KEMRP: Can a Knowledge Graph Enhance the Ability of LLMs to Solve Math Word Problems?

Anonymous ACL submission

Abstract

This paper introduces a novel framework that combines Large Language Models (LLMs) with mathematical knowledge graphs (KGs) to solve Math Word Problems (MWPs). Current methods leveraging LLMs for MWPs primarily rely on fine-tuning or prompt engineering; the former operates as a black box with limited interpretability, while the latter completely depends on the inherent abilities of LLMs. In contrast, our approach enables explicit and interpretable mathematical reasoning by dynamically linking linguistic patterns to structured mathematical knowledge. We present two comprehensive knowledge graphs—MWPEN-KG (English) and MT700-KG (Chinese)—that capture essential mathematical concepts and relationships for problem-solving. The framework leverages LLMs to decompose problems into mathematical concepts while simultaneously accessing relevant entities and paths in KG to guide step-by-step solutions. Extensive experiments across five MWP benchmarks (MAWPS, MathQA, Math23K, Ape210k, CM17K) using four different LLMs (DeepSeek-Chat, GPT-4o, GPT-3.5-Turbo, Qwen-Turbo) reveal the framework's superior performance compared to conventional methods, achieving state-of-the-art results on all five datasets. Our work demonstrates that combining LLMs with KGs has significant potential in solving MWPs. [1].

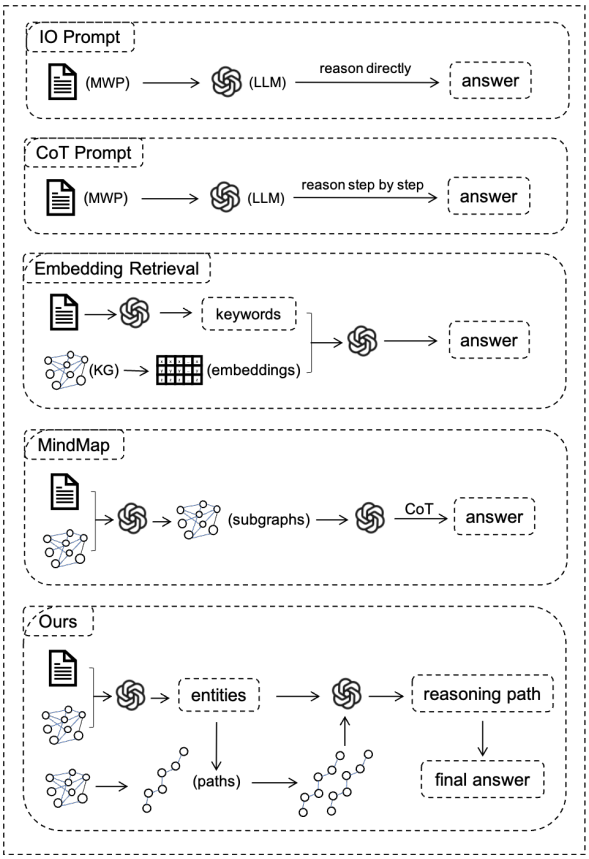

Figure 1: Comparing our method with other baselines. IO Prompt and CoT Prompt rely solely on the intrinsic knowledge of LLMs, while Embedding Retrieval, MindMap, and our method use KG to enhance LLMs.

1 Introduction

Math Word Problems (MWPs) present complex narrative texts that describe real-world scenarios with embedded quantitative relationships, requiring solutions for specific unknown quantities(Ahn et al., 2024). Solving MWPs remains a significant challenge in artificial intelligence research, demanding not only sophisticated linguistic comprehension but also precise semantic alignment between natural language expressions and mathematical principles to uncover the implicit logical frameworks within textual descriptions.

