STORM-BORN: A Challenging Mathematical Derivations Dataset Curated via a Human-in-the-Loop Multi-Agent Framework

Anonymous ACL submission

Abstract

High-quality math datasets are essential for advancing the reasoning capabilities of large language models (LLMs). However, current datasets face three major issues: (i) outdated and insufficient challenging content to match the rapid advancement of LLMs, (ii) an overemphasis on strict step-by-step derivations, neglecting human-like reasoning, and (iii) limited reliability from single-agent synthetic generation. To address these challenges, we introduce **STORM-BORN**, a dataset of challenging mathematical derivations derived from the latest and most influential academic papers. Unlike conventional numerical reasoning or formalized proof, STORM-BORN focuses on natural language mathematical derivations that include dense human-like approximations and heuristic cues. To ensure the reliability and quality of the dataset, we propose a novel human-in-the-loop, multi-agent data generation framework, integrating reasoning-dense filters, multi-agent collaboration, and human mathematicians' evaluations. We curates a set of 2,000 synthetic samples, from which 100 most challenging and high-quality problems are selected via human experts. Empirical evaluations reveal that state-of-the-art AI models, such as GPT-o1, solve fewer than 5% of the STORM-BORN problems, underscoring the dataset's inherent difficulty. As AI approaches mathematician-level reasoning, STORM-BORN offers a novel, challenging, and reliable resource to mimic human-like reasoning and serves as a high-difficulty evaluation benchmark.

1 Introduction

Mathematical reasoning has emerged as a cornerstone for scaling large language models (LLMs) and probing their upper bounds of intelligence (Shao et al., 2024; Ye et al., 2024; Glazer et al., 2024). Recent advances stem from architectural innovations (McLeish et al., 2024), enhanced pretraining data (Shao et al., 2024; Allal et al., 2025; Wang et al., 2024b), supervised fine-tuning (Yu et al., 2024b; Cobbe et al., 2021a), reinforcement learning (Wang et al., 2024a; Zelikman et al., 2022), and chain-of-thought prompting (Ye et al., 2024; Zhang et al., 2022). Current supervised fine-tuning mathematical datasets can be divided into two categories: numerical reasoning focuses on arithmetic computations that always yield a number (Cobbe et al., 2021a; Hendrycks et al., 2021; Glazer et al., 2024), and theorem proving uses formal languages to produce computer-verifiable proofs (Ying et al., 2024; Wu et al., 2024).

However, existing mathematical datasets still faces several challenges: **(C1) Lack of nuance and complexity.** Current datasets, often limited to grade-school calculations or competition problems(Cobbe et al., 2021a; Hendrycks et al., 2021), oversimplify mathematical reasoning, highlighting the need for more complex and nuanced problems, especially as LLMs approach perfect performance on these benchmarks. **(C2) Limited human-like reasoning.** While formal languages like Lean (de Moura et al., 2015) enable precise verification in recent dataset (Ying et al., 2024; Wu et al., 2024), they obscure intuitive human-like reasoning processes, which are more valuable and interpretable compared to strict and formal derivations (Chervonyi et al., 2025; Glazer et al., 2024). **(C3) Unreliable annotations.** LLMs are often used for automatic data synthesis to scale data (Yu et al., 2024b; Shao et al., 2024). However, this approach often leads to unreliable annotations, especially in step-by-step reasoning tasks, due to LLMs' tendency to hallucinate or make logical errors.

To address these limitations, we introduce **STORM-BORN**, a dataset of challenging mathematical derivations derived from recent top-tier academic papers (see Fig. 1 for examples). To curate nuanced and challenging dataset **(C1)**, we

(I) Numerical Reasoning (PRM-800K)

Question: An unfair coin lands on heads with probability $^3/_4$ and tails with probability $^1/_4$. A heads flip gains \$3, but a tails flip loses \$8. What is the expected worth of a coin flip? Express your answer as a decimal rounded to the nearest hundredth.

Steps: The expected value is equal to the probability of getting heads times the value of heads plus the probability of getting tails times the value of tails. Let's call the expected value E. So we can write that as $E = {^3/_4}*3 + {^1/_4}*(-8)$. That simplifies to $E = {^9/_4} - 2$. So the expected value is $E = {^1/_4} = 0.25$ dollars.

Answer: \$0.25.

(II) Formalized Theory Proving (MiniF2F)

Informal Question:
Expand the following expression: $7(3y + 2)$, show that it is $21y + 14$.

Formal Question:

```
theorem mathd_algebra_182 (y : ℂ)
    : 7 * (3 * y + 2) = 21 * y +
    14 := by
```

Answer:

```
theorem mathd_algebra_182 (y : C)
    : 7 * (3 * y + 2) = 21 * y +
    14 :=
by ring
```

(III) Human-like Derivations (Our STORM-BORN)

Question. Based on Formula (3):

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta D_{\text{KL}}[\pi_\theta(y|x)\|\pi_{\text{ref}}(y|x)],$$

which enforces a KL-constrained reward maximization, how can we derive Formula (4):

$$\pi_r(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\Big(\tfrac{1}{\beta}r(x,y)\Big),$$

where $Z(x)$ is the partition function ensuring π_r is a valid probability distribution?

Answer. We optimize the following objective:

$$\max_\pi \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\big[r(x,y)\big] - \beta\mathbb{D}_{\text{KL}}\big[\pi(y|x)\|\pi_{\text{ref}}(y|x)\big], \tag{1}$$

under any reward function $r(x,y)$, reference model π_{ref}. We now have:

$$\begin{aligned}
&\max_\pi \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\big[r(x,y)\big] - \beta\mathbb{D}_{\text{KL}}\big[\pi(y|x)\|\pi_{\text{ref}}(y|x)\big] \\
&= \max_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\Big[r(x,y) - \beta\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\Big] \\
&= \max_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\Big[\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta}r(x,y)\Big] \\
&= \max_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\big(\frac{1}{\beta}r(x,y)\big)} - \log Z(x)\right],
\end{aligned} \tag{2}$$

where we have partition function:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right). \tag{3}$$

Note that the partition function is a function of only x and the reference policy π_{ref}, but does not depend on the policy π. We can now define

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right), \tag{4}$$

which is a valid probability distribution as $\pi^*(y|x) \geq 0$ for all y and $\sum_y \pi^*(y|x) = 1$. Since $Z(x)$ is not a function of y, we can then re-organize the final objective in Eq. (2) as:

$$\begin{aligned}
&\min_\pi \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] \\
&= \min_\pi \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{D}_{\text{KL}}(\pi(y|x)\|\pi^*(y|x)) - \log Z(x)\right].
\end{aligned} \tag{5}$$

Now, since $Z(x)$ does not depend on π, the minimum is achieved by the policy that minimizes the first KL term. Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical. Hence we have the optimal solution:

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{6}$$

for all $x \in \mathcal{D}$. This completes the derivation.

