GraphGen: Enhancing Supervised Fine-Tuning for LLMs with Knowledge-Driven Synthetic Data Generation

Anonymous ACL submission

Abstract

Fine-tuning for large language models (LLMs) typically requires substantial amounts of high-quality supervised data, which is both costly and labor-intensive to acquire. While synthetic data generation has emerged as a promising solution, existing approaches frequently suffer from factual inaccuracies, insufficient long-tail coverage, simplistic knowledge structures, and homogenized outputs. To address these challenges, we introduce GraphGen, a knowledge graph-guided framework designed for three key question-answering (QA) scenarios: atomic QA, aggregated QA, and multi-hop QA. It begins by constructing a fine-grained knowledge graph from the source text. It then identifies knowledge gaps in LLMs using the expected calibration error metric, prioritizing the generation of QA pairs that target high-value, long-tail knowledge. Furthermore, GraphGen incorporates multi-hop neighborhood sampling to capture complex relational information and employs style-controlled generation to diversify the resulting QA data. Experimental results on knowledge-intensive tasks under closed-book settings demonstrate that GraphGen outperforms conventional synthetic data methods, offering a more reliable and comprehensive solution to the data scarcity challenge in supervised fine-tuning. The code and data are publicly available at https://anonymous.4open.science/r/GraphGen.

1 Introduction

The rapid advancement of large language models (LLMs) has created a growing need for fine-tuning general-purpose models to incorporate new knowledge efficiently. One widely adopted approach is supervised fine-tuning (SFT), which enables LLMs to learn domain-specific information from labeled training data (Parthasarathy et al., 2024; Lu et al., 2024). While SFT has proven effective in enhancing model knowledge (Mecklenburg et al., 2024), its success heavily depends on access to large-scale, high-quality training datasets, which are expensive to curate and require substantial domain expertise.

To mitigate this data bottleneck, researchers have explored LLM-based synthetic data generation (Liu et al., 2024), leveraging LLMs to autonomously generate training samples, such as question-answer (QA) pairs or textual knowledge snippets. Several existing methods (Zhang and Yang, 2023; Maini et al., 2024) attempt to enhance domain adaptation by expanding training resources. However, when applied to knowledge-intensive tasks in closed-book settings, these synthetic data generation pipelines exhibit critical limitations:

1. **Factual Inaccuracy:** LLMs often introduce factual errors due to their tendency to hallucinate incorrect or non-factual knowledge (Long et al., 2024), leading to unreliable training data.
2. **Insufficient Coverage of Long-Tail Knowledge:** Since LLMs are optimized for token prediction, they tend to prioritize generating high-frequency, common knowledge while failing to capture rare, domain-specific information (Li et al., 2024b). This results in inadequate coverage of long-tail knowledge, which is crucial for knowledge-intensive applications (Kandpal et al., 2023; Li et al., 2024a).
3. **Superficial Knowledge Representation:** Existing synthetic data pipelines generate simplistic QA pairs that do not effectively model complex knowledge structures, such as multi-hop reasoning, where information must be linked across multiple sources to form a coherent answer.
4. **Homogenization and Overfitting Risks:** Synthetic datasets often suffer from low diversity, with repetitive sentence templates and similar difficulty levels. This lack of variation can lead to overfitting, reducing the generalization ability of fine-tuned models and, in extreme cases, causing catastrophic forgetting or model collapse (Shumailov et al., 2024).

Recent efforts have attempted to improve synthetic data generation by incorporating Monte Carlo tree search and chain-of-thought reasoning (Zhao et al., 2024b; Wei et al., 2022). However, these methods primarily focus on logical problem-solving and do not effectively adapt to knowledge-intensive tasks in closed-book scenarios.

To address these challenges, we propose GraphGen, a knowledge graph (KG)-calibrated data synthesis framework that systematically improves synthetic data quality through structured knowledge guidance. GraphGen is designed to enhance data generation in three key scenarios: **atomic QA** (covering basic knowledge), **aggregated QA** (incorporating complex, integrated knowledge), and **multi-hop QA** (extending to k-hop reasoning).

Specifically, we first construct a fine-grained KG from a source corpus. We then compute the expected calibration error (ECE) (Guo et al., 2017) for each triple in the KG to identify points where the model's confidence does not align with its actual accuracy, exposing potential *knowledge blind spots*. The framework prioritizes these high-ECE triples for targeted data augmentation. To ensure the contextual coherence of newly generated examples, we employ a k-hop neighborhood subgraph sampler with structural constraints. Lastly, we employ style-controlled guided generation to convert the sampled subgraphs into diverse QA pairs suited for SFT. Our experiments show that GraphGen consistently outperforms five established data synthesis baselines in the aforementioned three scenarios. In summary, our main contributions are:

- We propose GraphGen, a KG-based data synthesis framework designed to preserve knowledge associations while addressing limitations in coverage, which is effective for scenarios of atomic QA, aggregated QA, and multi-hop QA.
- We develop an ECE-driven module that identifies knowledge blind spots, enabling LLMs to focus on high-value, long-tail data.
- Through extensive evaluations, we demonstrate that GraphGen leads to more effective SFT on knowledge-intensive tasks under closed-book conditions than existing state-of-the-art methods.

2 Related Work

2.1 Knowledge Graph-based Data Generation

KGs provide structured representations of domain-specific information, enabling systematic modeling of entities and their relationships. Early KG-based data generation approaches relied on hand-crafted templates (Jia and Liang, 2016; Seyler et al., 2017), which, despite ensuring syntactic correctness, often produced repetitive and rigid outputs, limiting linguistic diversity and scalability.

To overcome these limitations, learning-based methods leveraging recurrent neural networks with attention mechanisms were introduced to generate fluent questions directly from KG triples (Indurthi et al., 2017; Du et al., 2017). More recent advancements, such as LFKQG (Fei et al., 2022), incorporated controlled generation techniques to improve entity coverage while fine-tuning for adaptability. However, ensuring factual consistency and generating diverse, high-quality text remain open challenges.

2.2 LLM-based Data Generation

LLMs have demonstrated remarkable generalization and reasoning capabilities across natural language tasks (Zhang and Yang, 2023; Maini et al., 2024; Köksal et al., 2024). In the area of data generation, it has been proposed to generate data using large language models to train smaller models (West et al., 2022). Unlike KG-driven methods, LLMs can generate diverse, human-like text without reliance on predefined templates (Liang et al., 2023). However, they often suffer from limited controllability and hallucination (Ji et al., 2023), leading to factual inconsistencies.

Efforts to mitigate these issues include multi-stage refinement pipelines such as Genie (Yehudai et al., 2024), which enhances factual accuracy and coherence. Despite these refinements, ensuring domain-specific precision at scale remains a challenge for standalone LLMs.

2.3 Combining LLMs and Knowledge Graphs

To enhance factual consistency, hybrid approaches integrating LLMs with KGs have been explored (Guo et al., 2024a; Zhao et al., 2024a; Yang et al., 2024). These methods leverage KGs to guide text generation, improving reliability while maintaining fluency. However, most focus on general text generation or question answering rather than synthetic data generation for SFT.

3 Problem Setup

Synthesizing Data from Raw Corpora We focus on approaches that transforms raw text corpora D_{source} into structured synthetic data D_{synth}. To