The development of Large Language Models (LLMs) has driven substantial progress in MWP solving(Ahn et al., 2024), primarily through two directions: task-specific fine-tuning and prompt engineering. Task-specific fine-tuning involves adapting pre-trained LLMs to the MWP domain by training them on extensive datasets, improving their performance on benchmark tests. However, de-

[1]We will publicize our code after the paper has been accepted.

spite their good performance, the decision-making processes remain a black box(Hassija et al., 2024). This lack of transparency raises concerns about the reliability of the solutions they provide, especially in educational contexts. Traditional prompt engineering, on the other hand, leverages carefully designed input prompts to guide LLMs to generate desired responses without additional training(Sahoo et al., 2024). This approach is highly dependent on the quality of the prompts and often struggles to capture the subtle mathematical dependencies inherent in complex problems.

Recent studies in general question-answering tasks have demonstrated the efficacy of integrating Knowledge Graphs (KGs) with language models to enhance reasoning abilities(Sun et al., 2023; Wen et al., 2023; Jiang et al., 2023). KGs provide structured knowledge representations that can offer clear reasoning paths for language models, reducing their hallucinations. Even if the knowledge graph is incomplete, it can still increase the reasoning ability of language models to some extent(Saxena et al., 2020; Xu et al., 2024).

However, applying KGs to MWP solving presents unique challenges. While the integration of KGs with LLMs has shown potential in areas such as commonsense reasoning(Liu et al., 2021; Wang et al., 2023; Toroghi et al., 2024), its application to MWP solving remains relatively unexplored. MWPs require models not only to identify explicit keywords mentioned in the text but also to infer and apply latent mathematical concepts(e.g., rate calculations, proportional relationships). These concepts are often implied rather than stated directly, necessitating a level of abstraction and conceptual understanding that goes beyond surface-level text analysis(Verschaffel et al., 2020). Additionally, solving MWPs involves sequential and interdependent reasoning steps that must be executed in the correct order to arrive at the correct solution.

In order to explore the possibility of applying KGs to the field of MWP solving, we propose a novel framework, which dynamically integrates LLMs with mathematical KGs to enhance both the reasoning ability and reliability. Our approach employs LLMs as linguistic parsers to decompose problems into mathematical concepts while simultaneously activating relevant entities in the KG. The retrieved KG paths then serve as explicit reasoning scaffolds, guiding the LLM to solve the problems step by step. We call this framework KEMRP (Knowledge-Enhanced Math Reasoning Path).

The main contributions are summarized as follows:

1) We propose a novel framework named KEMRP that combines mathematical KGs and LLMs to solve MWPs for the first time, which leverages structured data from KGs and the inherent reasoning abilities of LLMs to enhance the accuracy and reliability in solving MWPs.

2) We construct two mathematical knowledge graphs: MWPEN-KG (English) and MT700-KG (Chinese). These structured resources explicitly ground mathematical concepts and their relationships for MWP solving.

3) We conducted extensive experiments with four backbone models (DeepSeek-v3, GPT-4o, GPT-3.5-Turbo, and Qwen-Turbo) across five MWP benchmarks: MAWPS(Koncel-Kedziorski et al., 2016), MathQA(Amini et al., 2019), Math23K(Wang et al., 2017a), Ape210k(Zhao et al., 2020), and CM17K(Qin et al., 2021). The results show consistent superiority over conventional prompting methods, achieving new state-of-the-art performance on all five datasets, and validating the effectiveness of KG-enhanced reasoning.

2 Related Work

2.1 KG-enhanced LLM

Due to the hallucination of LLMs and their lack of interpretability, researchers proposed to use KGs to enhance LLMs. There are three main directions: (1) incorporating KG into LLM in the pre-training stage(Zhang et al., 2019; Xiong et al., 2019; Sun et al., 2021; Wang et al., 2021), (2) fine-tuning LLMs with KGs(Liu et al., 2020; Ye et al., 2022; Wang et al., 2022; Dernbach et al., 2024; Tian et al., 2024), and (3) integrating KGs during the inference stage(Jiang et al., 2023; Wen et al., 2023; Sun et al., 2023; Dong et al., 2025; Jia et al., 2025). For (1), KGs are typically incorporated into training objectives, which usually have a higher training cost, but have the best effect for specific goals. For (2), the architecture of the encoder and decoder is designed to adapt KG structure, enabling models to internalize new knowledge while preserving their inherent reasoning abilities. However, this approach also involves significant costs for knowledge updates. For