Figure 1: (I) The Numerical Reasoning dataset (*e.g.* PRM-800K) requires generating numerical values, which may be too simplistic for state-of-the-art LLMs. (II) The Formalized Theory Proving dataset (*e.g.* MiniF2F) encodes problems in formal languages like Lean, hindering intuitive reasoning and real-world generalization. (III) In contrast, Our STORM-BORN dataset emphasizes human-like reasoning, particularly in the purple-colored segment, requiring deep understanding, creativity, and complex reasoning. This task is more challenging than (I) and offers better interpretability and generalizability than (II).

carefully select influential publications within the last two years via the arXiv repository, which also avoids data contamination and remains scalable. To capture human-like reasoning **(C2)**, we utilize heuristic paper filters identify key mathematical reasoning markers (*e.g.* "assume", "define", "proof"). Instead of isolates individual steps like previous works, we extract extracting full derivations to preserve logical reasoning flow. Our multi-agent LLM framework generates problems that require deep theoretical insight, with each derivation containing at least three reasoning steps. Furthermore, to ensure reliable annotations **(C3)**, authoritative sources and expert evaluations are employed.

In this process, we developed a novel multi-agent data curation framework, *STORM* (Synergistic Theorem and fORmula Mining), which integrates human-in-the-loop processes. This framework ensures that the generated data inherently requires complex reasoning and creativity. The development of *STORM*, including the employment of expert mathematicians for 2 months, incurred a total cost of 8000 USD. Even the most advanced LLM, GPT-o1-Pro, is able to solve fewer than 5% of the problems in our STORM-BORN dataset. In contrast, it achieves nearly 95% accuracy on GSM8K, which highlights the inherent complexity and challenge of our STORM-BORN dataset. Additionally, we evaluated the generalization ability of our dataset on numerical reasoning tasks. Remarkably, even models with relatively lower capacities, such as TinyLLaMA-1.1B, demonstrated significant improvements, even though our dataset derivation format differs from numerical reasoning. Our key contributions can be summarized as follows:

- We introduce **STORM-BORN**, a more challenging mathematical derivation dataset curated from recent high-impact papers, featuring complex problems that require theoretical understanding and creative insights.
- We develop a data generation framework, *STORM*, that integrates human-in-the-loop and multi-agent processes to extract complete derivation processes, ensuring both human-like reasoning patterns and reliable annotations in the final high-quality samples.
- Extensive human evaluation demonstrates the challenge of STORM-BORN, even most advanced LLMs solve fewer than 5% of the problems. Our dataset demonstrates generalization capabilities, particularly in the context of numerical reasoning tasks.

2 Related Work

2.1 Large Language Models for Mathematical Reasoning

Mathematical reasoning has become a critical benchmark for evaluating and improving the capabilities of large language models (LLMs). Advances in this field have been driven by multiple factors, including architectural improvements (McLeish et al., 2024), enhanced pretraining datasets (Shao et al., 2024; Allal et al., 2025; Wang et al., 2024b), supervised fine-tuning (Yu et al., 2024b; Cobbe et al., 2021a), reinforcement learning (Wang et al., 2024a; Zelikman et al., 2022), and prompt-based methods such as chain-of-thought reasoning (Ye et al., 2024; Zhang et al., 2022). Frieder et al. (2024) explored LLMs for assisting mathematicians, advocating a hybrid human-model approach. Chang et al. (2023) evaluated LLMs in mathematical reasoning, noting strengths and limitations. Testolin (2024) and Lu et al. (2023) analyzed deep learning in math problem-solving, highlighting challenges in generalization.

Despite advancements, LLMs in mathematical reasoning remain limited by reliance on dataset-driven learning, leading to brittleness and poor generalization (Ahn et al., 2024). To address this, reinforcement learning has been employed to enhance verification mechanisms (Wang et al., 2024a), while prompt engineering, such as physics-inspired prompting (Ye et al., 2024) and automated chain-of-thought generation (Zhang et al., 2022), has improved reasoning consistency. These findings highlight the need for structured reasoning techniques alongside architectural and data improvements to further advance mathematical capabilities in LLMs.

2.2 Mathematical Datasets

Mathematical datasets for LLMs can be broadly categorized into numerical reasoning and automated theorem proving (ATP). *For numerical reasoning*, PRM800K (Lightman et al., 2024), GSM8K (Cobbe et al., 2021b), and GSM_PLUS (Li et al., 2024a) focus on arithmetic problem-solving, requiring step-by-step derivations. FormulaReasoning (Li et al., 2024b) assesses formula-based numerical reasoning, while GAOKAO (Zhang et al., 2024b) benchmarks LLMs' ability to solve complex mathematical problems in Chinese university entrance exams. *For automated theorem proving*, MiniF2F (Zheng et al., 2022) compiles problems from formal proof assistants, including Metamath (Yu et al., 2024a), Isabelle (Frieder et al., 2024), and Lean (Han et al., 2022). ProofNet (Azerbayev et al., 2023) spans undergraduate-level mathematics, bridging LLMs with formal proof verification. Additionally, DRAW-1K (Upadhyay and Chang, 2017) aids in equation derivation, while Ying et al. (2024); Wu et al. (2024) introduced datasets for Lean, supporting machine-verifiable proof generation.

In contrast, our STORM-BORN dataset focuses on challenging mathematical derivations in natural language, demanding complex reasoning and creativity, and is more likely to contain dense, human-like thinking patterns, such as approximations and heuristic cues.

3 Overall Pipeline

In order to enhance LLMs' reasoning abilities for mathematical expressions found in research papers, we created **STORM-BORN**, a dataset that involves advanced mathematical reasoning. This section describes in detail the construction process of **STORM-BORN**.

