c_{3t}] = [1, 1.5, 2]$ and $[t_{12}, t_{23}] = [0.5, 1]$ in our experiments. In this section, we will illustrate the criteria and rules of the score gain c_{it} and the threshold t_{ij}, along with the analysis and proofs. For ease of representation, let $N = \sum_{i=1}^{3}(n_{it} + n_{if})$ in the subsequent analysis.

For the selection of score gains and thresholds, we followed simple cognitive principles to establish several rules that a reasonable difficulty control mechanism should satisfy:

R1. Balance: Correctly answering a more difficult question should gain a higher score. If the number of correctly answered *Easy* and *Hard* questions is equal, the level should be considered equivalent to the same total number of *Medium* questions. Additionally, the conventional range of the average score should correspond equally to the three difficulty levels in ascending order.

R2. Generalizability: The same formula, score gains, and thresholds should apply even to questions without difficulty labels, where difficulty is estimated solely based on the accuracy.

R3. Improvability: Correctly answering questions should increase the average score, and there should be a possibility of exceeding the threshold to advance to the next difficulty level.

R4. Stability: If the capability estimation remains at a certain difficulty for a long period, the likelihood of advancing to the next level should gradually decrease.

Following the above rules, we can analyze the mathematical conditions that c_{it} and t_{ij} must satisfy. From **R1**, we can get that $c_{1t} + c_{3t} = 2c_{2t} \wedge c_{1t} < c_{2t} < c_{3t}$. Since we have set the score for correctly answering a question without a difficulty level as c_{2t}, the range of avg_s can be determined as $[0\,,\, c_{2t}]$, representing the spectrum from answering all questions incorrectly to answering all correctly. Based on the condition about threshold in **R1**, we can get that $t_{12} = c_{2t}/3\ ,\ t_{23} = 2c_{2t}/3$ in the case that difficulty levels are absent. Due to **R2**, these threshold conditions can be generalized to the case where questions are provided with difficulty levels.

For **R3**, let avg_s denote the average score after answering N questions, and avg'_s denote the average score after answering $N+1$ questions. Then **R3** is equivalent to satisfying the following conditions:

$$\textbf{C1.}\quad avg'_s = \frac{avg_s N + c_{it}}{N+1} > avg_s, \quad \text{holds true always when } avg_s \leq t_{(i,i+1)} \tag{1}$$

$$\textbf{C2.}\quad \exists\ avg_{s0},\ N_0, \quad \text{s.t. } avg'_s = \frac{avg_{s0} N_0 + c_{it}}{N_0 + 1} > t_{(i,i+1)}$$

Simplifying Eq 1.C1, we find that $c_{it} > avg_s$ holds true always when $avg_s \leq t_{(i,i+1)}$. Therefore, we can conclude that $c_{it} > t_{(i,i+1)}$, meaning $c_{2t} > c_{1t} > t_{12}$ and $t_{23} < c_{2t} < c_{3t}$. Simplifying Eq 1.C2, we can obtain a relationship between avg_{s0} and N_0:

$$avg_{s0} > t_{(i,i+1)} - \frac{c_{it} - t_{(i,i+1)}}{N_0} \tag{2}$$

Combining the condition $c_{it} > t_{(i,i+1)}$ from Eq 1.C1, this relationship indicates that, under the premise of **R3**, advancing to the next difficulty level by answering questions correctly requires a higher avg_s as N increases. This description is essentially **R4**: the longer one remains at a specific difficulty level, the harder it becomes to advance to the next level. Therefore, **R3**⇒**R4** is true.

Finally, **R4** can be stated as follows: given the average score $a \leq t_{(i,i+1)}$ after N questions, the number n of consecutive questions that must be answered correctly to surpass the level threshold should increase with N. We can obtain the following inequality:

$$\begin{aligned} &\frac{aN + c_{it} n}{N+n} > t_{(i,i+1)} \\ &\Rightarrow (c_{it} - t_{(i,i+1)})n > (t_{i,i+1} - a)N \end{aligned} \tag{3}$$

If **R4** holds, which means n should increase as N increases, combined with the inherent condition $a \leq t_{(i,i+1)}$, it follows that $c_{it} > t_{(i,i+1)}$, from

which R3 can be deduced. Therefore, **R3**⇒**R4** and **R4**⇒**R3** hold simultaneously, meaning **R4** and **R3** are equivalent, demonstrating that we have now analyzed all the mathematical conditions that c_{it} and $t_{(i,i+1)}$ must satisfy:

$$\begin{aligned} &c_{1t} < c_{2t} < c_{3t} \\ &c_{1t} + c_{3t} = 2c_{2t} \\ &t_{12} = c_{2t}/3 \ , \ t_{23} = 2c_{2t}/3 \\ &c_{it} > t_{(i,i+1)} \\ &\qquad \Rightarrow c_{2t} > c_{1t} > t_{12} \ , \ t_{23} < c_{2t} < c_{3t} \end{aligned} \tag{4}$$

Therefore, we first set $c_{2t} = 1.5$, and so $t_{12} = 0.5$, $t_{23} = 1$. Actually, since it is not possible to move directly from *easy* to *hard* by correctly answering a single question, we can obtain the inequality chain $t_{12} < c_{1t} \leq t_{23} < c_{2t} < c_{3t}$. So c_{1t} is a constrained value determined by c_{2t}, and all other values can be derived from c_{2t}.

Furthermore, from Equation 2, by substituting $i = 1$, rearranging terms, and combining with the inequality chain, we obtain $t_{23} \geq c_{1t} > t_{12} + N_0(t_{12} - avg_{s0})$. Let N_c and N_f denote the numbers of correctly and incorrectly answered questions, respectively. Then $N_0 = N_c + N_f$ and $avg_{s0}N_0 = c_{1t}N_c$. By simplification, we obtain $c_{1t} > t_{12}(1 + N_f/(N_c + 1))$.

Therefore, c_{1t} should be determined based on the expected ratio of correct to incorrect answers. A lower c_{1t} requires a higher correct-to-incorrect ratio and more rounds of execution. Considering efficiency, JudgeAgent operates with fewer rounds, and thus a relatively higher value of c_{1t} will be selected. Thus, we set $c_{1t} = 1$, and so $c_{3t} = 2c_{2t} - c_{1t} = 2$.