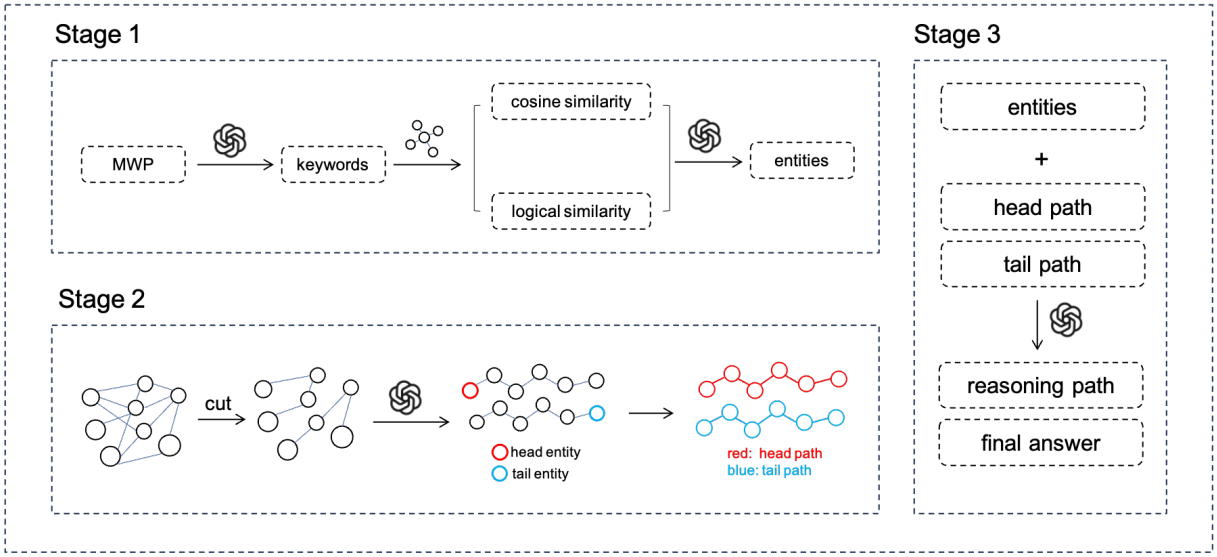

Figure 2: The overview of our framework. In the first stage, some related entities are selected. In the second stage, two reference paths, the head path and tail path, are selected. In the third stage, LLM is prompted to generate the final reasoning path and answer.

(3), KG is converted into natural language and integrated into prompts. Although this approach may not achieve complete knowledge internalization, it leverages the robust abilities of current LLMs to deliver satisfactory results. Besides, this method provides a high degree of scalability and flexibility(Sahoo et al., 2024).

2.2 Math Word Problem

The automatic solving of MWPs has attracted widespread attention for a long time(Ahn et al., 2024). Early approaches primarily relied on rule-based methods and statistical methods, which exhibited considerable limitations in their application scope and performance (Bakman, 2007; Hosseini et al., 2014; Koncel-Kedziorski et al., 2015; Mitra and Baral, 2016; Zhou et al., 2015). As Deep Neural Solver(DNS)(Wang et al., 2017b) was proposed, neural network approaches based on encoder-decoder architectures (such as Seq2Seq) were applied to address MWP solving tasks(Wang et al., 2018; Shen and Jin, 2020; Chatterjee et al., 2023; Bin et al., 2023; Raiyan et al., 2023; Hu et al., 2023). In recent years, with the development and widespread application of LLMs, researchers have increasingly focused on using LLMs to build MWP solvers.

Researchers primarily employ two directions to develop MWP solvers utilizing LLMs. The first direction implements knowledge distillation strategies(Liang et al., 2022, 2023a,b), wherein the mathematical problem-solving abilities of large-scale language models are transferred to smaller, specialized models through supervised learning. The second approach leverages prompt engineering, such as Chain-of-Thought (CoT)(Wei et al., 2023; Tan et al., 2024; Kojima et al., 2023; Zhou et al., 2022), where carefully crafted prompts are designed to guide LLMs through the mathematical reasoning process. These methods rely on models' inherent knowledge to solve MWPs. Few researchers explore the integration of external knowledge through KG into LLM for MWP solving. This study works for this.