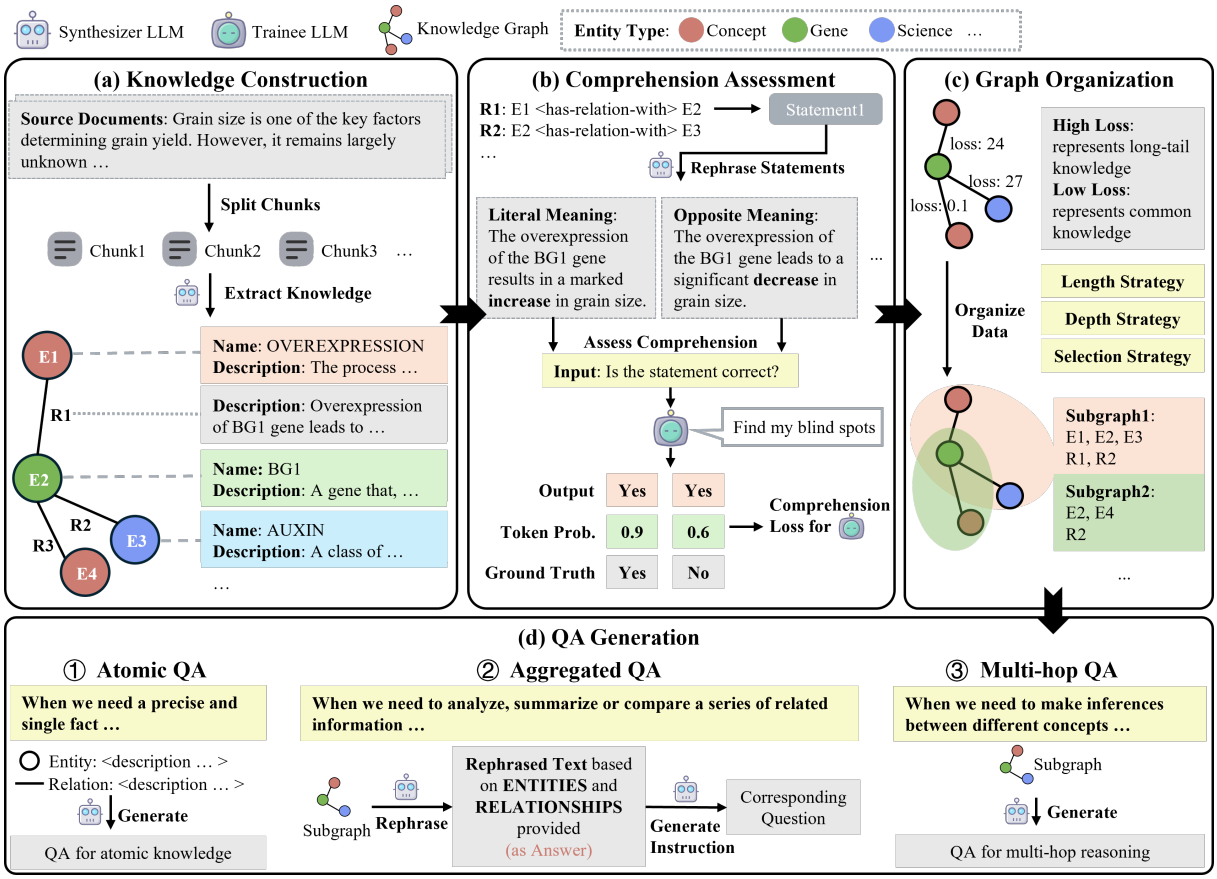

Figure 1: Pipeline of GraphGen. GraphGen optimizes LLM's performance by effectively organizing knowledge and identifying the specific data required for training the model. It comprises four core stages: (Step 1) Initially, entities/relationships are extracted to build a KG. (Step 2) Then, the Trainee Model's understanding of knowledge points is assessed. (Step 3) Then, subgraphs are formed for efficient training. (Step 4) Finally, subgraphs are converted into QA pairs for the three scenarios: atomic QA, aggregated QA and multi-hop QA.

achieve this, We propose a synthesis algorithm A_{synth} to generate data. Specifically, we utilize an algorithm A_{organize} that performs constrained graph traversal to extract subgraphs. The systematic workflow can be represented as follows:

$$A_{\text{synth}} : D_{\text{source}} \xrightarrow{A_{\text{extract}}} KG \xrightarrow{A_{\text{organize}}} D_{\text{synth}} \quad (1)$$

Evaluating the Quality of Synthetic Data The quality assessment of synthetic data necessitates both intrinsic quantitative analysis and validation through downstream tasks. We establish a set of multi-dimensional metrics $Met = \{\text{Metric}\}_{i=1}^{n}$ for data quality estimation. Additionally, we construct unbiased evaluation datasets D_{eval} to ensure task-specific validity. The performance on knowledge-intensive QA tasks under closed-book scenarios serves as critical evidence for testing whether the post-SFT model M_f has effectively acquired the knowledge in its parameters. The composite quality metric is formalized as:

$$Q_{D_{\text{synth}}} \propto (s(Met, D_{\text{synth}}), s(D_{\text{eval}}, M_f)) \quad (2)$$

where $s(Met, D_{\text{synth}})$ denotes the score of D_{synth} on the metrics, and $s(D_{\text{eval}}, M_f)$ indicates the performance of M_f on D_{eval}.

4 Method

In this section, we present GraphGen, a data synthesis framework, as illustrated in Figure 1. GraphGen is designed to generate data across three scenarios: atomic QA, aggregated QA, and multi-hop QA. From a knowledge organization perspective, these scenarios exemplify the most representative knowledge-intensive tasks in the context of closed-book QA. The framework comprises a four-step workflow involving two interdependent LLMs: the Synthesizer Model (M_{synth}) and the Trainee Model (M_{train}). M_{synth} possesses advanced general capabilities, as it is tasked with knowledge extraction

and rephrasing. In contrast, M_{train} serves as the target model that we aim to enhance in order to integrate additional knowledge. Detailed information regarding the prompt templates utilized in GraphGen, intermediate examples, and implementation details can be found in Appendix B.

STEP 1: Knowledge Construction Raw documents are segmented into smaller, semantically coherent fragments through context-aware chunking. Subsequently, M_{synth} are employed to identify and extract various entities and their relationships from these fragments. The types of entities to be extracted are predefined, with general categories including dates, locations, and events, while domain-specific categories encompass concepts such as genes. Each entity and relationship is assigned a description (*e.g.*, "SEED WEIGHT: The mass of seeds produced by a plant..."). During the extraction process, if the same entity or relationship appears in multiple fragments, their descriptions will be automatically combined together. Finally, cross-fragment entities and relationships are aggregated into a KG $G = (E, R)$. The combination of LLMs and KGs interrelates the *atomic* knowledge, addressesing challenges like long-text processing, format noise, and scattered knowledge distribution, while also ensuring a low rate of hallucination in the generated content (Ibrahim et al., 2024; Gillani et al., 2024). The specific implementation of STEP 1 is modified from the previous work (Guo et al., 2024b; Kong, 2025).

STEP 2: Comprehension Assessment We propose a method to assess whether the Trainee Model M_{train} have fully comprehended a knowledge point from the KG. For each edge in the KG, its description can be considered as a declarative statement R_i, which represents a knowledge point K_i that is unequivocally true, with a real-world probability of 1 (*i.e.*, $P(R_i \text{ is true}) = 1$). For example, the probability of the statement "water is a liquid at room temperature and pressure" being true is 1. To evaluate M_{train}'s understanding capabilities of these statements, we first generate multiple paraphrased statements $R_{i1}, R_{i2}, \ldots, R_{in}$ and their negations $\neg R_{i1}, \neg R_{i2}, \ldots, \neg R_{in}$ using M_{synth}. Unlike classification models, our focus in generative tasks is on accurately capturing semantic information for uncertainty modeling. Following the principle of ECE, a model is considered well-calibrated if its predicted confidence scores (*i.e.*, softmax probabilities) align with real-world probabilities of correctness. For LLMs, true understanding of a concept is achieved only when the model's confidence estimates match the actual likelihood of correctness in the real world. Therefore, we use a prompt (see Figure 2) to elicit M_{train}'s confidence in a single paraphrased statement. Then, by averaging the confidence scores from the n positive and n negative samples of R_i, M_{train}'s confidence in R_i is quantified via the following formula:

PROMPT: Please determine if the following statement is correct.

Note:
1. If the statement is correct, please reply with 'yes', otherwise reply with 'no'.
2. The answer should be either 'yes' or 'no', do not output any other content.

Statement:
{statement}
Judgement:

EXAMPLE OUTPUT: yes (Prob. 0.6)

Figure 2: Prompt for comprehension assessment. Through binary yes/no questions, we capture precise semantic information for confidence modeling, which can serve as an indicator of comprehension level when real-world probabilities are known

$$C_{R_i} = \frac{1}{2n}\left(\sum_{j=1}^{n} P(t|R_{ij}) + \sum_{j=1}^{n} P(f|\neg R_{ij})\right) \quad (3)$$

where $P(t|R_{ij})$ is the probability of the next token being "yes" given a true statement and $P(f|\neg R_{ij})$ denotes the probability of "no" in response to a false statement.

We further define a comprehension loss by calculating the cross-entropy between the true distribution and the predicted distribution:

$$\begin{aligned} \text{Loss}_{C_{R_i}} = &-\frac{1}{2n}\sum_{j=1}^{n} log(P(t|R_{ij})) \\ &-\frac{1}{2n}\sum_{j=1}^{n} log(P(f|\neg R_{ij})) \end{aligned} \quad (4)$$

which measures the gap between the LLM's current understanding and complete mastery of the knowledge point. By assessing the comprehension loss of M_{train}, we can systematically evaluate whether further training with these knowledge points is needed.

STEP 3: Graph Organization Subgraphs of KGs serve as the minimal unit for generating QA pairs. We perform k-hop subgraph extraction for effective graph organization, as detailed in Algorithm 1. To regulate the composition of these subgraphs, we implement several strategies. The depth strategy controls the k-hop depth, ensuring the subgraph spans a predefined number of hops from the start edge. For each candidate subgraph, we compute the premise length (denoted as pre_length), defined as the total number of tokens in the descriptions of entities and relationships within it. The length strategy enforces an upper bound on pre_length to maintain a balanced data distribution. When expanding the subgraph, we adopt a selection strategy with three options:

1. max_loss: Select edges with higher loss values, indicating greater uncertainty or potential information gain.
2. min_loss: Select edges with lower loss values, representing more confident or stable relations.
3. random: Select edges uniformly at random.

These strategies collectively balance subgraph complexity, relevance, and computational tractability.