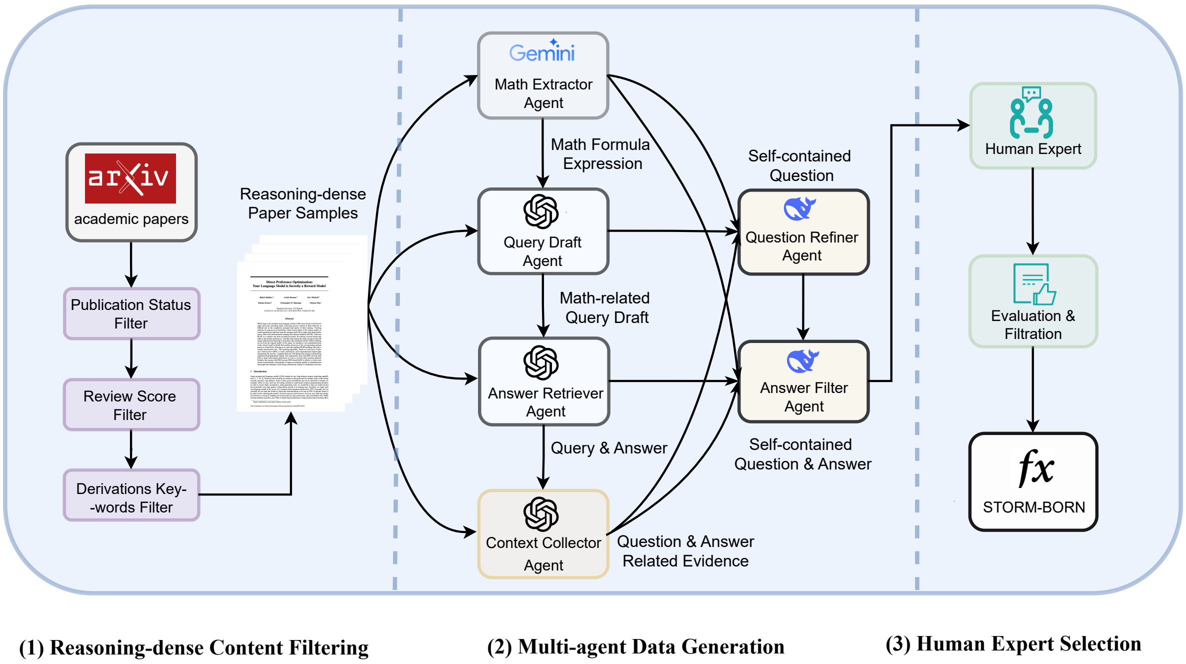

Figure 2: Overview of the data generation framework of STORM-BORN, which consists of three main components: **(1) Reasoning-dense Content Filtering** selects reasoning-dense arXiv papers through linguistic markers and complexity criteria to ensure high-quality mathematical derivations. **(2) Multi-agent Data Generation** orchestrates specialized agents for LaTeX extraction, query formulation, answer retrieval, and context enrichment, culminating in refined mathematical problems. **(3) Human Expert Selection** applies rigorous evaluation criteria to select the most challenging and well-structured problems, resulting in the final STORM-BORN dataset for advancing mathematical reasoning capabilities.

3.1 Reasoning-dense Content Filtering

Distinguishing between basic concept explanations and genuinely complex reasoning requires human-like cognitive processes. To ensure our dataset contains more data and of higher quality, a key aspect lies in the selection of data sources—academic papers. Different papers vary in the amount and quality of data they provide, with some containing extensive mathematical content and detailed proofs and derivation processes, while others do not. Therefore, the focus should be on papers that not only contain a sufficient number of formulas but also provide thorough theorem proofs and derivation processes. More specifically, we select papers based on the following principles.

Publication Status and Review Score. To ensure data reliability, we prioritize papers from reputable journals and conferences, which are peer-reviewed and meet stringent acceptance criteria. We also limit the selection to papers published from May 2023 to October 2024 to ensure content freshness and reduce the risk of using outdated material. Additionally, all selected papers must receive a score higher than "weak accept" from reviewers on the OpenReview platform, ensuring high data quality.

Richness of Mathematical Derivations. We use linguistic markers such as “assume”, “derive”, and “proof” to filter papers that contain detailed derivations and complete sequences of proofs (especially in the appendices). If the target keywords appear more than five times in a paper, we consider it to have a higher likelihood of being our target paper. This ensures that the filtered papers contain high-quality mathematical reasoning.

3.2 Multi-agent Data Generation

We present a six-agent methodology to generate data. This streamlined workflow (see Fig. 2) ensures that each mathematical expression is accompanied by a coherent proof or derivation, a self-contained question and human-like step-by-step answer. Subsequently, we will introduce the entire process. To achieve this goal, we repeatedly refined the workflow, distributed tasks across multiple agents, and continuously modified and validated the prompts. This process was tedious and time-consuming, consuming a lot of effort. We spent

200 USD for GPT-o1-Pro and spent about three weeks on prompts engineering. Appendix A contains further details. This multi-agent framework aims to generate high-quality mathmatical data by systematically extract expressions, pose meaningful questions, retrieve and refine answers, gather requisite background information, and present the self-contained results, ultimately providing more transparent insight into mathematical derivations and proofs. In each step, all mathematical symbols and expressions are converted to latex format.

Why not single-agent? We initially experimented with a single-agent approach for data generation, but the results were poor. The task is inherently complex and involves multiple steps. Using a single LLM leads to excessively long prompts with numerous critical points, making it difficult for the model to follow the instructions effectively. By employing a multi-agent system, we can decompose the task into smaller, more manageable components, allowing each LLM agent to focus on a specific step or key point, which improves the results. Additionally, this modular approach provides greater flexibility, making it easier to modify, refine, or integrate new modules for further improvements. In practice, the multi-agent system significantly enhances both the efficiency and quality of data generation.

Math Expression Extractor Agent We utilize lightweight multi-modal LLMs with extensive prompts for accurate LaTeX formula extraction, avoiding the limitations of traditional OCR techniques (He et al., 2024). It uses a multi-modal large language model (MLLM) that can recognize mathematical expressions in text. After collecting these expressions, the original paper and the extracted expressions are forwarded to the Query Draft Agent.

Query Draft Agent We employ the GPT-o1-Pro LLM as our Query Draft Agent, leveraging a well-structured and effective long prompt **exceeding 1k tokens**. It receives the entire paper rather than the chunked paper, which ensure it could comprehensively understand the entire paper. For each expression extracted from the Math Expression Extractor Agent, it generates at least one *query*, focusing on the theorem or formula derivation problems. We also added few-shots to enhance the output format stability. The details of its prompt is in Appendix A.2.

Answer Retriever Agent The Answer Retriever takes the entire paper, a given expression, and its corresponding query as input. The Answer Retriever Agent searches the paper for relevant content that can answer the query. Once relevant content is found, it extracts the entire answer directly from the paper rather than make a proof itself to avoid hallucination. Similar to Query Draft Agent, practice has proved that the task of this agent is also difficult and requires a more powerful LLM (*e.g.* GPT-o1-Pro). The effective prompt we finally get is also relatively long with *nearly 500 tokens*. The details of this prompt is in Appendix A.3.

Context Collector Agent Although Query Draft Agent and Answer Retriever Agent could generate high-quality *query* and *answer*, there still remains the possibility that they lack full information to make them self-contained, which means we could give the proof based on the query and check whether the proof is same as the *answer* retrieved from the original resource. The Context Collector captures these information and stores them as *evidence* for the target self-contained question and answer.

Question Refiner Agent The goal of this agent is to incorporate the information from the *evidence* into the *query* and *answer*, thereby generating *self-contained question* that can be answered independently without reading original resource.

Answer Filter Agent Since our goal is to focus on mathematical reasoning, the Answer Filter Agent filters out any irrelevant content after receiving the data processed by Question Refiner Agent, retaining only the essential information needed to understand how the expression is derived or proven. By filtering out unnecessary data, the subsequent modules can significantly reduce redundant workload and generates the self-contained question and answer.