3 Method

In this section, we present our framework, which consists of three main components: Entity Recognition and Expansion, Path Discovery, and Reasoning Construction. Figure 2 shows the overview of our framework.

1) **Entity Recognition and Expansion:** We identify and expand the set of key entities from problems using LLMs and KG-based similarity matching.

2) **Path Discovery:** We explore and extract relevant reasoning paths from the KG that connect key mathematical concepts.

3) **Reasoning Construction:** External knowledge is integrated into the input of LLM, allowing LLM to take advantage of both its inherent knowledge and external information

```
Role: Mathematics Education Expert, Language Expert
Style: Precise, Intuitive, Concise
Task: You will be given a math problem and some reference materials. The reference materials include some mathematical
concepts that might be related to the problem and two conceptual paths. Please use these materials to solve the math
problem, show your thinking step by step, and create a conceptual path for solving the problem.
Output Format: JSON
Example:
Input:
{
"problem": "Children from XX School's second grade are planting trees along one side of a path. They plant a tree every
2 meters (including trees at both ends of the path). They find that a total of 11 trees are planted. How long is the
path?",
"entities": ["Line Segment Length Calculation", "Tree Planting Problem", "Tree Planting at Both Ends Problem"],
"paths": ["concept1 -> concept2 -> concept3", "concept4 -> concept5 -> concept6"]
}
Output:
{
"path": "concept path generated from the problem, format as [xxx] -> [xxx] -> [xxx]",
"interpret": "your interpretation for the concepts on the concept path",
"analyze": "the process of solving the problem, step by step",
"answer": "Provide the final answer without any extra information, if there are multiple answers, separate them with
commas",
}
Your task:
```

Figure 3: The prompt template for the Reasoning Construction stage. It enhances LLMs with related entities and two paths (head and tail) to generate the final result.

to generate the final reasoning path and the answer.

3.1 Entity Recognition and Expansion

In this stage, we construct an entity set $E^p = \{e_0^p, e_1^p, \ldots, e_N^p\}$ through the integration of LLM and KG to facilitate the final problem-solving process.

For a given a Math Word Problem (MWP) query Q, we initially employ a LLM to extract several mathematical keywords through one-shot prompting. The prompt is given in Appendix B. Subsequently, these identified keywords K and entities E in KG are encoded into dense embeddings respectively. We then compute the cosine similarity between the mathematical entities from the problem and those in the KG, selecting the entities with the highest similarity scores to form the entity set $E^{p0} = \left\{e_0^{p0}, e_1^{p0}, \ldots, e_N^{p0}\right\}$.

Specifically, for Chinese language tasks, we implement an additional semantic rule-based similarity matching approach(We call it Logical Similarity). For each Chinese keyword, we perform a comprehensive comparison with every entity in the KG. Given a keyword $k \in K$ and an entity $e \in E$, we obtain their respective word segmentation sets W_k and W_e. The overlap between these sets is evaluated, and if the number of common elements exceeds predetermined thresholds t_e or t_{ne}, the entity e is incorporated into the entity set E_p. The detailed implementation is outlined in Algorithm 1.

Following entity identification, we extend E^{p0} with the neighbors of all entities in the set E^{p0}. The extended entity set is then refined through LLM-based filtering to eliminate irrelevant entities, resulting in $E^{p1} = \left\{e_0^{p1}, e_1^{p1}, \ldots, e_N^{p1}\right\}$. This process is iteratively executed for m rounds to construct the final refined entity set $E^{pm} = \{e_0^{pm}, e_1^{pm}, \ldots, e_N^{pm}\}$, ensuring both coverage and relevance of the mathematical concepts. For simplicity, we denote E^{pm} as E^p in the following discussions.

3.2 Path Discovery

In this stage, we discover meaningful paths within the knowledge graph that connect relevant mathematical concepts to solve the given problem. This process consists of two main steps: path preparation and path selection. First, we prepare a set of candidate paths derived from the knowledge graph, along with a set of path pairs that include several similar pairs. Then, we select the most relevant paths by identifying start and end entities and analyzing path similarities. The following sections detail these steps.