STEP 4: QA Generation After extracting a subgraph, we can create three types of QA pairs based on its intended use. For the atomic QA scenario, the subgraph should consist of a single node or edge, allowing M_{synth} to generate a QA pair representing basic knowledge directly from its description. To analyze, summarize, or compare related information involving a set of entities and relationships within a subgraph, we prompt M_{synth} to organize and rephrase the data into a coherent text. Then, using the generated text as an answer, we use M_{synth} to generate its corresponding question. For multi-hop QAs, we first clarify the relationships between entities and then instruct M_{synth} to produce a QA pair that requires multi-step reasoning.

Algorithm 1 K-hop Subgraph Extraction

Input: Graph G, edge $R_i = (E_{\text{src}}, E_{\text{tgt}})$, graph organization strategies S
Output: Subgraph G'

$G' \leftarrow \{R_i\}$
$C \leftarrow \text{GETADJACENTEDGES}(G, \{E_{\text{src}}, E_{\text{tgt}}\})$
while $C \neq \emptyset$ **do**
  Select e from C according to S
  $G' \leftarrow G' \cup \{e\}$
  $C \leftarrow C \setminus \{e\}$
  if $\text{MEETSCONSTRAINTS}(G')$ **then**
    break
  for $v \in \text{GETENDPOINTS}(e)$ **do**
    $C \leftarrow C \cup \text{GETADJACENTEDGES}(G, v)$

return G'

5 Experiments

5.1 Experimental Setup

Domain Corpus and Evaluation Datasets Given that our method focuses on knowledge-intensive tasks under closed-book scenarios, we created three datasets, each corresponding to a key scenario that represents the most important knowledge-intensive tasks in the context of closed-book QA. We curated *SeedEval*[1], a domain-specific dataset designed for atomic QA. *SeedEval*related to seed knowledge (agriculture), covering one-shot and zero-shot scenarios. Additionally, we adapted the *PQArefEval* dataset from *PQAref* (Bašaragin et al., 2024), which is domain-specific and centers on medicine, constructing it for aggregated QA applications. Furthermore, we created *HotpotEval*, an adaptation of *HotpotQA* (Yang et al., 2018), intended for multi-hop QA tasks. Each dataset comprises two components: the QA test set (D_{eval}) and the corresponding source texts (D_{source}). See Appendix E for the source and details of the dataset.

Quality Evaluation Metrics We employ a set of natural language metrics (Cao et al., 2024) to evaluate the quality of generated text. Details are provided in Appendix F.2. Since most of these metrics are better suited for evaluating complete sentences than brief responses, we compared the aggregated QAs produced by GraphGen with those from baseline methods. The reward score averages scores from two reward models, labeled as *Ind* and *Deb*. The unieval score comprises three evaluation components from the UniEval model, denoted as *Nat*, *Coh*, and *Und*.

Baselines We modified the code for WRAP (Maini et al., 2024), Genie (Yehudai et al., 2024), LongForm (Köksal et al., 2024), EntiGraph (Yang et al., 2024), and SELF-QA (Zhang and Yang, 2023) to accommodate our data synthesis needs,

[1] https://anonymous.4open.science/r/GraphGen/resources/data/test/

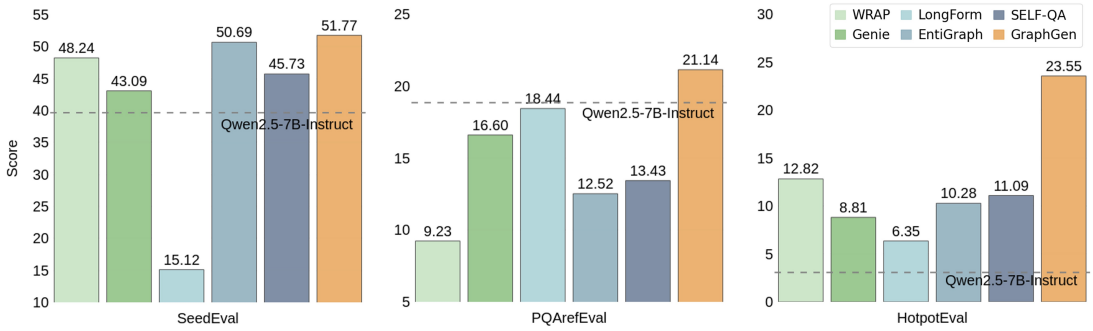

Figure 3: Performance comparison on knowledge-intensive evaluation datasets. We use data generated through various methods to optimize Qwen2.5-7B-Instruct. The baseline methods exhibit varying performance across the three datasets, while GraphGen consistently achieves optimal results.

Dataset	Domain	Scenario	Language	#Samples		Avg #Tokens		Max #Tokens	
				Corpus	Test	Corpus	Test	Corpus	Test
SeedEval	Agricultural	Atomic QA	English & Chinese	30,578	582	328	48	908	194
PQArefEval	Medical	Aggregated QA	English	58,078	5,815	357	518	1,837	1,680
HotpotEval	General	Multi-Hop QA	English	73,642	7,405	135	25	1,985	82

Table 1: Description of datasets employed for experiments. For calculating the token count, the tokenizer used is from Qwen2.5 series (Yang et al., 2025). The corpus is employed for graph construction and data synthesis, while the test set is utilized to evaluate the performance of the Post-SFT model trained with the synthesized data.

using them as the baselines for this study. See Appendix D for details of the baselines.

Implementation Details In this study, we specified M_{train} to be Qwen2.5-7B-Instruct[2] and M_{synth} to be Qwen2.5-72B-Instruct[3]. Two models are representative open-source LLM with robust performance and affordable computational cost. Considering the characteristics of the tasks associated with the three datasets, and to thoroughly validate our methodology, GraphGen generates atomic, aggregated, and multi-hop QA pairs for dataset *SeedEval*, *PQArefEval*, and *HotpotEval*, respectively. Additional setups can be found in Appendix C.

5.2 Performance Comparison

Results on Quality Evaluation Metrics We demonstrate that the metrics can be used to intuitively measure data quality. We compare the aggregated responses generated by GraphGen for the aggregated QA scenario with those from baseline methods. As shown in Table 2, on average, GraphGen outperforms the baselines. GraphGen excels in the MTLD metric due to its capability to aggregate cross-document knowledge, generating a significantly larger number of tokens compared to other methods that yield shorter QA responses. We note that graph-based methods lead the Uni-Score metrics, indicating that data generated through graph structures—particularly those illustrating multiple entity relationships—align more closely with everyday QA interactions. Notably, since LongForm directly uses D_{source} as the answer in a QA pair, it reflects the quality of D_{source}. Possibly influenced by its training corpus, the Deb metric exhibit a distinct bias towards the original text, which may not be suitable for more chaotic D_{source}.

Results on Evaluation Datasets We conducted SFT on M_{train} using generated data and evaluated M_f on corresponding test sets, with the results shown in Figure 3. Data generated by GraphGen brought the greatest performance improvement to the base model. Notably, we observed that on the *PQArefEval* dataset, the performance of baseline methods after training was inferior to their pre-training performance, which is counterintuitive. We hypothesize that this decline is due to the limitation of baseline methods, which use solely a single text segment for generating QA pairs when handling the aggregated QA task. Consequently, these models may lose their ability to form cross-document associations, negatively impacting their performance on tasks that require multiple refer-

[2] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

[3] https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

Method	#Samples	Avg #Tokens	Results						Avg Score
			MTLD	Uni			Rew		
				Nat	Coh	Und	Ind	Deb	
WRAP	476,626	32.4	13.4	91.2	87.1	91.6	44.0	4.0	42.5
Genie	56,938	83.8	40.2	91.7	94.7	92.7	64.1	44.0	62.4
LongForm	57,854	357.1	47.6	85.7	93.7	87.3	84.2	82.5	73.3
EntiGraph	532,971	47.4	30.1	92.6	93.3	93.1	56.3	28.8	55.2
SELF-QA	561,798	83.4	34.7	91.3	92.8	92.3	59.5	39.3	58.8
GraphGen	54,287	657.9	75.8	87.8	95.7	90.4	85.0	31.8	75.2

Table 2: Comparison results with other data synthesis methods on data quality evaluation metrics. The results indicate that the quality of data generated by GraphGen is comparatively high. The scores presented represent the average values derived from the generated datasets across the evaluated metrics. The top two performers in each column are highlighted.

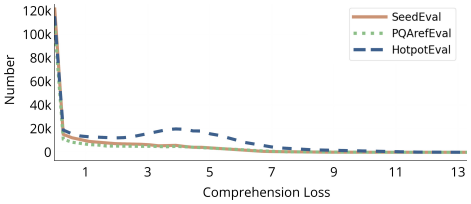

Figure 4: Distribution of comprehension loss for the Trainee Model. The model's comprehension loss is relatively low for the vast majority of data, which indicates most of the data generated by the Synthesizer Model has already been mastered by the Trainee Model.