3.3 Human Expert Selection

Through Multi-agent Data Generation, we obtained 2k samples. We could directly train on our 2k samples, however, our goal is to extract the most challenging and high-quality dataset. To achieve this, we employ an expert mathematicians group to conduct a rigorous selection process, ultimately arriving at a refined set of 100 samples. We sent the self-contained question and answer generated in (Sec. 3.2) to human experts who are familiar with the reasoning-dense paper samples for selection. Human experts conducted strict audits on data quality, retained data that meets the standards, eliminated data that has no research value, and manually modified and optimized data that is not of borderline quality but can be improved. Each paper was processed by experts for about 30 samples of question and answer, and the processing of a single paper took about 15 minutes. Through iterative expert feedback and revision, we refined the dataset, ensuring that each sample meets the high-quality standards set by our guiding principles. This expert-driven process was critical to ensuring that the dataset reflects complex human-like mathematical reasoning, resulting in the final **STORM-BORN** dataset. This process was guided by the following five core principles: Reasoning Density, Problem Clarity, Derivation Correctness, Reasoning Density, and Evidence Quality.

(Q1) Reasoning Type: *Does the problem demand creative insight and complex reasoning?* Initially, mathematicians determine whether the problem involves genuinely complex reasoning like deriving or proving a formula, as opposed to simple explanation or definition.

(Q2) Problem Clarity: *Is the problem clear, well-defined, and solvable with the existing information?* This step evaluates the explicitness of the problem's goal and conditions. Ambiguities or incomplete queries, where critical context is missing, are flagged for refinement. Human expert intervention is crucial here, as mathematical clarity often requires subjective interpretation, especially when key information is implied or subtly conveyed.

(Q3) Derivation Correctness: *Are all derivation steps logically valid, error-free, and complete?* Mathematicians carefully review each derivation step for correctness, ensuring that all logical transitions are accurate and coherent. This stage presents a significant challenge, as identifying logical errors or omissions often requires a deep theoretical understanding and specialized expertise.

(Q4) Reasoning Density: *Does the reasoning process include sufficient logical steps, exhibit heuristic reasoning cues, and demonstrate trial-and-error similar to human problem-solving?* This requires human expertise to assess whether the reasoning is sufficiently dense, complete, and heuristic. Mathematicians identify patterns in the reasoning that reflect human-like trial-and-error approaches. Missing or incomplete justifications are flagged for further revision.

(Q5) Evidence Quality: *Are external references, if provided, accurate and relevant?* The final challenge assesses whether the references used to substantiate derivations are both accurate and relevant. Human expertise is essential for ensuring the appropriateness and correctness of these references, as the task often involves subjective interpretation of their relevance to the derivation.

4 Experiments

4.1 Case Study

In this preliminary case study, we compared three different types of datasets (see Fig. 1): (I) Numerical reasoning datasets such as PRM-800K, which mainly examine numerical calculations, but may be too simple for advanced language models. For example, it can be solved like the expected value of a coin toss, which first calculates the probability of heads and tails, then calculate the payoff. (II) Formal proof datasets such as Minif2F, which use formal languages such as Lean to describe problems. Although rigorous, they are not easy to understand intuitively and are not easy to associate with real-world scenarios. Moreover, the answer examples can be solved with only one ring. (III) Our proposed STORM-BORN dataset focuses more on human-like reasoning processes and requires deeper understanding, flexible thinking, and complex reasoning. It is not only more challenging than (I), but also more interpretable and general than (II). In our example, the system in the DPO (Rafailov et al., 2023) paper, the system captures the derivation of important formulas and extracts the complete details of the derivation from the appendix of the paper, demonstrating the effectiveness of our method in scenarios of complex research.

4.2 Human Evaluation

Since our data mainly contains difficult mathematical proofs and derivation processes, rather than numerical data, it is difficult to directly evaluate the correctness. Existing similarity evaluation methods and LLMs also have difficulty in ensuring the accuracy of the evaluation (Fig. 3). So for experimental results on our dataset, we rely on human evaluation (following Q3, Q4 in Sec. 3.3).

Based on the above human evaluation criteria. We systematically evaluated six leading language

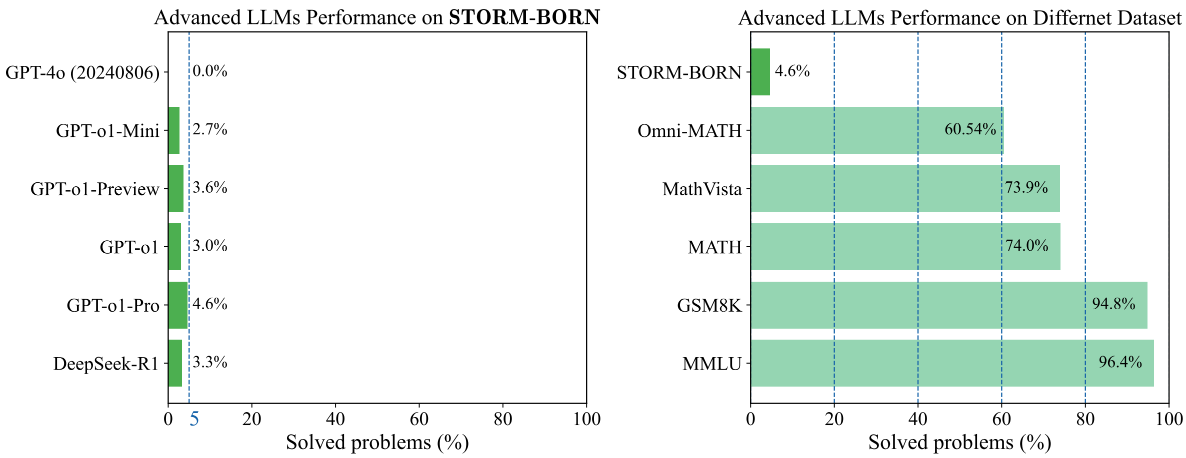

Figure 3: Performance of leading language models on STORM-BORN based on a human expert evaluation. All models show consistently poor performance, with even the best models solving less than 5% of problems. When re-evaluating problems that were solved at least once by any model, GPT-o1-Pro demonstrated the strongest performance across repeated trials.

Model	GSM8K	GSM_PLUS	MATH
Tiny-Llama-1.1B-chat	1.36%	1.18%	1.20%
Tiny-Llama-1.1B-chat (Ours)	**2.05%** (↑ 0.69)	**1.29%** (↑ 0.11)	**4.00%** (↑ 2.80)
Tiny-Llama-1.1B-chat (GSM8K)	8.79%	4.52%	2.80%
Tiny-Llama-1.1B-chat (GSM8K + Ours)	**9.55%** (↑ 0.76)	**4.79%** (↑ 0.27)	**4.80%** (↑ 2.00)
Tiny-Llama-1.1B-chat (MiniF2F)	**1.67%**	**1.42%**	3.20%
Tiny-Llama-1.1B-chat (MiniF2F + Ours)	1.59% (↓ 0.08)	1.41% (↓ 0.01)	**3.80%** (↑ 0.60)
Llama2-7B-hf	7.96%	2.80%	1.60%
Llama2-7B-hf (Ours)	**8.80%** (↑ 0.84)	**4.85%** (↑ 2.05)	**2.60%** (↑ 1.00)

Table 1: Experimental Results of 1.1B and 7B LLMs on GSM8K, GSM8K_PLUS, and MATH. $(\cdot)$ means finetuned dataset (*e.g.* GSM8k, MiniF2F, Our STORM-BORN), "+" denotes data combination. The best results are highlighted in **bold**.

models on our dataset - GPT-o1-Pro, GPT-o1, GPT-o1-Preview, GPT-4o, and DeepSeek-R1. Experimental results show that GPT-o1-Pro has an accuracy rate of 5% on the test data, which is the best performance among all the tested models (see Fig. 3). Compared to other datasets (*e.g.* MMLU, Omni-MATH), which are almost solved, obviously, even the most advanced models still have limited performance on our dataset, which further highlights the challenge of this dataset and the complexity of mathematical reasoning tasks.