C Construction of Context Graph

For the dataset selected during the Benchmark Grading stage, the process of constructing a context graph from its knowledge base (e.g., the set of all reference texts) can be divided into three components: text chunking, node construction, and node linking.

Text Chunking: To preserve the integrity of knowledge, the granularity of chunking is not refined to the sentence level. Instead, several sentences or an entire paragraph are grouped to form a single unit, treated as the minimal hierarchical "chunk". On this basis, chunks are further aggregated into higher-level "documents" according to whether they belong to the same article or serve as reference texts for the same question.

Node Construction: An LLM is employed to extract entities from each chunk, and the detailed prompt is provided in Appendix G. Each extracted entity serves as the subject for constructing a node. All chunks containing the same entity are assigned to the corresponding entity node. Formally, given an entity e, along with the chunks containing the entity $\{c_1, c_2, ...\}$ and the documents containing the entity $\{d_1, d_2, ...\}$, the node in the context graph can be defined as follows:

$$\begin{aligned} &N = (e \ , \mathcal{C} \ , \mathcal{D}) \ , \\ &\quad \text{where } \mathcal{C} = \{c_1, c_2 ...\} \ , \ \mathcal{D} = \{d_1, d_2, ...\} \end{aligned} \tag{5}$$

In practice, we store only the IDs of chunks and documents within each node to conserve space.

Node Linking: During construction, if two entities appear together in the same chunk or the same document-level texts, their corresponding nodes in the context graph are treated as neighbors. Formally, given two nodes $N_1 = (e_1 \ , \mathcal{C}_1 \ , \mathcal{D}_1)$ and $N_2 = (e_2 \ , \mathcal{C}_2 \ , \mathcal{D}_2)$, if $\exists \ c_{1i} \in \mathcal{C}_1, c_{2j} \in \mathcal{C}_2$, s.t. $c_{1i} \equiv c_{2j}$ or $\exists \ d_{1i} \in \mathcal{D}_1, d_{2j} \in \mathcal{D}_\in$, s.t. $d_{1i} \equiv d_{2j}$, then N_1 and N_2 will be treated as neighbors in the context graph.

D Statistics and Preprocessing of Datasets

D.1 Details of Datasets

The following benchmarks are used in our experiments, whose details are shown in Table 5.

MedQA (Jin et al., 2021) contains multiple-choice questions in the style of the Medical Licensing Examination. Questions in this dataset are collected from medical board exams in the US, Mainland China, and Taiwan, where human doctors are evaluated on their professional knowledge and ability to make clinical decisions. The background knowledge texts of MedQA are provided in the form of additional complete articles, and the questions only provide meta information.

MultiHop-RAG (Tang and Yang, 2024) consists of phrase Q&A queries, their ground truth answers, and the associated supported evidence constructed from news articles published between September and December 2023. The background knowledge texts of MultiHop-RAG are provided as supporting evidence along with the questions.

QuALITY (Pang et al., 2022) is a multiple-choice question dataset for long document comprehension, whose questions are written and validated by human contributors based on the long passages. The sources of QuALITY include: (1)

Datasets	Question Type	Categories	Splits	Used Splits	Split Size
MedQA	multiple-choice	medical clinical	train validation test	test	1273
MultiHopRAG	phrase QA	technology entertainment sports science business health	test	test	2556
QuALITY	multiple-choice	fiction stories magazine articles long articles	train validation test	validation	2086

Table 5: Details of datasets in our experiments. The language of all the datasets is English.

Project Gutenberg fiction stories, which are mostly science fiction; (2) Slate magazine articles from the Open American National Corpus; (3) other nonfiction articles taken from The Long+Shor, Freesouls, and the book Open Access. QuALITY is organized by articles, with each data item consisting of a long article and several related questions, and the article serves as the background knowledge for each question.

D.2 Preprocessing of Datasets

To verify JudgeAgent's ability to evaluate knowledge deficiencies in target models, we remove the background knowledge of the questions from MedQA and MultiHopRAG during evaluation. Otherwise, it would only test the target's reading comprehension ability rather than knowledge deficiencies. In contrast, we provided the background text when using QuALITY for evaluation, as the questions in QuALITY are highly dependent on the text content, and many of the texts are fictional narratives. Therefore, we use QuALITY to validate the JudgeAgent's effectiveness in guiding comprehension and reasoning rather than discovering knowledge deficiencies.

Additionally, we have reprocessed the difficulty levels of the questions from QuALITY. Based on the accuracy of human annotators in answering questions, QuALITY originally classified questions into two levels, *Easy* and *hard*, using a 50% accuracy threshold. To align with the difficulty levels defined by JudgeAgent's difficulty control module (*Easy*, *Medium*, and *Hard*), we re-labeled the questions using the same accuracy criteria. Questions of QuALITY with an accuracy below 1/3 are re-labeled as *Easy*, those with an accuracy between 1/3 and 2/3 as *Medium*, and the rest as *Hard*.

E Additional Experiment Analysis

E.1 Efficiency of JudgeAgent

Target	MedQA		MultiHopRAG	
	avg.token	avg.time	avg.token	avg.time
GLM4-Flash	1558	7.24	2424	23.38
Qwen3	1909	9.95	3453	30.96

Table 6: Efficiency of JudgeAgent. *avg.tokens* and *avg.time* respectively denote the average tokens and time (seconds) cost of JudgeAgent per seed question.

We calculated the time and token costs incurred by JudgeAgent during multi-turn interactions with GLM4-Flash and Qwen3 as target models, calculated as averages based on the number of questions in the benchmark selected during the Benchmark Grading stage. The results are shown in Table 6.

According to Algorithm 1 and 3, the graph construction and evaluation processes of JudgeAgent mainly involve unidirectional traversal, with complexity close to linear time. The cost of introducing new knowledge and expanding the benchmark is not higher than manual expansion. Therefore, JudgeAgent's efficiency and scalability are well supported by both theory and experimental results.

E.2 Stability of JudgeAgent

To investigate the stability of JudgeAgent's results, this section presents additional experiments conducted under 5 different random seeds, with the

Target	MedQA			
	ACC1	ACC2	CR↑	CtE↓
Qwen3	91.76±0.09	96.18±0.17	4.83±0.17	0.42±0.08
GLM4-Flash	80.36±0.34	92.65±0.29	13.00±0.38	0.72±0.18

Target	MultiHopRAG			
	ACC1	ACC2	CR↑	CtE↓
Qwen3	63.97±0.53	70.41±0.59	15.78±1.29	9.34±1.35
GLM4-Flash	51.54±0.38	66.56±0.51	23.44±0.88	8.22±1.03

Table 7: The experiment results with standard deviation calculated based on 5 different random seeds.

standard deviation calculated. Experiments are conducted using Qwen3 and GLM4-Flash as target models on the MedQA and MultiHop-RAG benchmarks. The results are shown in Table 7.