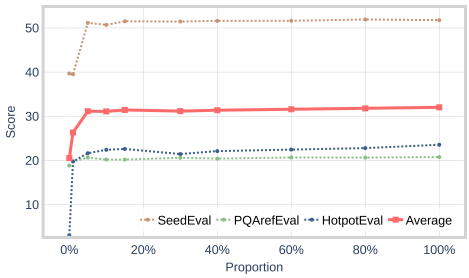

Figure 5: Performance comparison conducted with varying proportions of training data. The proportions are arranged in descending order based on loss. "Average" represents the mean score across three datasets. As the amount of training data increases, we observe a noticeable and consistent upward trend in the results.

ences. In contrast, M_f using data generated by GraphGen successfully addresses this challenge. Moreover, GraphGen's performance on the multi-hop QA scenario is particularly notable. This indicates that the knowledge associations derived from subgraphs enhance the multi-hop reasoning capabilities of the Post-SFT model, rather than merely enabling the acquisition of superficial knowledge.

It is noteworthy that at this stage, we only used knowledge-related data, without mixing general instruction-following data. The purpose was to highlight the role of generated data in injecting new knowledge. In addition, to alleviate concerns about overfitting caused by synthetic data, we mixed the generated data with 100,000 general instruction-following data and conducted tests on a broader evaluation dataset. See Appendix F for test results.

5.3 Analysis of Scaling Law

The scaling law of LLMs indicate that model performance improves with increasing amounts of training data (Kaplan et al., 2020). In this study, we obtained the comprehension loss for each knowledge point used in training the model. Through statistical analysis of the Loss_C for all knowledge points in the KG, we observed that the distribution of Loss_C is highly skewed, as illustrated in Figure 4. This finding supports the notion that M_{synth} has a preference for generating common knowledge, while the knowledge that M_{train} needs to acquire during training often resides in the rare, long-tail data. To further investigate the relationship between long-tail data and training effectiveness, we explored the scaling law of data generated by GraphGen. Similar to the concept of hard example mining (Shrivastava et al., 2016), we sorted the synthetic data in descending order of Loss_C and divided it into different proportions for sequential training to emphasize the importance of focusing on the most challenging instances. Surprisingly, we found that even when trained on only a small proportion of data (less than 5%), the model can still maintain a relatively high proportion of performance, as can be seen in Figure 5. As the total amount of training data increases, the overall score shows minimal improvement. Therefore, the head of the generated data contributes little new knowledge to the model.

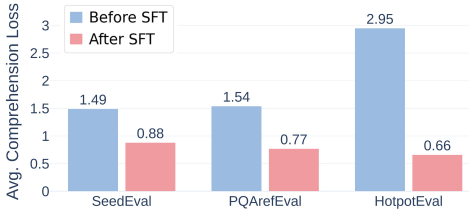

Figure 6: Comprehension loss of the Trainee Model. The reduction in loss after training highlights the effectiveness of data synthesis and the enhanced comprehension ability of the Trainee Model.

Additionally, we conducted a comparative experiment by training the model using the top 30% and bottom 30% of the data sorted according to Loss_C. The results showed that the model trained on the top 30% data achieved better performance than that trained on the bottom 30% data. This finding demonstrates that data with higher Loss_C can bring greater performance gains to the Trainee Model, as can be seen in Appendix F.1.

5.4 Comprehension Loss Change

After the SFT phase, we conducted a comprehension assessment on M_f. Although we did not directly use yes/no judgment questions as part of the training data, we observed a significant reduction in Loss_C, as can be seen in Figure 6. This indicates that GraphGen has enhanced M_{train}'s understanding of the knowledge domain, enabling it to reliably differentiate between correct and incorrect statements. Consequently, the model shows greater accuracy in knowledge-intensive tasks.

5.5 Ablation Studies

Selection of Entities and Relationships In our experiments for generating atomic QAs, we compared the effectiveness of using only entities versus only relationships as the sources for generation. The results indicate that relying solely on relationships outperformed using entities alone and even slightly exceeded the performance of using the entire KG. We attribute this to the fact that relationships more effectively encapsulate the intrinsic properties of knowledge. Additionally, the presence of overlap in knowledge organization within the KG may contribute to a decline in performance. The results can be seen in Appendix F.3.

Selection of Graph Organization Strategies We changed the length strategy of GraphGen by setting pre_length to 256, 512, 768, and 1024, and conducted evaluations on quality metrics for each case. The results can be seen in Appendix F.3. We found that, although the average length of the generated data increased, the final score tended to stabilize. This indicates that the score is not directly correlated with the data length. We also evaluated the data on D_{eval}, as can be seen in Appendix F.3. We found that although a longer pre_length may enhance the long-text ability of large models, the evaluation results with a pre_length of 256 were the best. The analysis of the length distribution of the final generated data revealed that a potential explanation lies in the characteristics of the data length distribution. Specifically, with a pre_length of 256, the distribution displays sharper traits for lengths below 5000. In contrast, an increased pre_length leads to a distribution with greater extension in length. An excessive amount of lengthy data may significantly prolong the convergence time required for the model. We also conducted experiments using the selection strategy as control variables. The analysis indicated that the influence of the strategy on the results was minimal, as demonstrated in Appendix F.3. This finding suggests that, as long as a correlation exists, variations in understanding levels within the subgraphs do not significantly impact the final outcomes. However, the underlying patterns merit further exploration in future research.

6 Conclusion

In this paper, we propose GraphGen, an effective KG-based approach to synthetic data generation for fine-tuning LLMs on knowledge-intensive tasks in closed-book QA settings. GraphGen is specifically designed to meet the needs of three scenarios: atomic QA, aggregated QA, and multi-hop QA. Experiments demonstrate that GraphGen successfully addresses limitations of existing synthetic data generation methods by leveraging KGs to guide the creation of high-quality QA pairs. This approach offers a promising solution to the data bottleneck in supervised fine-tuning.

Future research could focus on enhancing the knowledge graph construction by integrating external knowledge sources and dynamic updates to improve data quality and coverage. Additionally, exploring adaptive graph organization strategies and subgraph sampling methods could optimize the training data for better model performance.

Limitations

Despite the promising results exhibited by GraphGen, several limitations necessitate further investigation and enhancement. One significant concern is the framework's requirement for substantial computational resources when building and processing large-scale KGs. This demand may restrict its applicability in resource-constrained environments or when dealing with extensive datasets. Therefore, optimizing computational efficiency is vital for broader adoption.

While GraphGen has demonstrated strong performance across three representative domains, its adaptability to diverse fields remains to be explored. Current experiments have primarily focused on closed-book QA tasks, leaving the framework's generalization to other areas—such as mathematics, reasoning, and coding—largely unexamined. Furthermore, the integration of synthetic data generated by GraphGen into model training requires meticulous tuning. It is essential to investigate the balance between synthetic and real data, as well as their effects on model convergence and generalization.

In this article, we do not discuss open-book question answering, such as Retrieval-Augmented Generation (RAG). RAG's effectiveness is contingent upon the quality and recall capacity of the retrieval corpus, while failures in retrieval can potentially exacerbate the incidence of hallucinations. The integration of data synthesis with RAG methodologies signifies a promising avenue for future in-depth research.

References

Bojana Bašaragin, Adela Ljajić, Darija Medvecki, Lorenzo Cassano, Miloš Košprdić, and Nikola Milošević. 2024. How do you know that? teaching generative language models to reference answers to biomedical questions. *arXiv preprint*, arXiv:2407.05015.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2024. Instruction mining: Instruction data selection for tuning large language models. *arXiv preprint*, arXiv:2307.06290.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass. Accessed: 2025-02-13.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352.

Zichu Fei, Xin Zhou, Tao Gui, Qi Zhang, and Xuan-Jing Huang. 2022. Lfkqg: A controlled generation framework with local fine-tuning for question generation over knowledge bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6575–6585.

Khasa Gillani, Erik Novak, Klemen Kenda, and Dunja Mladenić. 2024. Knowledge graph extraction from textual data using llm.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330.

Shasha Guo, Lizi Liao, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024a. Sgsh: Stimulate large language models with skeleton heuristics for knowledge base question generation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4613–4625.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024b. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint*, arXiv:2410.05779.

Nourhan Ibrahim, Samar Aboulela, Ahmed Ibrahim, and Rasha Kashef. 2024. A survey on augmenting knowledge graphs (kgs) with large language models (llms): models, evaluation metrics, benchmarks, and challenges. *Discover Artificial Intelligence*, 4(1):76.

Sathish Reddy Indurthi, Dinesh Raghu, Mitesh M Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint*, arXiv:2001.08361.

Huanjun Kong. 2025. Huixiangdou2: A graph-based augmented generation approach. https://github.com/tpoisonooo/HuixiangDou2. Accessed: 2025-02-13.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2024. Longform: Effective instruction tuning with reverse instructions. *arXiv preprint*, arXiv:2304.08460.

Dongyang Li, Junbing Yan, Taolin Zhang, Chengyu Wang, Xiaofeng He, Longtao Huang, Hui Xue, and Jun Huang. 2024a. On the role of long-tail knowledge in retrieval augmented large language models. *arXiv preprint*, arXiv:2406.16367.