4.3 Data Quality By Downstream Application

To evaluate the impact of STROM-BORN on enhancing mathematical reasoning abilities, we perform full fine-tuning on Tiny-Llama-1.1B-chat (Zhang et al., 2024a) and Llama-2-7B (Touvron et al., 2023) and evaluate them on the GSM8K (Cobbe et al., 2021b), GSM-Plus (Li et al., 2024a), and MATH (Hendrycks et al., 2021) datasets.

Experimental results indicate that STROM-BORN improves model performance across multiple mathematical reasoning benchmarks. To quantify its impact, we first fine-tune models using only the 73 training samples from STROM-BORN and evaluate them on benchmarks. The results show that fine-tuning solely on STROM-BORN leads to an accuracy improvement of 2.80 percentage points on MATH for Tiny-Llama-1.1B-chat and 1.00 percentage point for Llama-2-7B. These findings suggest that STROM-BORN enhances multi-step logical reasoning capabilities, particularly in complex problem-solving scenarios.

To ensure alignment between the training and testing data distributions, we randomly insert 73 STROM-BORN training samples into the GSM8K training split and evaluate the models on GSM8K, GSM-Plus and MATH. The results indicate that this strategy yields a 0.76 percentage point improvement on GSM8K and a 2.00 percentage point improvement on MATH, further demonstrating that STROM-BORN contributes positively when integrated into larger training corpora.

Furthermore, the learnability of STROM-BORN is compared with that of the MiniF2F (Zheng et al., 2022) dataset, which primarily consists of formal mathematical proofs. Fine-tuning Tiny-Llama-1.1B-chat exclusively on MiniF2F results in lower accuracy on GSM8K and MATH compared to fine-tuning on STROM-BORN, with performance differences of 0.38 and 0.80 percentage points, respectively. Even when jointly trained with MiniF2F, the model's accuracy on MATH remains largely dependent on the contribution of STROM-BORN, yielding a 0.20 percentage point improvement. These results suggest that STROM-BORN is more learnable and better aligns with the reasoning patterns of language models, making it a more effective fine-tuning dataset for mathematical problem-solving tasks.

5 Conclusion

In conclusion, we present **STORM-BORN**, a novel dataset designed to address the limitations of existing mathematical derivation datasets. Curated from recent top-tier academic papers via the arXiv repository, STORM-BORN is both nuanced and scalable, while avoiding data contamination. Unlike isolated steps, we capture full derivations to preserve logical flow and encourage deep theoretical reasoning. Using a human-in-loop and multi-agent LLM framework *STORM*, we generate problems requiring at least three reasoning steps, ensuring complexity and creativity. Expert evaluations ensure reliable annotations. Empirical results highlight the dataset's challenge, with advanced LLMs like GPT-o1-Pro solving fewer than 5% of the problems, compared to 95% accuracy on GSM8K. Additionally, STORM-BORN demonstrates strong generalization capabilities, offering a high-difficulty evaluation benchmark for AI's approach to mathematician-level reasoning.

Limitations

This study addresses an important gap in the field, but it also faces certain limitations. Specifically, the automated evaluation of data quality remains challenging, as our focus on complex mathematical derivations rather than numerical computing makes quality assessment difficult (a problem also noted by Glazer et al. (2024)). Currently, we rely primarily on a carefully designed multi-agent curation pipeline and manual inspection by mathematicians. However, with the rapid advancement and scaling of LLMs, we believe that in the future, LLMs can be fully employed to automate this process, iteratively improving and optimizing it.

Ethics Statement

The dataset construction process in this study strictly adheres to ethical guidelines and fully complies with relevant legal regulations. We obtain publicly accessible, high-quality academic papers from ArXiv and utilize a combination of multi-modal models and human evaluation feedback for data processing and optimization, ensuring data quality and reliability before generating the final dataset. The entire data collection and processing workflow is transparent and traceable, with all papers sourced from legal and publicly available channels, guaranteeing compliance and traceability of data. The dataset constructed in this study is intended solely for academic research and experimental purposes, with no involvement in commercial applications or risk of sensitive information leakage.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *Preprint*, arXiv:2307.03109.

Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V. Le, and Thang Luong. 2025. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Preprint*, arXiv:2502.03544.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *CADE*.

Simon Frieder, Julius Berner, Philipp Petersen, and Thomas Lukasiewicz. 2024. Large language models for mathematicians. *Preprint*, arXiv:2312.04556.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *Preprint*, arXiv:2411.04872.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward Ayers, and Stanislas Polu. 2022. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Xiao Li, Bolin Zhu, Sichen Liu, Yin Zhu, Yiwei Liu, and Gong Cheng. 2024b. Formulareasoning: A dataset for formula-based numerical reasoning. *Preprint*, arXiv:2402.12692.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. *Preprint*, arXiv:2212.10535.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. 2024. Transformers can do arithmetic with the right embeddings. *Preprint*, arXiv:2405.17399.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Alberto Testolin. 2024. Can neural networks do arithmetic? a survey on the elementary numerical skills of state-of-the-art deep learning models. *Applied Sciences*, 14(2):744.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 494–504, Valencia, Spain. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Zengzhi Wang, Xuefeng Li, Rui Xia, and Pengfei Liu. 2024b. Mathpile: A billion-token-scale pretraining corpus for math. *Preprint*, arXiv:2312.17120.

Zijian Wu, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean-github: Compiling github lean repositories for a versatile lean prover. *Preprint*, arXiv:2407.17227.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *Preprint*, arXiv:2407.20311.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *Preprint*, arXiv:2406.03847.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024a. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024b. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024a. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2024b. Evaluating the performance of large language models on gaokao benchmark. *Preprint*, arXiv:2305.12474.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

A Workload and Prompts

We invested a lot of work, energy, and time in this research. Our goal is to generate high-quality formula derivation and question-answering. At first glance, this seems to be a simple task, but in fact it involves extremely complex and extensive workload. Initially, we explored various technical solutions, such as optical character recognition (OCR), but when using OCR for formula recognition and extraction, we often encountered incomplete positioning (only part of the formula was framed out), resulting in inaccurate formula extraction. After repeated comparisons and experiments, we finally chose the method of multi-agent large language model (LLM) collaboration, which has consumed some time and energy.