The experiment results demonstrate that the results across multiple random seeds remain stable and are consistent with the ranking of results presented in the main results in Table 1. This stability benefits from JudgeAgent's context-graph control and knowledge-driven synthesis pathway for evaluation questions. These mechanisms endow the generated questions and the evaluation feedback with controllability and stability.

E.3 The Resilience Against Data Contamination

Can JudgeAgent mitigate the challenges of data contamination in static benchmarking paradigms? In this section, we simulate a scenario where the static benchmarking evaluation paradigm suffers from data contamination by deliberately exposing the evaluation questions to LLMs during their training process. We selected Llama3-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen2.5-7B-Instruct as base models, and constructed supervised fine-tuning (SFT) data from the MedQA and MultiHop-RAG benchmarks, which are intended for evaluation in the main experiments. These training data were used to fine-tune the selected base models. By comparing the performance differences between the original base models and the fine-tuned models on both the static benchmark questions and the extended questions generated by JudgeAgent, we analyze and verify the resilience of JudgeAgent and its derivative JudgeAgent against data contamination.

Specifically, the procedure can be summarized as follows: for a base LLM $\mathcal{M}$ and a static evaluation benchmark $\mathcal{D}$, a fine-tuned LLM $\mathcal{M}$-sft is obtained by supervised fine-tuning with the training data constructed from $\mathcal{D}$. The performance of $\mathcal{M}$ and $\mathcal{M}$-sft, which is measured by the accuracy (ACC) in answering the questions, is then compared on both $\mathcal{D}$ and the extended questions $\mathcal{D}$@K generated at the K-th iteration. The severity of data contamination (Δ) is measured by the improvement in performance from the fine-tuned model $\mathcal{M}$-sft to the base model $\mathcal{M}$, which is formalized as $\Delta = ACC_{\mathcal{M}\text{-sft}} - ACC_{\mathcal{M}}$. To mitigate the effects of the LLM's randomness, each question was answered by the LLM 5 times. If the LLM produced a correct answer in three or more of these trials, it was considered to have answered the question correctly. We conduct our experiments on an Ubuntu machine with one 40GB NVIDIA A100 GPU. The results are shown in Table 8.

The experiment results indicate that, across various LLMs and benchmarks, fine-tuning with evaluation data leads to a notable improvement in model performance on base questions (Δ-base), particularly evident on the MultiHop-RAG benchmark. These findings underscore the risks of data contamination: even when the original model exhibits limited performance on benchmarks, exposure to the benchmark evaluation data can artificially inflate its performance. Consequently, the model's genuine capabilities may be obscured by overestimated benchmark performance, leading to misapplication in scenarios beyond the actual capabilities.

In contrast, when evaluated on extended questions generated by JudgeAgent, the fine-tuned models and base models show little difference in performance. Additionally, fine-tuning even resulted in a decline on the MedQA benchmark. These results suggest that under the JudgeAgent dynamic evaluation paradigm, the generated questions main-

Models	MedQA				MultiHop-RAG			
	base	@1	@2	@3	base	@1	@2	@3
Llama3-8B-Instruct	62.17	69.94	67.72	69.77	48.77	28.30	28.38	29.15
Llama3-sft	82.53	65.59	65.80	65.42	88.45	29.59	29.19	29.79
Δ	**20.34**	-4.35	-1.92	-4.35	**39.69**	1.29	0.81	0.65
Mistral-7B-Instruct-v0.3	57.91	59.40	58.76	56.97	59.51	37.75	38.60	37.22
Mistral-sft	62.73	53.18	51.98	51.98	92.17	32.74	32.58	32.62
Δ	**4.82**	-6.23	-6.78	-4.99	**32.66**	-5.01	-6.02	-4.60
Qwen2.5-7B-Instruct	85.93	78.04	75.44	77.31	46.31	20.59	21.64	21.48
Qwen2.5-sft	94.20	78.00	75.78	77.27	88.90	26.60	25.56	24.75
Δ	**8.27**	-0.04	0.34	-0.04	**42.59**	6.02	3.92	3.27

Table 8: The results for validating the resilience against data contamination. *base* means the performance on base static benchmark questions, and @K means extended questions at the K-th iteration.

tain the validity of the evaluation, even when the original questions have been exposed to the target model. Furthermore, under the same setting of LLM and benchmark, there is a marked gap between the performance gain on base questions and extended questions after fine-tuning, demonstrating that JudgeAgent exhibits considerable resilience to data contamination.

Target	**ACC1**	**ACC2**$_d$	**ACC2**$_f$
Qwen3	91.71	99.18	96.38
GLM4-Flash	80.09	98.37	92.82
GPT-4.1	84.97	98.28	92.44

Table 9: The comparative results on MedQA of directly providing correct answers (ACC2$_d$) and providing feedback from JudgeAgent (ACC2$_f$). ACC1 represents the accuracy of target models before evaluation. All values are percentages.

Additionally, we conducted a comparative experiment providing the target models with both standard evaluation feedback from JudgeAgent and evaluation feedback containing the correct answers as cheating information, investigating the risk of data contamination during the evaluation. The results are presented in Table 9. As can be observed, when the correct answers are directly provided, nearly all LLMs achieve almost perfect accuracy. This further illustrates the risk of data contamination in existing static benchmarks. It also demonstrates that the evaluation feedback from JudgeAgent does not contain cheating information tailored specifically to the seed questions

E.4 Quality of Evaluation Questions

We use Dingo[3] to check the factuality of evaluation questions generated by JudgeAgent in multi-turn interactions. The results are shown in Table 10, N_{eq} represents the number of generated questions in Stage Interactive Extension.

Datasets	N_{eq}	avg. fact ratio	fact check passed N_{passed}	passed ratio
MedQA	3819	97.93%	3680	96.36%
MultiHop-RAG	7668	94.38%	7185	93.70%

Table 10: The fact check results by Dingo.