Huihan Li, Yuting Ning, Zeyi Liao, Siyuan Wang, Xiang Li, Ximing Lu, Wenting Zhao, Faeze Brahman, Yejin Choi, and Xiang Ren. 2024b. In search of the long-tail: Systematic generation of long-tail inferential knowledge via logical rule guided search. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2348–2370.

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

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, et al. 2024. Best practices and lessons learned on synthetic data for language models. *arXiv preprint*, arXiv:2404.07503.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On llms-driven synthetic data generation, curation, and evaluation: A survey. *arXiv preprint*, arXiv:2406.15126.

Keer Lu, Keshi Zhao, Zheng Liang, Da Pan, Shusen Zhang, Xin Wu, Weipeng Chen, Zenan Zhou, Guosheng Dong, Bin Cui, et al. 2024. Versatune: Fine-tuning multi-ability llms efficiently. *arXiv preprint*, arXiv:2411.11266.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint*, arXiv:2401.16380.

Philip M McCarthy and Scott Jarvis. 2010. Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior research methods*, 42(2):381–392.

Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, et al. 2024. Injecting new knowledge into large language models via supervised fine-tuning. *arXiv preprint*, arXiv:2404.00213.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint*, arXiv:2408.13296.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2017. Knowledge questions from knowledge graphs. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, page 11–18. ACM.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

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

Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2022. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2025. Qwen2.5 technical report. *arXiv preprint*, arXiv:2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint*, arXiv:1809.09600.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candès, and Tatsunori Hashimoto. 2024. Synthetic continued pretraining. *arXiv preprint*, arXiv:2409.07431.

Asaf Yehudai, Boaz Carmeli, Yosi Mass, Ofir Arviv, Nathaniel Mills, Assaf Toledo, Eyal Shnarch, and Leshem Choshen. 2024. Genie: Achieving human parity in content-grounded datasets generation. *arXiv preprint*, arXiv:2401.14367.

Xuanyu Zhang and Qing Yang. 2023. Self-qa: Unsupervised knowledge guided language model alignment. *arXiv preprint*, arXiv:2305.11952.

Runhao Zhao, Jiuyang Tang, Weixin Zeng, Ziyang Chen, and Xiang Zhao. 2024a. Zero-shot knowledge graph question generation via multi-agent llms and small models synthesis. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3341–3351.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024b. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint*, arXiv:2411.14405.

Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2023–2038.

A Additional System Modules

A.1 Entity Enrichment Module with Wikipedia

We have developed a plug-in module for GraphGen aimed at enriching entity information within the KG through targeted Wikipedia searches. Entities from D_{source} may initially contain only rudimentary descriptions. However, by leveraging the extensive and authoritative content of Wikipedia, we can provide a more comprehensive understanding of these entities, including historical context, definitions, and related facts. This enhancement not only enriches the content of the knowledge graph but also serves as a means to verify the accuracy of existing information, identifying potential errors or gaps within the data.

A.2 Coreference Resolution Module

We have developed an additional plug-in module for coreference resolution that processes text segments from D_{source}. By designating the first segment as a reference point, we analyze subsequent segments to identify any ambiguous pronouns. Utilizing a large language model (LLM), we generate responses that clarify these references based on the context established by the reference segment. This approach enables us to accurately identify and resolve pronouns and other referring expressions, thereby enhancing the clarity and coherence of the text segments.

A.3 User Interface

The user interface of GraphGen is designed to provide users with an intuitive tool for modifying and adjusting various settings. As can be seen in Figure 7, the settings include "Input Configuration", "Traverse Strategy" and "Model Configuration". Users can save their current parameter configurations as presets for easy retrieval in the future. This is especially useful for those who frequently adjust settings, enhancing efficiency.

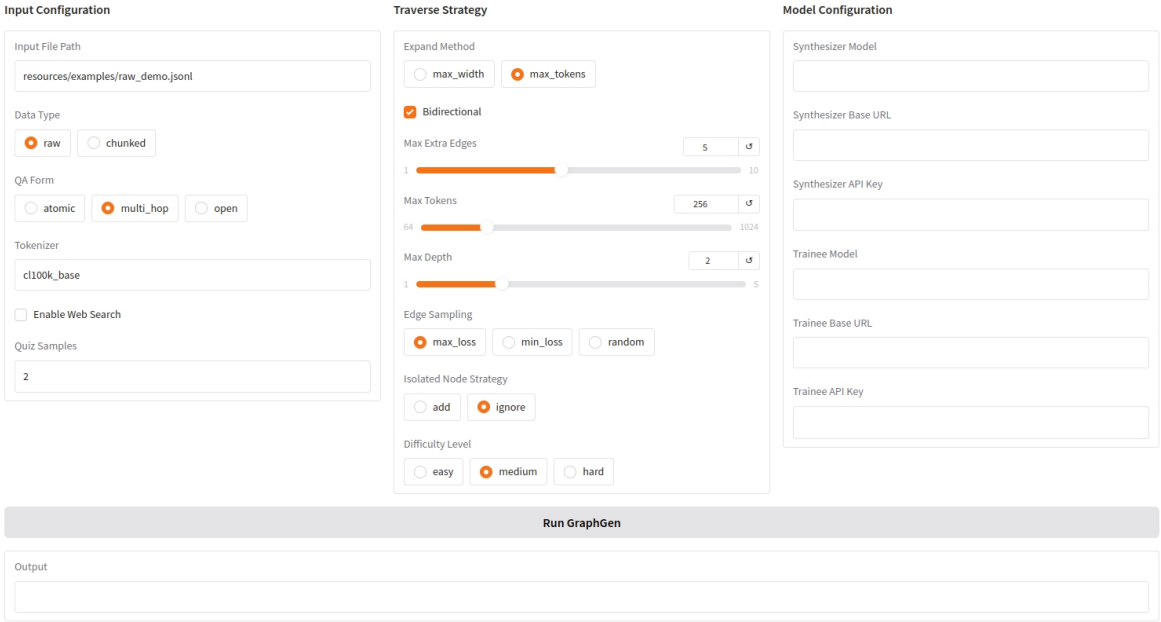

Figure 7: User interface of GraphGen. "Input Configuration" is utilized to specify the data sources and the target format. "Traverse Strategy" determines the method of graph organization. "Model Configuration" is employed to set parameters for the LLMs.

B GraphGen Details

B.1 Prompt Templates

In the GraphGen framework, we used the following prompt:

- Prompt for extracting entities and relationships of the KG (Figure 8).
- Prompt for summarizing multiple descriptions when the descriptions of an entity or relation come from various sources (Figure 9).
- Prompt for rephrasing the description into a positive or a negative statement (Figure 10 and 11).
- Prompt for atomic QA generation (Figure 12).
- Prompt for aggregated QA generation (Figure 13 and 14).
- Prompt for multi-hop QA generation (Figure 15).

You are an NLP expert, skilled at analyzing text to extract named entities and their relationships.

-Goal-
Given a text document that is potentially relevant to this activity and a list of entity types, identify all entities of those types from the text and all relationships among the identified entities.
Use {language} as output language.

-Steps-
1. Identify all entities. For each identified entity, extract the following information:
- entity_name: Name of the entity, use same language as input text. If English, capitalized the name.
- entity_type: One of the following types: [{entity_types}]
- entity_summary: Comprehensive summary of the entity's attributes and activities
Format each entity as
("entity"{tuple_delimiter}<entity_name>{tuple_delimiter}<entity_type>{tuple_delimiter}<entity_summary>)

2. From the entities identified in step 1, identify all pairs of (source_entity, target_entity) that are *clearly related* to each other.
For each pair of related entities, extract the following information:
- source_entity: name of the source entity, as identified in step 1
- target_entity: name of the target entity, as identified in step 1
- relationship_summary: explanation as to why you think the source entity and the target entity are related to each other
Format each relationship as
("relationship"{tuple_delimiter}<source_entity>{tuple_delimiter}<target_entity>{tuple_delimiter}<relationship_summary>)

3. Identify high-level key words that summarize the main concepts, themes, or topics of the entire text. These should capture the overarching ideas present in the document.
Format the content-level key words as
("content_keywords"{tuple_delimiter}<high_level_keywords>)

4. Return output in {language} as a single list of all the entities and relationships identified in steps 1 and 2. Use **{record_delimiter}** as the list delimiter.

5. When finished, output {completion_delimiter}

######################
-Examples-
######################
{examples}

######################
-Real Data-
######################
Entity_types: {entity_types}
Text: {input_text}
######################
Output:

Figure 8: Prompt for KG extraction.

You are an NLP expert responsible for generating a comprehensive summary of the data provided below.
Given one entity or relationship, and a list of descriptions, all related to the same entity or relationship.
Please concatenate all of these into a single, comprehensive description. Make sure to include information collected from all the descriptions.
If the provided descriptions are contradictory, please resolve the contradictions and provide a single, coherent summary.
Make sure it is written in third person, and include the entity names so we the have full context.
Use {language} as output language.