The biggest challenge appeared in the prompt design and optimization stage. Practice has shown that LLM will encounter a series of problems, such as identifying key data in long texts, following instructions, and producing stable output. To solve these difficulties, we continuously refined the overall workflow and assigned complex tasks to multiple appropriate numbers of agents (see Fig. 2) for collaborative execution. At the same time, the prompts of each agent were modified, iterated, and verified for multiple rounds. This process is tedious and time-consuming, and consumes a lot of energy.

Regarding manual evaluation and feedback, each paper required individuals with relevant academic background to read, assess, and provide feedback on the generated data, which increases labor and time costs.

For resource costs and time costs, please see Appendix B.

Thanks to this painstaking and systematic workflow, we were finally able to obtain high-quality question-answering data. We will introduce our prompts below, hoping to provide further insight into the complexity of this study, the extensive workload involved, and our efforts to overcome a variety of challenges.

A.1 Math Expression Extractor Agent

We encountered many problems in the process, such as: the set of extracted mathematical expressions omitted important items, contained unnecessary items and repeated items; the output latex format did not meet the requirements. To solve these problems, we added new rules to the prompt and repeatedly verified the effect in practice, **and iterated continuously**. Through repeated iterations in practice, these problems were solved, which enables the MLLM to follow the instructions to extract all important mathematical expressions (formulas, theorems, lemmas, etc.), ignore unimportant mathematical expressions (such as intermediate expressions that appear in the derivation process, mathematical content inserted in the paragraph), and ensure that the output expression is in the correct format.

Prompt of Math Expression Extractor

```
"""Read the paper, then:

1. Formula Recognition:
- Identify all mathematical formulas, theorems, lemmas, and corollaries in
    the paper. Especially Numbered formulas.Retain the formula's number (if
    any).
- For formulas without explicit labels (i.e., those not labeled as "theorem,
    " "lemma, " or "corollary"), classify them as "formula."
- Required types of formulas to recognize:
    - Numbered formulas.
    - Formulas that appear on separate lines (for example, occupying a line
    or multiple lines by themselves in the paper).
- Ignore:
    - Formulas that appear in the middle of a paragraph without separate
    lines or numbers.
- Make sure there are no duplicates in the results (duplicates refer to
    formulas that are exactly the same after conversion to LaTeX. If the same
    formula appears in the paper under different numbers, treat them as the
    same formula).
```

```
2. LaTeX Conversion (Convert the formulas identified in step 1 into LaTeX
    format strings):
- Symbols: Convert mathematical symbols accurately.
- Subscripts and superscripts: Convert subscripts and superscripts correctly.
- Uppercase and lowercase: Preserve the original variable and constant casing.
- Formula structure: Keep the entire structure of the formula intact.
- Formula numbering: Retain the formula's number (if any).
- Italics: For italicized variables in the text, wrap them with \textit{} in
    LaTeX.
- Math environment: Use `$ ... $` for inline formulas and `$$ ... $$` for block
    (display) formulas.
- Additional conditions: Check whether the paper includes definitions or
    explanations immediately following the formula (for example, "where X is
    ...") and incorporate them if present.

3. JSONL Output:
- Output all converted LaTeX strings in multi-line JSONL format so they can
    be parsed line by line.
- Each line should be a JSON object whose key is the type of the formula
    ("formula", "lemma", "theorem", "corollary", etc.) and whose value is the
    LaTeX string obtained from step 2.
- Be sure to follow the requirements in step 2!

Ensure the formulas are exactly the same as in the original text!"""
```

A.2 Query Draft Agent

The more difficult task also leads to more problems encountered in the process, such as the generated questions are too rigid, the questions lack prerequisites, and only the formula reference number is output without the original formula which emphasizes the need of Context Collector Agent and Question Refiner Agent.

Prompt of Query Draft

```
"""I will provide you with a dataset extracted from this paper, in JSONL
    format. Each entry is a dictionary whose keys are "formula, " "lemma, "
    "theorem, " etc., representing the category of the mathematical
    expression, and whose values contain a mathematical expression in LaTeX
    format, extracted from the paper.

Carefully read and understand the paper's content, especially the parts
    related to each formula in the JSONL. For each formula, please complete
    the following steps:

---

Step 1:
Locate where the formula is first defined or fully derived in the paper, and
    use the relevant context to extract all the direct necessary conditions
    for deriving or proving that formula. These preconditions include, but
    are not limited to:

1. Which other formulas this formula is derived from or depends on. For each
    such formula, record its full content (in LaTeX format), its numbering
    (if any), and its name (if any).
2. Relevant problem settings.
3. The specific meaning of symbols or variables involved in the formula.

---

Step 2:
Based on the extracted preconditions, generate a complete question that
    clearly asks how to derive or prove the formula. The question should
    include:
```

```
1. The formula itself: Present the full content of this formula (in LaTeX
    format). Do not only reference its number.
2. The preconditions: Explicitly integrate the preconditions extracted from
    the paper into the question. List out the full contents of all the
    formulas it depends on and reference them by their respective numbers or
    names. Do not produce a question such as "What are the preconditions?"

The form of the question must meet the following requirements:

- If a formula is derived from one or more other formulas, you must
    explicitly list the full content (in LaTeX) of these preceding formulas
    and reference them by their numbers or names, and explain how the current
    formula is derived from them. For example, if the paper contains Formula
    3 (content: X) and Formula 4 (content: Y), and Formula 4 is derived from
    Formula 3, then the generated question should be:

"Based on Formula 3: X, how can we derive Formula 4: Y?"

- If the formula is a theorem, lemma, or corollary, please generate a
    question asking how to prove it, for example:

"How can we prove Lemma 1: X is true?"

Note: The question must be structured and logical, clearly showing the
    derivation or proof process of the formula and explicitly reflecting the
    dependency between formulas while fully presenting all related formulas.

---

Step 3:
Match each formula with its corresponding question and output the result in
    multi-line JSONL format.

Each data entry should be a dictionary containing the following two key-value
    pairs:

1. Formula type:
- The key is "formula, " "lemma, " "theorem, " etc.
- The value is the LaTeX content of the formula.
2. Generated question:
- The key is "query."
- The value is the complete question generated according to Step 1 and Step 2.

---

Important Notes:
1. Format Requirements:
- Ensure the output is in JSONL format, with each line corresponding to one
    data entry.
2. Formula Accuracy:
- If the question contains mathematical expressions, convert them into LaTeX
    format. Make sure they align with the original mathematical meaning.
    Minor formatting differences can be ignored.
3. LaTeX Conversion (Converts the mathematical expressions contained in the
    problem to strings in LaTeX format):
- Symbols: Convert mathematical symbols accurately.
- Subscripts and superscripts: Convert subscripts and superscripts correctly.
- Uppercase and lowercase: Preserve the original variable and constant casing.
- Formula structure: Keep the entire structure of the formula intact.
- Formula numbering: Retain the formula's number (if any).
- Italics: For italicized variables in the text, wrap them with \textit{} in
    LaTeX.
- Math environment: Use `$ ... $` for inline formulas and `$$ ... $$` for block
    (display) formulas.
4. Completeness of Preconditions:
- The question content must include all direct necessary conditions.
    Particularly, indicate which other formulas the current formula is
    derived from or depends on, and clearly specify the entire content,
```

```
    numbering, or name of those referenced formulas. Do not produce questions
    such as "What are the preconditions?"

---

Examples:
Here are some example questions and their corresponding output formats for
    reference:

- Suppose the paper contains the following formula:
{"lemma": "Lemma 1.  The function $f (x)$ is continuous."}
The generated question might be:
{"query":"How can we prove Lemma 1: The function $f (x)$ is continuous. is
    true?"}

- Suppose the paper contains the following formula:
{"formula": "y = mx + b"}
and it is explained that this formula is derived from y = f (x) and f (x) =
    mx + b. Then the generated question might be:
{"query":"Based on the formulas: $y = f (x)$ and $f (x) = mx + b$, how can we
    derive the formula: $y = mx + b$?"}

- Suppose the paper contains the following formula:
{"formula": "$$\\pi_r (y | x) = \\frac{1}{Z (x)} \\pi_{ref}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y))$$"}
and it is explained that this formula is derived from Formula 3, $KL (\\pi_r
    (y|x) || \\pi_{ref}(y|x)) \\leq \\epsilon$. Then the generated question
    should be:
{"query":"Based on Formula 3: $KL (\\pi_r (y|x) || \\pi_{ref}(y|x)) \\leq
    \\epsilon$, how can we derive Formula: $\\pi_r (y | x) = \\frac{1}{Z (x)}
    \\pi_{ref}(y | x) \\exp (\\frac{1}{\\beta} r (x, y))$?"}

The dataset is as follows:\n
```