As shown in Table 10, in Dingo's fact-checking task, the extended questions from JudgeAgent maintain a high fact ratio and detection pass ratio (using Dingo's default fact ratio threshold of 80%). The average fact ratio is above 94%, while the pass ratio exceeds 93% for both MedQA and MultiHop-RAG. The results demonstrate that the most generated questions exhibit strong factuality, ensuring that the dynamic evaluation results are based on high-quality questions, and the probability of an evaluated LLM being incorrectly judged as wrong when its answer is correct remains low.

As for the diversity of evaluation questions, the experiment results of data contamination in Table 8 in Appendix E.3 can indirectly demonstrate that contaminated models perform poorly on synthesized questions, which indirectly confirms that synthesized questions differ from seed questions and that their diversity is assured.

In addition, we verify whether the difficulty of

[3] https://github.com/MigoXLab/dingo

Base Benchmarks	JudgeAgent's Label		
	easy	*medium*	*hard*
MultiHopRAG	0.846	0.674	0.524

Table 11: Average difficulty scores of questions under different JudgeAgent's difficulty labels on different base benchmarks.

the questions generated by the JudgeAgent accurately follows the algorithm's control. We quantify the difficulty of the questions by examining the performance of multiple models of different scales, which roughly reflect different capability levels. Specifically, we evaluate five LLMs, including Qwen2.5-1.5B/3B/7B-Instruct and Llama-3/3.1-8B-Instruct. We use the proportion of models that answer correctly for each question as a metric for the question's difficulty. We then compute the average difficulty of questions under the *easy*, *medium*, and *hard* difficulty labels, as shown in Table 11. It can be observed that the average difficulty score is positively correlated with the difficulty labels assigned by JudgeAgent, and there is clear separation between different difficulty levels. This indicates that the evaluation questions generated by JudgeAgent follow the intended difficulty control effectively, and the question quality is well ensured.

E.5 Cross Validation of the Evaluation Suggestions

Do the suggestions provided by JudgeAgent only take effect for the seed questions? To address this question, we designed cross-validation experiments from two perspectives.

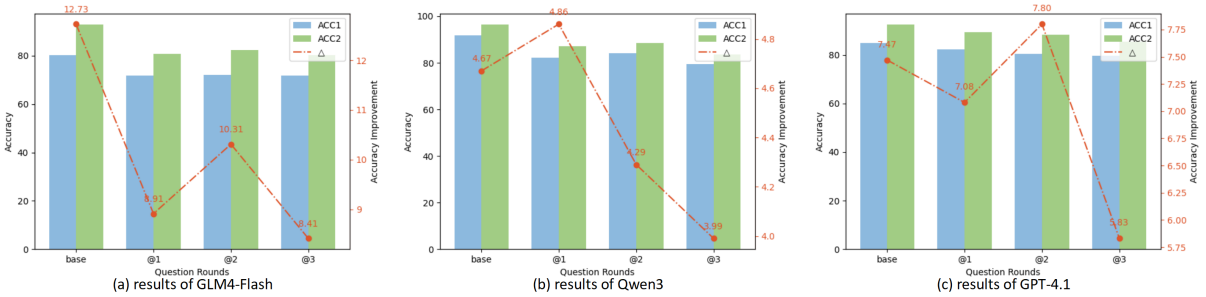
First, considering that the evaluation suggestions are derived from a comprehensive evaluation of the responses to both the seed questions and their extended questions, we evaluated and compared the accuracy improvement of the suggestions on the extended questions versus the seed questions, aiming at verifying that the suggestions do not contain "cheating information" specific to seed questions in the scenario after evaluation. The results are shown in Figure 6.

Secondly, we transferred the evaluation suggestions derived from seed questions to other questions with related knowledge concepts in the benchmark, to verify the effectiveness of the suggestions within the same knowledge domain. Specifically, we categorized questions based on the knowledge entities they contain, formalized as follows: given the context graph $\mathcal{G}$, for two questions q_1 and q_2 with knowledge entities $E_1 = \{e|e \in q_1 \wedge e \in \mathcal{G}\}$ and $E_1 = \{e|e \in q_2 \wedge e \in \mathcal{G}\}$, if E_1 and E_2 have a non-empty intersection, then q_1 and q_2 are considered related questions. In this experiment, the suggestions for a question were constructed from the suggestions of its related questions, excluding suggestions from the question itself, to assess and validate the transferability of the suggestions provided by the JudgeAgent. We screened out questions without relevant questions and those with only relevant questions based on non-knowledge entities, such as male, female, 2 years, etc. The results are shown in Table 12.

First, we analyze the difference in the effectiveness of evaluation suggestions on seed questions versus extended questions. As shown in Figure 6, suggestions effectively improve performance for both seed questions and extended questions, with a relatively small gap in the degree of improvement. Notably, when Qwen3 and GPT-4.1 are used as target models, the first round of Qwen3 and the second round of GPT-4.1 exhibit even greater improvement than seed questions. These results indicate that although the suggestions only supplement knowledge for several key concepts, such as the case in Figure 8, such concise suggestions can still benefit both seed and extended questions, demonstrating that JudgeAgent is capable of identifying and addressing knowledge gaps in the target model, rather than simply providing "cheating information" specific to seed questions.

Furthermore, it is observed that the effectiveness of suggestions for third-round questions is consistently low in the experiments. This may be because, by the third round, knowledge path sampling has expanded beyond the scope of knowledge related to seed questions to a broader range. As a result, the generated questions diverge more significantly in core knowledge from earlier questions, thereby reducing the effectiveness of the knowledge guidance provided in the suggestions.

Next, we analyze the difference in the effectiveness of suggestions on questions that share the same knowledge concepts. As shown in Table 12, compared to their effectiveness on seed questions, the suggestions exhibit a slight decrease when applied to related questions, as indicated by a decline in CR and Δ, and an increase in CtE. But the difference is minor, and the improvement remains notable, suggesting that the suggestions can be effectively transferred to other questions involving the same knowledge concepts. This further demon-



(a) results of GLM4-Flash (b) results of Qwen3 (c) results of GPT-4.1

Figure 6: The cross-validation results of extended questions versus seed questions on MedQA. ACC1 and ACC2 indicate the accuracy before and after evaluation. $\Delta = \text{ACC2} - \text{ACC1}$ refers to the overall accuracy improvement. All the values are percentages. @K represents the results of the questions expanded at the K-th round.