#######
-Data-
Entities: {entity_name}
Description List: {description_list}
#######
Output:

Figure 9: Prompt for KG summarization.

-Goal-
Transform the input sentence into its opposite meaning while:

1. Preserving most of the original sentence structure
2. Changing only key words that affect the core meaning
3. Maintaining the same tone and style
4. The input sentence provided is a right description, and the output sentence should be a wrong description
5. The output sentence should be fluent and grammatically correct

#################
-Examples-
#################
Input:
The bright sunshine made everyone feel energetic and happy.

Output:
The bright sunshine made everyone feel tired and sad.

#################
-Real Data-
#################
Input:
{input_sentence}
#################
Please directly output the rewritten sentence without any additional information.
Output:

Figure 10: Prompt for description rephrasing (opposite meaning).

-Goal-
Transform the input sentence into a sentence with the same meaning while:

1. Preserving most of the original sentence structure
2. Changing only key words that affect the core meaning
3. Maintaining the same tone and style
4. The output sentence should be fluent and grammatically correct

################
-Examples-
################
Input:
The bright sunshine made everyone feel energetic and happy.

Output:
The bright sunshine made everyone feel energetic and joyful.

################
-Real Data-
################
Input:
{input_sentence}
################
Please directly output the rewritten sentence without any additional information.
Output:

Figure 11: Prompt for description rephrasing (literal meaning).

You are given a text passage. Your task is to generate a question and answer (QA) pair based on the content of that text.
The answer should be accurate and directly derived from the text. Make sure the QA pair is relevant to the main theme or important details of the given text.
For example:
Question: What is the effect of overexpressing the BG1 gene on grain size and development?
Answer: Overexpression of the BG1 gene leads to significantly increased grain size, demonstrating its role in grain development.

Question: What role does TAC4 play in the gravitropism of rice shoots?
Answer: TAC4 is a key regulator of gravitropism in rice shoots, promoting the bending of shoots towards the gravity vector.

Here is the text passage you need to generate a QA pair for:
{doc}

Figure 12: Prompt for atomic QA generation.

---Role---

You are an NLP expert responsible for generating a logically structured and coherent rephrased version of the TEXT based on ENTITIES and RELATIONSHIPS provided below.
Use {language} as output language.

---Goal---
To generate a version of the text that is rephrased and conveys the same meaning as the original entity and relationship descriptions, while:
1. Following a clear logical flow and structure
2. Establishing proper cause-and-effect relationships
3. Ensuring temporal and sequential consistency
4. Creating smooth transitions between ideas using conjunctions and appropriate linking words like "firstly," "however," "therefore," etc.

---Instructions---
1. Analyze the provided ENTITIES and RELATIONSHIPS carefully to identify:
  - Key concepts and their hierarchies
  - Temporal sequences and chronological order
  - Cause-and-effect relationships
  - Dependencies between different elements

2. Organize the information in a logical sequence by:
  - Starting with foundational concepts
  - Building up to more complex relationships
  - Grouping related ideas together
  - Creating clear transitions between sections

3. Rephrase the text while maintaining:
  - Logical flow and progression
  - Clear connections between ideas
  - Proper context and background
  - Coherent narrative structure

4. Review and refine the text to ensure:
  - Logical consistency throughout
  - Clear cause-and-effect relationships

################
-ENTITIES-
################
{entities}

################
-RELATIONSHIPS-
################
{relationships}

################
Please directly output the coherent rephrased text below, without any additional content.

Rephrased Text:

Figure 13: Prompt for aggregated answer rephrasing.

The answer to a question is provided. Please generate a question that corresponds to the answer.

################
Answer:
{answer}
################
Question:

Figure 14: Prompt for question generation (aggregated QA).

Please generate a multi-hop reasoning question and answer based on the following knowledge subgraph. You will be provided with a knowledge subgraph that contains a series of entities, relations, and facts. Your task is to generate a question that requires multiple steps of reasoning to answer. The answer to the question should be inferred from the given knowledge subgraph. Ensure that the question is of moderate difficulty and requires multiple steps of reasoning to answer.

For example:
########
--Entities--
1. Apple
2. Fruit
3. Vitamin C
########
--Relations--
1. Apple-Fruit: Apple is a type of fruit
2. Fruit-Vitamin C: Fruits are rich in Vitamin C
########
Question: What substance, obtained through eating apples, helps maintain health?
Answer: Vitamin C
########

########
--Entities--
{entities}
########
--Relations--
{relationships}
########
Output the generated question and answer directly, please do not copy the example question and answer directly, and do not provide irrelevant information.

Figure 15: Prompt for multi-hop QA generation.

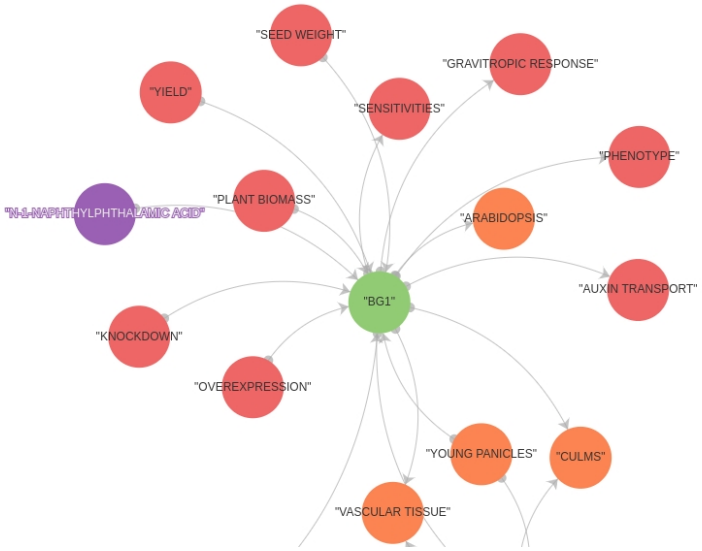

Figure 16: An example of the extracted KG. Different colors represent different entity types.

B.2 Examples

Here we present some output examples from the GraphGen workflow. Figure 16 shows an example of the extracted KG. In the graph, entities are interconnected based on the relationships obtained from D_{source}, and each entity or relation has its own description.

Figure 17 illustrates the three styles of data generated by GraphGen. We can clearly observe that atomic QA focuses on simple, single knowledge points, while aggregated QA generates a coherent and logical long answer within complex subgraphs, thereby producing more complex and comprehensive responses. Multi-hop QA, on the other hand, emphasizes reasoning and connecting multiple knowledge points. These methods each demonstrate different levels of knowledge extraction and semantic understanding capabilities.

Example of Atomic QA

Question
How does the overexpression of the BG1 gene affect the yield of rice crops?
Answer
The overexpression of the BG1 gene leads to increased grain size, which is crucial for improving crop yield.

Example of Aggregated QA

Question
What role does the BG1 gene play in enhancing plant biomass and regulating growth directionality in rice plants?
Answer
The total mass of living plant material, known as plant biomass, can be enhanced through the manipulation of the BG1 gene. This gene, when overexpressed, not only leads to an increase in grain size but also plays a crucial role in regulating auxin transport. **Consequently,** the expression of BG1 significantly influences the gravitropic response of plants, which is their directional growth in response to gravity. This relationship establishes a clear link between gene function and the directionality of plant growth.

Furthermore, BG1 is specifically expressed in the young panicles, which are the developing flower clusters of rice plants. This suggests that the gene is vital for the proper development of the reproductive structures in these plants. **Additionally,** the expression of BG1 alters the sensitivities of plants to auxin, a key hormone, indicating its critical role in hormonal regulation. **Therefore,** manipulating the BG1 gene not only enhances plant biomass but also affects growth directionality and reproductive structure development, showcasing its multifaceted impact on overall plant growth.

Example of Multi-hop QA

Question
How does the expression of the BG1 gene influence the grain size phenotype in plants, and what role does auxin transport play in this relationship?
Answer
The expression of the BG1 gene influences the grain size phenotype **by regulating auxin transport**, which is essential for proper plant growth and development. Increased expression of BG1 **leads to** enhanced auxin transport, **resulting in** larger grain size in plants.

Figure 17: Examples of the GraphGen data. The words indicating contrasts or clear logical relationships between knowledge points is highlighted.

B.3 Implementation Details

Here we present the complete configuration of GraphGen's organization strategy, as can be seen in Table 3.

Parameter	Default Value	Description	Options
qa_form	atomic	Type of QA form desired.	• atomic: single-step • multi_hop: multi-step • aggregated: open-ended
expand_method	max_tokens	Method for controlling graph expansion.	• max_width: limit by number of edges in the subgraph • max_tokens: limit by token length of entities and relations in the subgraph
bidirectional	True	Expanding the graph in both directions (True) or one direction (False).	
max_extra_edges	5	Maximum number of edges to expand.	
max_tokens	256	Maximum number of tokens.	
max_depth	2	Maximum depth for traversal in each direction.	
edge_sampling	max_loss	Strategy for edge selection at the same layer.	• max_loss: prioritize highest loss edges • min_loss: prioritize lowest loss edges • random: random selection
isolated_node_strategy	add	Handling strategy for isolated nodes.	• add: include isolated nodes • ignore: exclude isolated nodes

Table 3: Configuration of the graph organization strategy.