A.3 Answer Retriever Agent

In order to solve the problems encountered in the process, such as: **the answer is not extracted from the original text but the large model generates the answer itself**, the answer retrieved in this agent may lack the important complete proof process in the appendix, or is a summary of the answer in the original text, the effective prompt we finally get is also relatively long with nearly 500 tokens.

Prompt of Answer Retriever

```
"""I will provide a JSONL-format dataset extracted from this paper. Each
    piece of data in the dataset is a dictionary containing two main
    key-value pairs:
1. **Formula-related keys ("formula", "lemma", "theorem", etc.)** indicating
    the type of mathematical expression; the value is the LaTeX-formatted
    mathematical expression extracted from the paper.
2. **query**, whose value is a question generated by a large model based on
    the paper and the mathematical expression.

Please process this dataset according to the following steps and requirements.

---

### Step One:
For the "expression" and "query" in each piece of data, determine whether the
    answer to that question can be found in the paper. The specific steps are
    as follows:

1. **Find the first occurrence**
   - Locate where the expression first appears in the paper and check the
    surrounding context for relevant clues.
   - If there are any references or citations, follow those as well.
```

```
2. **Check the appendix and other sections**
   - Search the paper's appendix or other relevant chapters to see if the
    proof or derivation steps for that expression are provided. This may well
    be the answer to the question.

3. **Confirm feasibility**
   - If the paper does not include any relevant content addressing the
    question, you may skip this expression and proceed to the next one.
   - If the paper does indeed contain content that can answer the question,
    extract the relevant content from the original text.

When extracting the answer, please note the following requirements:
- **Completeness**: The extracted answers should cover all the relevant steps
    needed to solve the problem in the paper.
- **Consistency**: Include only content from the original text in the answer
    (you may make minimal necessary edits for coherence, but do not change
    the original meaning). Avoid adding extra content or descriptions not
    found in the original text.
- **Citation handling**: If the answer cites other formulas or theorems from
    the paper, also include their original content in the derivation or proof
    process, rather than leaving only references or labels.
- **LaTeX conversion**: Ensure all mathematical expressions are converted to
    the same LaTeX format as in the original text, including:
  - Accuracy of symbols, subscripts, superscripts, and capitalization.
  - Preserving the original structure and numbering (if any).
  - Using \textit{} for italicized variables.
  - Using $...$ for inline math expressions and $$...$$ for display math
    expressions.

---

### Step Two:
Match the answers extracted in Step One with the corresponding entries in the
    dataset, and add a new key-value pair to form a new data record. The
    specific requirements are:

- For each original data entry, add a new key called `whole_label`, whose
    value is the LaTeX-formatted answer content extracted from the paper.
- Output format must be **multi-line JSONL**, one piece of data per line:
  1. The original two key-value pairs remain unchanged and must not be
    modified.
  2. Add the `whole_label` key as the third key-value pair.

---

### Output Requirements:
1. **Multi-line JSONL format**: One data entry per line.
2. **Accuracy of content**: Formulas must match the original text of the
    paper exactly, with correct symbols, subscripts, superscripts, and
    capitalization.
3. ** Content consistency ** : Only retain the original content in the answer
    (you can make a small amount of necessary cohesive editing, but do not
    change the original meaning), and try to avoid adding additional content
    or descriptions that do not appear in the original.
---

### Note:
- Please strictly follow the above requirements to avoid omitting any key
    content.
- Ensure there are no errors or incomplete parts in the output text.

---

Below is the dataset:
"""
```

B Resource and Time Costs

At the outset, it is important to highlight the considerable workload entailed in our approach, with the associated resource and time costs reflecting the extensive efforts required for its implementation.

B.1 Resource Costs

To support the multi-agent system, we subscribed to GPT-o1-Pro for a one-month period at an approximate cost of 200 USD. For simpler tasks, such as Math Expression Extraction and Answer Filtering, we utilize free LLMs as agents. For more complex tasks, such as Query Generation and Answer Retrieval, we rely on the paid GPT-o1-Pro model to ensure enhanced performance and accuracy. An illustrative example is provided below. After the Math Expression Extraction step, using GPT-o1-Pro for both Query Generation and Answer Retrieval yielded the results shown in Appendix B.1.1. In contrast, using the gemini-2.0-flash-exp model for the same tasks resulted in Appendix B.1.2. Furthermore, when using the query generated by GPT-o1-Pro and applying gemini-2.0-flash-exp for Answer Retrieval, the result was Appendix B.1.3. This comparison clearly illustrates the necessity of using more powerful models like GPT-o1-Pro for complex tasks.