Target	ACC1	Non-transfer				Transfer			
		ACC2	CR↑	CtE↓	Δ ↑	ACC2	CR↑	CtE↓	Δ ↑
GLM4-Flash	78.42	87.49	18.30	9.23	9.07	85.56	16.98	9.84	7.13
GPT-4.1	84.21	91.09	10.23	3.35	6.88	89.78	9.50	3.93	5.57
Qwen3	92.44	96.06	5.96	2.34	3.62	95.17	5.17	2.44	2.73

Table 12: The cross-validation results of transferring suggestions to related questions. All the values are percentages. *Non-transfer* refers to suggestions being applied to the seed questions, whereas *Transfer* refers to them being applied to related questions.

strates that JudgeAgent can provide suggestions that do not simply serve as "cheating information" specific to seed questions.

However, the slight decline in evaluation effectiveness also indicates that the knowledge guidance provided in the suggestions is not fully aligned with the related questions. This may be because the overlapping entities between these related questions and seed questions do not correspond to core knowledge concepts. For example, suggestions centered on "blood type" may be transferred to a question where "serum" is the core knowledge concept, resulting in a partial mismatch.

The above experiment results demonstrate that the evaluation suggestions provided by JudgeAgent are not only applicable to seed questions but can also be transferred to other questions that share relevant core knowledge concepts.

E.6 Comparative Experiments between JudgeAgent and Self-Correction

Does JudgeAgent genuinely identify model deficiencies, or does it merely activate the model's self-correction ? To address this problem, we conduct experiments where the evaluated model performs self-correction. We include the model's initial answer in the prompt and instruct it with: "Please try to rethink your answer and conduct self-correction, then answer the question again." The evaluation results of GLM4-Flash and Qwen3 on MultiHopRAG are shown in Table 13.

Target	MultiHopRAG			
	ACC1	ACC2	CR↑	CtE↓
GLM4-Flash				
w/ self-correction	51.25	57.37	22.74	16.62
w/ JudgeAgent		65.92	24.26	9.59
Qwen3				
w/ self-correction	63.65	67.80	19.06	14.91
w/ JudgeAgent		70.07	17.49	11.07

Table 13: Comparison of the model's performance after self-correction and JudgeAgent's guidance.

The results show that prompting the evaluated model for self-correction does provide some improvement, but the gains are clearly smaller than those achieved by JudgeAgent. Moreover, self-correction leads to a higher correction-to-error rate (CtE) and exhibits less stable optimization performance. This phenomenon may arise because, under self-correction prompting, the model is more likely to negate its previous answers. In contrast, JudgeAgent benefits from graph-based guidance, making its feedback more targeted. Combined with the experimental results in Table 1, this indicates that JudgeAgent can indeed identify flaws in the evaluated model, rather than merely activating its self-correction ability.

Furthermore, according to the ablation study

results in Figure 3, this ability to identify flaws largely stems from the introduction of dynamic interaction and the context graph. Dynamic interaction allows JudgeAgent to probe the model's flaws based on relevant knowledge and estimated model capability, while the context graph provides tightly connected knowledge and text segments.

E.7 Supplementary Parameter Analysis

What is the impact of batch size in the *Benchmark Grading* stage on the evaluations of the JudgeAgent? In *Benchmark Grading* stage, questions are divided into batches to comprehensively assess the target's capabilities at a base level. These batches are also the basic units for question extension and evaluation feedback. Given a fixed number of rounds, the batch size is inversely proportional to the number of batches, extended questions, and evaluation suggestions, thereby influencing the time and resource consumption of the entire evaluation process. **Can the batch size be maximized to reduce resource consumption while maintaining the effectiveness of evaluations?** To address this question, we conducted a parameter analysis experiment, examining the evaluation effectiveness and time consumption under different batch sizes. The results are shown in Figure 7.

As observed from the trend of the curves in Figure 7, both the target's accuracy after receiving evaluation suggestions (ACC2) and the average time consumption per question for evaluation (Time Cost) decrease as the batch size increases. Among them, the decline rate in ACC2 gradually accelerates with larger batch sizes, while the decline rate in Time Cost gradually slows, exhibiting a marginal effect. The reason for the marginal effect in Time Cost lies in the fact that the time required for the target model to answer the seed questions, which is a component of the overall evaluation process, varies little with changes in batch size. As a result, there is a threshold beyond which further reductions in time cost have diminishing returns.

The decline in ACC2 with increasing batch size can be attributed to the expansion of the question's knowledge domain. As the batch size grows, the generated questions during evaluation become more heterogeneous and less coherent with the knowledge relevant to seed questions, making it difficult for JudgeAgent to identify appropriate knowledge guidance from the dispersed question-answer pairs. Moreover, when the batch size exceeds the number of extension rounds, the disorder of knowledge scopes intensifies more rapidly, ultimately leading to a sharp drop in accuracy. The results in Figure 7 show that when the batch size reaches 9, the evaluation suggestions even become counterproductive (ACC2 $<$ ACC1), interfering with the normal reasoning of the target model.

Therefore, considering both the evaluation time cost and effectiveness, we selected a batch size of 3 in our experiments as a balanced choice.

F Algorithm

To clarify the entire workflow of JudgeAgent, we use pseudocode to show the dynamic evaluation process in Algorithm 1, the validation process of evaluation in Algorithm 2, the construction of context graph in Algorithm 3, and the generation of extended questions in Algorithm 4.

G Prompts

We show the detailed prompt for Entity Extraction in Prompt 14, the prompt for generating extended questions in Prompt 15, the prompt for evaluating the performance of the target LLM in Prompt 16, and the prompt for querying the target LLM with suggestions in Prompt 17.

H Detailed Content of Case Study

The detailed content of the comparative case in section 4.6 is shown in Figure 8.