C Additional Setups

In this section, we present the detailed configurations for generating, training and evaluation settings. Table 5 provides the hyperparameters employed during training, while Table 4 outlines the parameters used in our evaluation pipeline. The time required for processing varies with the size of the dataset and changes in the graph organization strategy. On average, generating a batch of approximately 50,000 data entries takes about 2 hours, while SFT on Qwen2.5-7B-Instruct requires around 1 hour and evaluation takes about 10 minutes, utilizing 8 NVIDIA A100 40GB GPUs.

When generating data, for M_{synth}, we set the following parameters: topk $= 50$, topp $= 0.95$, repetition_penalty $= 1.05$, max_tokens $= 10240$, and temperature $= 0$. It is noteworthy that when rephrasing descriptions, the temperature is adjusted to 1 for diverse expressions. When judging statements, for M_train, we need to obtain the softmax probabilities of the output tokens. Therefore, we set the parameters as follows: logprobs $=$ True, top_logprobs $= 5$, and max_tokens $= 1$.

For evaluation, we leverage the OpenCompass framework (Contributors, 2023) as a standardized evaluation toolkit. The evaluation process is controlled through key hyperparameters that dictate model behavior, including sequence length constraints, batch processing, and sampling configurations. The detailed parameter settings are presented in Table 4.

Parameter	Value	Description
Maximum Sequence Length	7168	The maximum length of input tokens the model can process in a single instance.
Maximum Output Length	2048	The maximum number of newly generated tokens allowed in model output.
Batch Size	80	The maximum number of prompts that LMDeploy receives in the 'generate' function.
Temperature (Gen.)	0	Controls randomness in sampling; lower values lead to more deterministic outputs (greedy search). A value of 0 enforces fully deterministic generation.

Table 4: Evaluation configuration parameters.

In the training phase, we employ a transformer-based architecture with the AdamW optimizer. The learning rate is linearly scheduled with a warm-up phase, and gradient clipping is applied to stabilize training. These configurations ensure effective optimization and robust model convergence.

D Baseline Details

Among the baseline methods, WRAP, Genie, LongForm, and SELF-QA are generation methods based on prompt engineering, while EntiGraph is based on KG. Specifically, LongForm utilizes the text segments from D_{source} directly as answers in QA pairs, subsequently generating corresponding questions based on these answers. Genie feeds raw text into a LLM to produce a QA pair. WRAP also extracts QA pairs from raw text but varies in the number of QA pairs generated for each text segment. SELF-QA involves two critical steps: first, it generates ten questions based on the original text, and then it answers these questions contextually, yielding a total of ten QA pairs. EntiGraph begins by extracting entities from the text, then combines these entities in pairs or triplets to create QA pairs, informed by the analysis of the original text. To optimize performance and prevent the generation of excessive, redundant information that could waste computational resources, we implemented a limit on the number of entities selected by EntiGraph during its execution.

Parameter	Value	Description
Maximum Length	2048	Maximum sequence length for model inputs.
Learning Rate	2e-5	Step size for weight updates, controlling optimization speed.
Weight Decay	0.1	Regularization term to mitigate overfitting by penalizing large weights.
Gradient Clipping	1	Caps gradient norm to stabilize training and prevent exploding gradients.
Batch Size	64 (16×4)	Total number of samples processed per optimization step.
Optimizer	AdamW	Variant of Adam with decoupled weight decay for improved generalization.
Betas (β_1, β_2)	(0.9, 0.999)	Exponential decay rates for first and second moment estimates in Adam.
Warmup Ratio	0.03	Fraction of total training steps used for gradual learning rate ramp-up.
Number of Epochs	2	Total number of complete passes over the training dataset.

Table 5: Training configuration parameters.

E Dataset Details

SeedEval is adapted from *SeedBench* [4], a benchmark with 11 tasks related to seed knowledge. For this study, we selected Task QA–4 (covering one-shot and zero-shot scenarios) related to textual knowledge question answering. *PQArefEval* is derived from *PQAref*, from which we extracted 5,818 instances for our analysis. *HotpotQA* is a dataset for diverse, explainable multi-hop question answering, where questions require integrating information from multiple sources. We used the test set of *HotpotQA* as the new evaluation dataset, *HotpotEval*. Each dataset comprises two components: the QA test set (D_{eval}) and the corresponding source texts (D_{source}). The Corpus for *SeedEval* is provided by anonymous agricultural experts, while *PQArefEval* and *HotpotEval* are constructed from the original references of *PQAref* and *HotpotQA*, respectively. Table 6 presents examples from the datasets.

[4] https://anonymous.4open.science/r/SeedBench

Dataset	Examples	
	Corpus	Test
SeedEval	Grain size is one of the key factors determining grain yield. However, it remains largely unknown how grain size is regulated by developmental signals. Here, we report the identification and characterization ...	**Question:** What were the findings regarding the effect of elevated DEP1 accumulation on grain length? **Answer:** Elevated DEP1 accumulation increased the grain length by 6.85–9.58% with a normal plant stature.
PQArefEval	Clinical analysis of the acromial height-measuring device combined with new-type clavicular hook plate and standard clavicular hook plate in the treatment of Neer type II distal clavicle fractures. Background: Distal clavicular fracture is a shoulder joint injury that is common in clinical settings and is generally surgically treated using the clavicular hook plate technique with a confirmed curative effect ...	**Question:** Is Bridging Necessary? **Answer:** The necessity of bridging depends on the context in which the term is used. In the context of myocardial bridging, it is not a procedure that is done but rather a congenital condition where a segment of a coronary artery takes an intramyocardial course, which can be asymptomatic or lead to complications such as ...
HotpotEval	Ed Wood is a 1994 American biographical period comedy-drama film directed and produced by Tim Burton, and starring Johnny Depp as cult filmmaker Ed Wood ...	**Question:** The Vermont Catamounts men's soccer team currently competes in a conference that was formerly known as what from 1988 to 1996? **Answer:** the North Atlantic Conference

Table 6: Dataset examples.

F Additional Experimental Results

F.1 Effect of Comprehension Loss on Model Performance

Figure 18 compares model performance when trained on the top 30% and bottom 30% of data sorted by comprehension loss. The results indicate that models trained on higher-loss data achieve better performance, suggesting that such data contributes positively to model optimization and generalization.

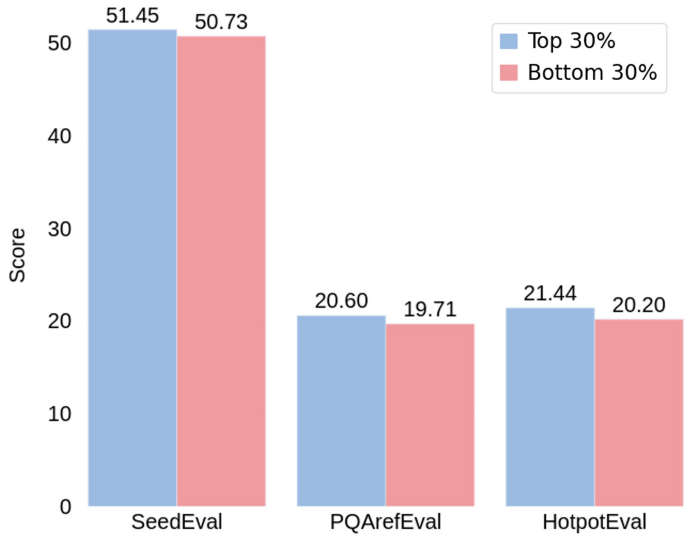

Figure 18: Comparison of model performance trained on top 30% and bottom 30% data sorted by comprehension loss. This figure demonstrates that the model trained on data with higher loss (top 30%) outperforms the one trained on data with lower loss (bottom 30%), highlighting the positive impact of high-loss data on model performance enhancement.

F.2 Quality Evaluation Metric Details

Table 7 presents the metrics used to evaluate the intrinsic quality of the text. To normalize the metric scores to a range of 0 to 100, we employ min-max normalization as specified in Formula 5. In this analysis, the minimum and maximum values for MTLD are established at 0 and 200, respectively. The three metrics included in the Uni-Score are scaled from 0 to 1. For the Reward Score, the range for *Ind* is 0–5, while for *Deb*, it spans from 0 to 3. We use Formula 6 to compute the final average score S_{Avg}, which provides a comprehensive assessment of data quality.

$$S(x) = \frac{x - x_{min}}{x_{max} - x_{min}} \times 100 \tag{5}$$

$$\begin{aligned} S_{Uni} &= \frac{S_{Nat} + S_{Coh} + S_{Und}}{3} \\ S_{Rew} &= \frac{S_{Ind} + S_{Deb}}{2} \\ S_{Avg} &= \frac{S_{MTLD} + S_{Uni} + S_{Rew}}{3} \end{aligned} \tag{6}$$

F.3 Ablation Study on the Selection Strategy

Tables 8 and 9 present the results of an ablation study evaluating different edge selection strategies for GraphGen on the *PQArefEval* and *HotpotEval* datasets. The study compares the following strategies: max_loss, min_loss, and random. Performance is measured using the ROUGE-F metric.