B.1.1 GPT-o1-Pro Results

Here we provide the results of GPT-o1-Pro as follows:

GPT-o1-Pro as Query Draft and Answer Retriever.

formula:

```
"$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y)), $$(4)"
```

query:

```
"Based on Formula (3): $$\\max _{\\pi_{\\theta}} \\mathbb{E}_{x \\sim D, y
    \\sim \\pi_{\\theta}(y|x)}[r_{\\phi}(x, y)]-\\beta D_{K
    L}[\\pi_{\\theta}(y | x) || \\pi_{r e f}(y | x)], $$ which enforces a
    KL-constrained reward maximization, how can we derive Formula (4): $$
    \\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp\\Bigl
    (\\tfrac{1}{\\beta} r (x, y)\\Bigr), $$ where $Z (x)$ is the partition
    function ensuring $\\pi_{r}$ is a valid probability distribution?"
```

answer:

```
"In the paper's Appendix A.1 (pp. 15-16), ``Deriving the Optimum of the
    KL-Constrained Reward Maximization Objective, '' the authors
    write:\n\n``[We] optimize the following objective:\n$$\\max_{\\pi}\\,
    \\mathbb{E}_{x\\sim D, y\\sim\\pi (y\\mid x)}\\bigl[r (x, y)\\bigr]\\;
    -\\; \\beta\\, D_{\\mathrm{KL}}\\bigl[\\pi (y\\mid x)\\, \\|\\,
    \\pi_{\\mathrm{ref}}(y\\mid x)\\bigr].$$\n\n... We now
    have:\n$$\\max_{\\pi}\\, \\mathbb{E}_{x\\sim D, y\\sim \\pi (y\\mid
    x)}\\bigl[r (x, y)\\; -\\; \\log\\frac{\\pi (y\\mid
    x)}{\\pi_{\\mathrm{ref}}(y\\mid x)}\\bigr]\\; =\\; \\max_{\\pi}\\,
    \\mathbb{E}_{x\\sim D, y\\sim \\pi (y\\mid x)} \\bigl[\\, r (x, y)\\;
    -\\; \\log\\pi (y\\mid x)\\; +\\; \\log\\pi_{\\mathrm{ref}}(y\\mid
    x)\\bigr].$$\n\nSince $Z (x)$ does not depend on $\\pi$, the minimum is
    achieved by the policy that minimizes the first KL term. Gibbs'
    inequality tells us that the KL-divergence is minimized at 0 if and only
    if the two distributions are identical. Hence we have the optimal
    solution:\n$$\\pi^{*}(y\\mid x)\\; =\\; \\frac{1}{Z (x)}\\,
    \\pi_{\\mathrm{ref}}(y\\mid x)\\; \\exp\\Bigl (\\tfrac{1}{\\beta}\\, r
    (x, y)\\Bigr), $$\nfor all $x\\in\\mathcal{D}$. This completes the
    derivation.''\n\n\\(\\textit{Here, }\\; Z (x)=\\sum_{y}\\,
    \\pi_{\\mathrm{ref}}(y\\mid x)\\exp\\bigl (\\tfrac{1}{\\beta}r (x,
    y)\\bigr)\\; \\textit{ensures}\\; \\pi^{*}(y\\mid x)\\; \\textit{is a
    valid probability distribution.}\\)\n\nevidence: Appendix A.1, pp. 15-16
    of the paper"}
```

B.1.2 Gemini-2.0-Flash-Exp Results

Here we provide the results of Gemini-2.0-Flash-Exp as follows:

Gemini-2.0-Flash-Exp as Query Draft and Answer Retriever, using the same formula as Appendix B.1.1. It is evident that compared to Appendix B.1.1, the query generated by gemini-2.0-flash-exp did not fully display the formula conditions (formula 4 in Fig. 1), and the derivation process was overly general rather than a detailed step-by-step derivation.

formula:

```
"$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y)), $$(4)"
```

query:

```
"Given the RL objective in Formula (3):$$\\max _{\\pi_{\\theta}}
    \\mathbb{E}_{x \\sim D, y \\sim \\pi_{\\theta}(y|x)}[r_{\\phi}(x,
    y)]-\\beta D_{K L}[\\pi_{\\theta}(y | x) || \\pi_{r e f}(y | x)], $$ how
    can we express the optimal solution $\\pi_{r}(y | x)$ to the
    KL-constrained reward maximization objective, where $Z (x)$ is the
    partition function?"
```

answer:

```
"Following prior work [31, 30, 19, 15], it is straightforward to show that
    the optimal solution to the KL-constrained reward maximization objective
    in Eq. 3 takes the form:\n$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e
    f}(y | x) \\exp (\\frac{1}{\\beta} r (x, y)), $$(4)\nwhere $$Z (x) =
    \\sum_{y} \\pi_{r e f}(y | x) \\exp (\\frac{1}{\\beta} r (x, y)), $$ is
    the partition function. See Appendix A.1 for a complete derivation."
```

B.1.3 Gemini-2.0-Flash-Exp Results

Here we provide the results of Gemini-2.0-Flash-Exp as follows:

Gemini-2.0-Flash-Exp as Answer Retriever, using the same formula and query as Appendix B.1.1. It is evident that the result was less comprehensive and detailed than Appendix B.1.1.

formula:

```
"$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y)), $$(4)"
```

query:

```
"Based on Formula (3): $$\\max _{\\pi_{\\theta}} \\mathbb{E}_{x \\sim D, y
    \\sim \\pi_{\\theta}(y|x)}[r_{\\phi}(x, y)]-\\beta D_{K
    L}[\\pi_{\\theta}(y | x) || \\pi_{r e f}(y | x)], $$ which enforces a
    KL-constrained reward maximization, how can we derive Formula (4): $$
    \\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp\\Bigl
    (\\tfrac{1}{\\beta} r (x, y)\\Bigr), $$ where $Z (x)$ is the partition
    function ensuring $\\pi_{r}$ is a valid probability distribution?"
```

answer:

```
"Following prior work [31, 30, 19, 15], it is straightforward to show that
    the optimal solution to the KL-constrained reward maximization objective
    in Eq. 3 takes the form:\n$$\\pi_{r}(y | x)=\\frac{1}{Z (x)}
    \\pi_{\\textit{ref}}(y | x) \\exp (\\frac{1}{\\beta} r (x, y)), $$
    (4)\nwhere $Z (x)=\\sum_{y} \\pi_{\\textit{ref}}(y | x) \\exp
```

```
(\\frac{1}{\\beta} r (x, y))$ is the partition function. evidence:
Section A.1, Appendix"
```

B.2 Time Costs

On average, the processing time for each paper using the multi-agent system is approximately 20 minutes, with variability depending on the paper's length and the number of formulas. When utilizing GPT-o1-Pro for more challenging tasks, the processing time can be significantly longer. Moreover, output failures may occur, requiring multiple retries—sometimes two or even three times—leading to substantial time costs. Additionally, issues such as the "dumbing down" of LLMs during intensive tasks can further hinder experimental progress, creating delays in task completion. This represents a significant source of time cost, as repeated attempts are necessary to recover from failures and ensure valid results. In addition, there is no API for GPT-o1-Pro, so we have to use the web version. And the model can not receive pdf files, so we can only convert the paper into page screenshots and gradually upload, which increases the labor costs and time costs.