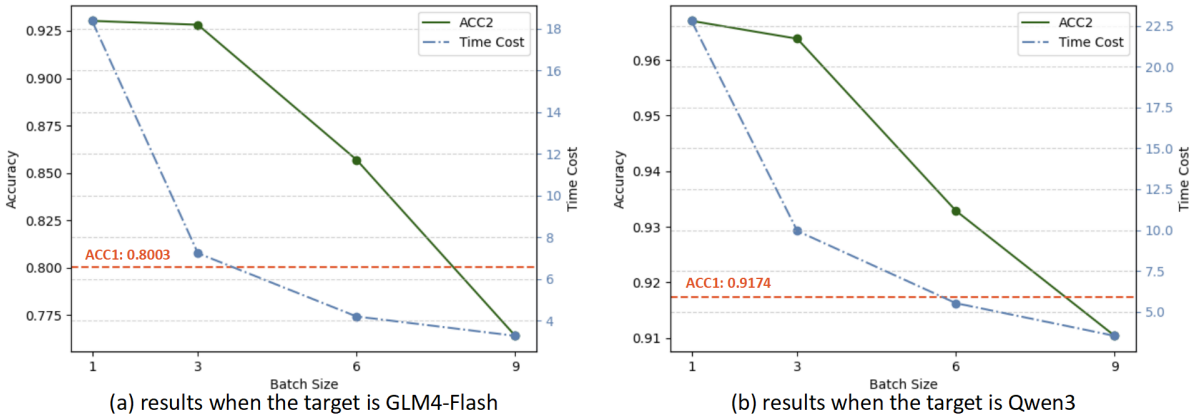


Figure 7: The results of different batch sizes on MedQA. ACC1 and ACC2 indicate the accuracy before and after receiving evaluations. Time Cost is the average time consumption for each question.

Prompt for Entity Extraction

Please identify and label the entities in the following multiple sentences, and return the entity labeling results for each sentence.
The results for each sentence should be independent, in JSON format, containing the sentence number, sentence text, and the list of recognized entities (including entity text, type, and position).
Return format is a dictionary, with only one key 'labeled_data', and the value is a list, each element is a dictionary containing the sentence text and the entity list.

```
{{
    "labeled_data":
    [
        {{"text":"Sentence 1", "entity_list": [{{"entity_text": "", "entity_type": ""}}]}},
        {{"text":"Sentence 2", "entity_list": [{{"entity_text": "", "entity_type": ""}}]}},
        ...
    ]
}}
```

Notice that "text" should be only the sentence, not the whole article. Sentence list:
{ *Sentences* }

Table 14: Prompt for entity extraction.

Prompt for Generating Extended Questions

As an interviewer, you are tasked with designing questions based on the provided texts. Your role involves crafting questions and correct answers that fulfill the following criteria:

1. **Focus on the Entity**: Ensure all questions consistently center around the specified entity from the article.
2. **Ensure Accuracy and Conciseness of Answers**: Verify that the provided answer is both correct for your designed question within the context and logic of the given text fragments, and ensure the answer is sufficiently concise—presented as a word or phrase, avoiding redundancy.
3. **Conform to difficulty requirements**: You need to design questions for the required difficulty levels, with specific requirements as follows:

(1). [easy]: **Encourage Knowledge Memorization**, design questions that assess whether respondents have memorized relevant knowledge. Create questions by directly extracting and blanking out content from the given passage.

(2). [medium]: **Encourage Knowledge Comprehension**: Design questions that prompt respondents to dissect and comprehend concepts involved in the topic. Avoid assessing only superficial knowledge retention.

(3). [hard]: **Encourage Knowledge Deep Analysis**: Design questions that prompt respondents to engage in deep thinking and analysis. Avoid merely testing knowledge recall or conceptual comprehension; do not simply extract fragments from the given passage to create fill-in-the-blank items. Encourage respondents to focus on entities within the question and employ logical skills for complex reasoning.

Here are examples:
{ *Examples* }

Now, given the following text fragments:
{ *context* }

Based on the provided texts, please generate questions by following the requirements above and referencing the examples.
Output in the specified JSON format below: {{

```
"generated_question":
[
    {{
        "question": "Generated Question",
        "answer": "Correct Answer of Generated Question"
    }},
...
]
```

}}

Table 15: Prompt for generating extended questions based on different difficulties.

Prompt for Evaluation

The following is the performance of an LLM in answering a series of questions:

{ *list of questions, correct answers, and LLM's answers* }

Please evaluate and analyze the interviewee's performance based on the above performance using concise language from the following perspectives, and provide suggestions that help the LLM answer the same questions better. Suggestions should provide specific and detailed guidance on logical thinking steps, required knowledge, and abilities, ensuring the LLM can answer correctly for the same questions.
Output in the following JSON format:

```
{{
    "flaws_knowledge": "The lack of background knowledge.",
    "flaws_capability": "The flaws in logic and capability.",
    "comprehensive_performance": "The Comprehensive performance of all questions.",
    "suggestions": "Suggestions that help the LLM answer questions bette"
}}
```

Table 16: Prompt for evaluating the performance of the target LLM.

Prompt for Querying LLM with suggestions

Please complete the following question:
[question]: { *question* }

In your previous responses to these questions, the interviewer has provided the following suggestions for you to help you answer better:
[suggestions]: { *suggestions* }

Please consider the above [suggestions], and answer the above [question], in the following JSON format:
{{"answer": "Your answer"}}

Table 17: Prompt for querying the target LLM with suggestions.