Metric	Notation	Explanation
MTLD	$MTLD$	The metric for assessing lexical diversity in texts (McCarthy and Jarvis, 2010).
Unieval Score	Uni	Naturalness, coherence and understandability score provided by the UniEval dialogue model (Zhong et al., 2022).
Reward Score	Rew	The reward model inference score of QA pairs. Reward models in this paper include *BAAI/IndustryCorpus2_DataRater*[1] and *OpenAssistant/reward-model-deberta-v3-large-v2*[2].

[1] https://huggingface.co/BAAI/IndustryCorpus2_DataRater
[2] https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2

Table 7: Key metrics for evaluating the quality of generated text. The table provides a summary of the notation and explanations for each metric.

Method	Selection Strategy	Score
GraphGen (256)	max_loss	21.14
	min_loss	21.22
	random	21.06
GraphGen (512)	max_loss	20.72
	min_loss	20.92
	random	20.93
GraphGen (768)	max_loss	20.46
	min_loss	20.47
	random	20.64
GraphGen (1024)	max_loss	20.50
	min_loss	20.56
	random	20.40

Table 8: Ablation Study on Different Selection Strategies of GraphGen on the *PQAref* Dataset. The model is evaluated under different sequence length settings (pre_length) and three distinct generation strategies: random, min_loss, and max_loss.

Method	Selction Strategy	Score
GraphGen	max_loss	23.55
	min_loss	23.79
	random	23.49

Table 9: Performance of different generation strategies on the Hotpot dataset. The performance is measured using ROUGE-F.

Figure 19 20 21 presents the character length distributions of answers in a medical QA dataset, comparing three sampling strategies (maximum loss, minimum loss, and random selection) across four token length constraints (256-1024).

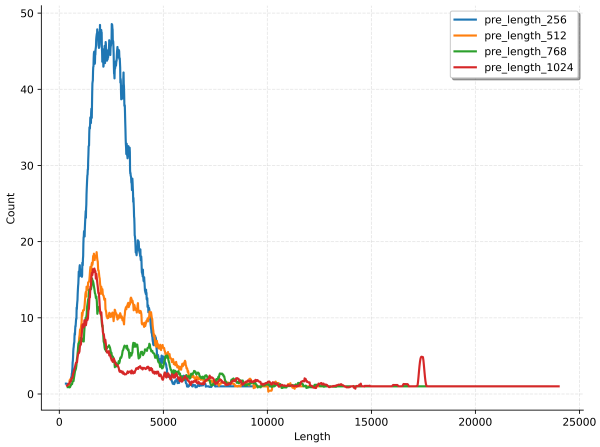

Figure 19: Character length distribution under the maximum loss sampling strategy.

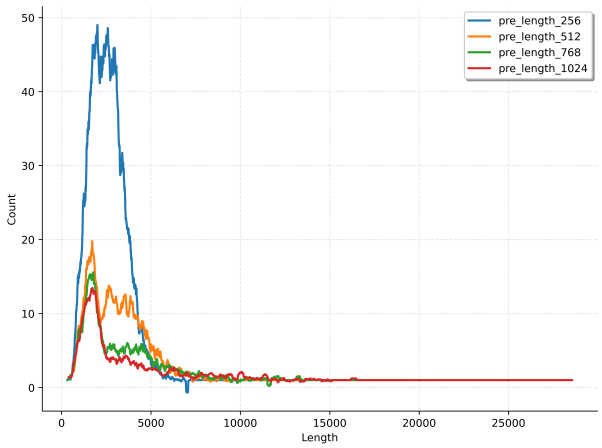

Figure 20: Character length distribution under the minimum loss sampling strategy.

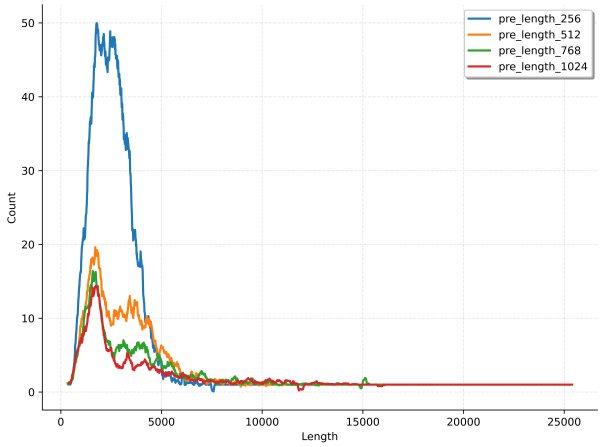

Figure 21: Character length distribution under the random selection strategy.

Table 10 presents the results of an ablation study on different sequence length settings in GraphGen, evaluating their impact on various quality metrics. The results show that increasing sequence length generally improves lexical diversity while maintaining consistent performance across evaluation metrics.

Method	**#Samples**	**Avg #Tokens**	**Results**						**Avg Score**
			MTLD	**Uni**			**Rew**		
				Nat	Coh	Und	Ind	Deb	
GraphGen(256)	119501	532.6	71.9	87.8	95.7	90.4	84.6	38.0	74.8
GraphGen(512)	54287	657.9	75.8	87.8	95.7	90.4	85.0	31.8	75.2
GraphGen(768)	38246	718.1	75.8	87.9	95.9	90.4	84.8	31.0	75.0
GraphGen(1024)	32137	749.3	75.0	87.9	95.9	90.5	84.5	31.7	74.8

Table 10: Ablation study on quality metrics.

F.4 Ablation Study on Knowledge Representation Strategy

Table 11 presents an ablation study evaluating different knowledge representation strategies for **GraphGen** on the agricultural dataset. The study compares three configurations—using only entities, only relations, and both entities and relations. Performance is measured using the ROUGE-F metric to assess the impact of different knowledge structures on model effectiveness.

Method	Knowledge Representation Strategy	ROUGE-F
GraphGen	Only Entities	50.68
	Only Relations	51.88
	Entities + Relations	51.77

Table 11: Ablation study on different knowledge representation strategies for GraphGen on the agricultural dataset. The model is evaluated under three different configurations: using only entities, only relations, and both entities and relations.

G Evaluation on General and Agricultural Tasks

Table 12 presents the evaluation results of different models on both general tasks and the SeedBench agricultural benchmark. The study examines six model variants, including **GraphGen**, **EntiGraph**, **Genie**, **LongForm**, **Self-QA**, and **Wrap**. Performance is reported across multiple metrics, including GPQA, CMLU, GSM8K, BBH, MATH, Lukaemon, and various SeedBench scores.

Metric	GraphGen	EntiGraph	Genie	LongForm	SELF-QA	WRAP
General Benchmarks						
GPQA	33.84	27.78	29.80	30.81	34.85	33.84
CMLU	77.61	77.56	78.58	77.42	77.54	77.89
GSM8K	80.89	79.45	80.44	80.74	80.21	81.20
BBH	68.51	67.72	67.80	67.57	66.92	66.57
MATH	52.45	52.14	53.73	52.91	52.99	54.71
Lukaemon	73.56	71.11	73.61	72.77	72.43	70.55
Agricultural Benchmarks (SeedBench)						
QA-1	61.75	65.25	61.25	61.00	66.25	63.25
QA-2	75.67	74.49	75.35	72.59	75.02	74.09
QA-3	22.46	30.15	24.69	24.37	26.43	26.64
QA-4	50.71	51.61	51.91	49.48	49.10	48.76
SUM-1	52.59	61.80	58.95	66.35	58.53	57.11
SUM-2	52.20	63.90	65.30	71.48	65.44	63.68
RC-1	96.91	96.91	96.02	96.46	96.46	96.02
RC-2	96.67	87.96	88.63	90.41	89.09	90.08
RC-3	77.82	83.66	83.99	84.45	83.74	83.97
RC-4	63.34	71.78	70.95	72.22	65.44	66.81
RC-5	75.27	77.60	74.91	76.35	75.81	76.17

Table 12: Evaluation results of different models on general tasks and the SeedBench agricultural benchmark. General benchmarks are listed first, followed by agricultural benchmarks. In SeedBench, QA-1 corresponds to multiple-choice questions, QA-2 refers to multiple-answer questions, QA-3 involves fill-in-the-blank tasks, and QA-4 pertains to open-ended generative questions. SUM-1 represents simple summarization tasks, while SUM-2 focuses on key information extraction. RC-1 denotes multiple-choice reading comprehension, RC-2 covers multiple-answer reading comprehension, RC-3 consists of fill-in-the-blank reading comprehension tasks, RC-4 includes generative reading comprehension, and RC-5 represents subcategory classification tasks.