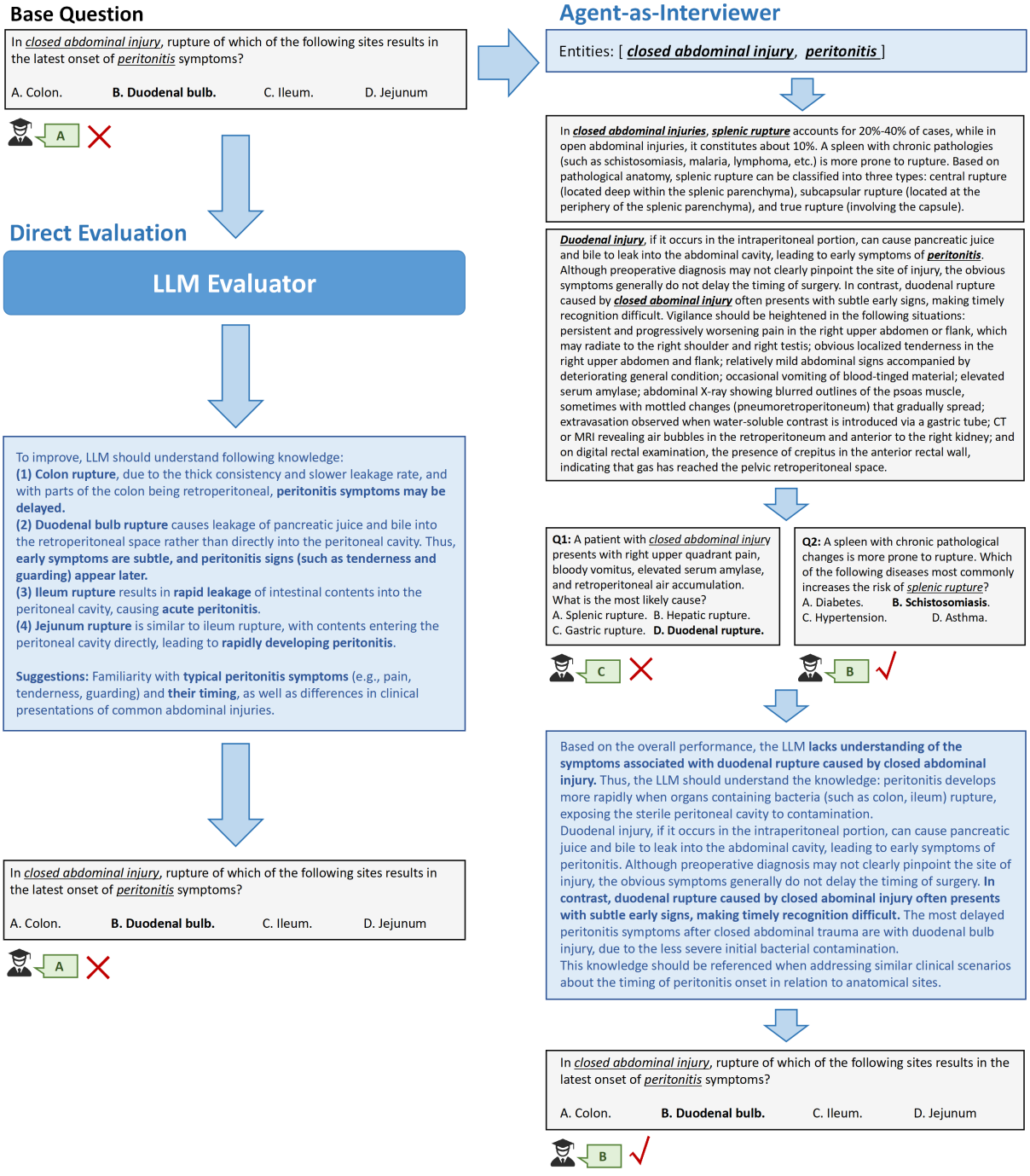


Figure 8: The detailed content of the case.

Algorithm 1 Dynamic evaluation process of JudgeAgent

Input: Target LLM $\mathcal{M}_t$, Base dataset $\mathcal{D} = \{(q_i, a_i, d_i)\}_{i=1}^{N}$ (each item include question q, answer a, and difficulty d), Knowledge bases $\mathcal{K} = \{k_1, k_2, ...\}$, Core LLM of JudgeAgent $\mathcal{M}_c$, Predefined batch capacity N_B, Max extension round RND_e, Max hop of sampling H
Output: Evaluation Score sc, Batched dataset with suggestions $\mathcal{D}_S = \{(\mathcal{B}_1\,,\,s_1)\,,\,\dots\}$
// Construct context graph
$\mathcal{G} \leftarrow$CONSTRUCT_CONTEXT_GRAPH($\mathcal{K}\,,\,\mathcal{M}_c$)
// Split base dataset into batches
$\{\mathcal{B}_1 = \{(q_{1i}, a_{1i}, d_{1i})\}_{i=1}^{N_B}\,,\dots\} \leftarrow$SPLIT_BATCHES($\mathcal{D}\,,\,N_B$)
// Begin Evaluation
$sc \leftarrow 0\,,\; N_{total} \leftarrow 0$
$\mathcal{D}_S \leftarrow \{\}$
for $\mathcal{B} \leftarrow \{\mathcal{B}_1\,,\,\mathcal{B}_2\,,\,\dots\}$ **do**
    $sc_{\mathcal{B}} \leftarrow 0$
    $RND_{total} \leftarrow 0$
    $\mathcal{Q}_{tested} \leftarrow \{\}$
    // Stage1: Benchmark Grading
    for $(q, a, d) \leftarrow \mathcal{B}$ **do**
        Get answer $a_{\mathcal{M}} \leftarrow$QUERY_LLM($q\,,\,\mathcal{M}_t$)
        if $a_{\mathcal{M}}$ is correct based on q and a **then**
            $sc_{\mathcal{B}} \leftarrow sc_{\mathcal{B}} +$DIFFICULTY_SCORE($d$)
        end if
        $RND_{total} \leftarrow RND_{total} + 1$
        Add $(q, a, d, a_{\mathcal{M}})$ to $\mathcal{Q}_{tested}$
    end for
    Decide difficulty $d_e \leftarrow$DECIDE_DIFFICULTY($sc_{\mathcal{B}}\,,\,RND_{total}$)
    // Stage2: Interactive Extension
    for $i \leftarrow \{1, 2, \dots, RND_e\}$ **do**
        $(q_e, a_e, t) \leftarrow$GENERATE_EXTENDED_QUESTIONS($\mathcal{B}\,,\,\mathcal{G}\,,\,\mathcal{M}_c\,,\,H\,,\,d_e$)
        Get answer $a_{\mathcal{M}} \leftarrow$QUERY_LLM($q_e\,,\,\mathcal{M}_t$)
        if $a_{\mathcal{M}}$ is correct based on q_e and a_e **then**
            $sc_{\mathcal{B}} \leftarrow sc_{\mathcal{B}} +$DIFFICULTY_SCORE($d_e$)
        end if
        $RND_{total} \leftarrow RND_{total} + 1$
        Add $(q_e, a_e, d_e, a_{\mathcal{M}})$ to $\mathcal{Q}_{tested}$
        Decide difficulty $d_e \leftarrow$DECIDE_DIFFICULTY($sc_{\mathcal{B}}\,,\,RND_{total}$)
    end for
    // Stage3: Evaluation Feedback
    Get evaluation suggestions $s \leftarrow$EVALUATE($\mathcal{Q}_{tested}\,,\,\mathcal{M}_c$)
    Add $(\mathcal{B}, s)$ to $\mathcal{D}_{\mathcal{S}}$
    $sc \leftarrow sc + sc_{\mathcal{B}}$
    $N_{total} \leftarrow N_{total} + RND_{total}$
end for
$sc \leftarrow sc\,/\,N_{total}$
return $sc\,,\;\mathcal{D}_S$

Algorithm 2 Validation process of evaluation results from JudgeAgent

Input: Target LLM $\mathcal{M}_t$, Batched dataset with suggestions $\mathcal{D}_S = \{(\mathcal{B}_1\,,\,s_1)\,,\,\dots\}$, in which $\mathcal{B}_i = \{(q_{i1}, a_{i1}, d_{i1})\,,\,\dots\}$ is a batch of base dataset $\mathcal{D}$, and $\mathcal{S}_i = \{s_{i1}\,,\,\dots\}$ is relevant suggestions from JudgeAgent.
Output: Accuracy of target LLM before evaluation acc_1, Accuracy after evaluations acc_2, Correction Rate cr, Correct-to-Error Rate ce
// Initialize counter of questions
$N_{acc1} \leftarrow 0\,,\ N_{acc2} \leftarrow 0\,,\ N_{cr} \leftarrow 0\,,\ N_{ce} \leftarrow 0\,,\ N_{total} \leftarrow 0$
for $(\mathcal{B}, s) \leftarrow \mathcal{D}_S$ **do**
    $N_{\mathcal{B}} \leftarrow$LEN($\mathcal{B}$)
    $N_{total} \leftarrow N_{total} + N_{\mathcal{B}}$
    for $i \leftarrow \{1, 2, \dots, N_{\mathcal{B}}\}$ **do**
        $(q, a, d) \leftarrow \mathcal{B}[i]$
        Get answer1 $a_1 \leftarrow$QUERY_LLM($q\,,\,\mathcal{M}_t$)
        Get answer2 $a_2 \leftarrow$QUERY_LLM_WITH_SUGGESTIONS($q\,,\,s\,,\,\mathcal{M}_t$)
        correct1$\leftarrow$whether a_1 is correct based on q and a
        correct2$\leftarrow$whether a_2 is correct based on q and a
        if correct1 **then**
            $N_{acc1} \leftarrow N_{acc1} + 1$
            if not correct2 **then**
                $N_{ce} \leftarrow N_{ce} + 1$
            end if
        end if
        if correct2 **then** State $N_{acc2} \leftarrow N_{acc2} + 1$
            if not correct1 **then**
                $N_{cr} \leftarrow N_{cr} + 1$
            end if
        end if
    end for
end for
$acc_1 \leftarrow N_{acc1}/N_{total}\,,\ acc_2 \leftarrow N_{acc2}/N_{total}\,,\ cr \leftarrow N_{cr}/N_{total}\,,\ ce \leftarrow N_{ce}/N_{total}$
return $acc_1\,,\ acc_2\,,\ cr\,,\ ce$

Algorithm 3 Construction process of context graph

Input: Knowledge Base of Dataset $\mathcal{K} = \{k_1\ ,\ k_2\ ,\ \dots\}$, an LLM $\mathcal{M}$
Output: Context Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, in which $\mathcal{N}$ is the node set, and $\mathcal{E}$ is the edge set.
$\mathcal{N} \leftarrow \{\}\ ,\ \mathcal{E} \leftarrow \{\}$ *// Initialize node set and edge set as empty dictionary*
for $k \leftarrow \mathcal{K}$ **do**
// Spliting text into chunks
    $C = \{c_1, c_2, \dots\} \leftarrow$SPLIT_TO_CHUNKS($k$)
// Prompting LLM to label entities
    $P_e \leftarrow \emptyset$
    for $c \leftarrow \{c_1, c_2, \dots\}$ **do**
        $S_e = \{e_1, e_2, \dots\} \leftarrow$LABEL_ENTITY($c\ ,\ \mathcal{M}$)
        for $e \leftarrow S_e$ **do**
            if e not in $\mathcal{N}$ **then**
                $\mathcal{N}[e] \leftarrow (e\ ,\ \mathcal{C} = \emptyset\ ,\ \mathcal{D} = \emptyset)$
                $\mathcal{E}[e] \leftarrow \emptyset$
            end if
            Add c to set $\mathcal{N}[e].\mathcal{C}$
            Add k to set $\mathcal{N}[e].\mathcal{D}$
            Expand set $\mathcal{E}[e]$ with S_e
        end for
        Expand set P_e with S_e
    end for
    for $e \leftarrow P_e$ **do**
        Expand set $\mathcal{E}[e]$ with P_e
    end for
end for
$\mathcal{G} \leftarrow (\mathcal{N}\ ,\ \mathcal{E})$
return $\mathcal{G}$

Algorithm 4 Generation process of extended questions

Input: Base Question $\mathcal{Q} = \{q_1, q_2, \dots\}$, Context Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, LLM $\mathcal{M}$, Max hop of path H, Difficulty d
Output: Extended question with its answer and background text $(q_e\ ,\ a_e\ ,\ t)$
// Prompting LLM to label entities from $\mathcal{Q}$
$S_e \leftarrow \{\}$
for $q \leftarrow \mathcal{Q}$ **do**
    $set_e = \{e_1, e_2, \dots\} \leftarrow$LABEL_ENTITY($q\ ,\ \mathcal{M}$)
    Add set_e to S_e
end for
$e \leftarrow$RANDOM_SAMPLE($\{e_1, e_2, \dots\}\ ,\ 1$)
// Sample knowledge paths
$e' \leftarrow$the most similar entity in $\mathcal{N}$ of $\mathcal{G}$
$E_{visited} \leftarrow \{e'\}$
$t \leftarrow$RANDOM_SAMPLE($\mathcal{N}[e'].C\ ,\ 1$)
for $i \leftarrow \{1, 2, \dots, H-1\}$ **do**
    $E_{candidate} \leftarrow$the most similar 5 entities to e' in $\mathcal{E}[e']$ that not in $E_{visited}$
    $e' \leftarrow$RANDOM_SAMPLE($E_{candidate}\ ,\ 1$)
    $C_{candidate} \leftarrow$the most similar 5 chunks to q in $\mathcal{N}[e'].C$
    $c \leftarrow$RANDOM_SAMPLE($C_{candidate}\ ,\ 1$)
    Concatenate c to new line of t
    Add e' to $E_{visited}$
end for
$(q_e, a_e) \leftarrow$GENERATE_QUESTION($t\ ,\ \mathcal{M}$)
return $(q_e\ ,\ a_e\ ,\ t)$