3.2.1 Path Preparation

The knowledge graph is represented as $G = \{\langle e, r, e' \rangle | e, e' \in \Psi, r \in R\}$, where Ψ and R denote the set of entities and relations, respectively. A triple $\langle e, r, e' \rangle$ represents that there exists a relation r between the entity e and the entity e'. In this step, we first construct a set of candidate paths $p = \{p | p = (e_1, e_2, \ldots, e_n)\}$ within the KG. Specifically, for each entity in the KG, we generate all possible paths originating from that entity, with path lengths ranging from 2 to 6. This length constraint is based on the "six degrees of separation"

Algorithm 1 Logic Similarity Check

Require: $keyword$, $entity$, t_e, t_{ne}
Ensure: boolean $result$

$list1 \leftarrow Segment(keyword)$
$list2 \leftarrow Segment(entity)$
if $len(list1) = len(list2)$ **then**
    $benchmark \leftarrow len(list2) \times t_e$
else
    $benchmark \leftarrow len(list2) \times t_{ne}$
end if
$score \leftarrow 0$
$result \leftarrow False$
for $seg1 \in list1$ **do**
    for $seg2 \in list2$ **do**
        if $seg1 = seg2$ **then**
            $score \leftarrow score + 1$
            if $score \geq benchmark$ **then**
                $result \leftarrow True$
            end if
        end if
    end for
end for

theory(Guare, 2016), which states that any two individuals in social networks can be connected through a chain of at most six intermediate connections. This theory suggests that any two nodes can be linked through a chain of "friend-of-a-friend" relationships within six steps. Subsequently, duplicate paths are removed.

Next, we construct a set of similar path pairs $SP = \{\langle p_i, p_j \rangle | p_i, p_j \in P\}$. Each element in this set is a pair of paths, where the two paths are considered to be similar. Specifically, for any two paths p_i and p_j in the set P, we determine whether they share any common entities. If they do, the pair is added to the set SP.

3.2.2 Path Selection

For a given problem query Q, we prompt LLMs to identify the start entity e_{start} and end entity e_{end} from E^p. The prompt can be found in Appendix B. Subsequently, we extract two subsets from the path set P: the start path set P_{start} containing all paths that contain e_{start}, and the end path set P_{end} containing all paths that contain e_{end}. For each possible pair of paths $\langle p^i_{start}, p^j_{end} \rangle$, where $p^i_{start} \in P_{start}$ and $p^j_{end} \in P_{end}$, we compute their semantic similarity, which depends on whether $\langle p^i_{start}, p^j_{end} \rangle$ is in the set SP and the cosine similarity of p^i_{start} and p^j_{end}. The specific implementation is provided in Algorithm 2. The paths with the highest similarity score, denoted as p_{start} and p_{end}, are selected as the optimal start and end paths respectively.

Algorithm 2 Path Semantic Similarity

Require: $path1$, $path2$, SP
Ensure: Similarity score $score$

$score \leftarrow 0$
if $\langle path1, path2 \rangle \in SP$ **then**
    $score \leftarrow score + 1$
end if
$cos\_score \leftarrow cos\_similarity(path1, path2)$
$score \leftarrow score + cos\_score$
return $score$

3.3 Reasoning Construction

In this stage, LLMs are prompted to generate the final reasoning path and ultimate solution. The prompt is constructed with the following components: role and style, task instructions, the problem statement, the entity set E^p obtained from the first stage, the start path p_{start} and the end path p_{end} derived from the second stage, and an illustrative example. The LLM is expected to produce a reasoning path, an interpretation of the reasoning process, an analysis of the problem, and the final answer. The prompt template is provided in figure 3. Notably, the generated reasoning path can be considered as a reference to augment the mathematical knowledge graph, enhancing its alignment with MWPs.

4 Experiments

4.1 DataSets

MWP We consider five MWP datasets, including two English datasets: MAWPS(Koncel-Kedziorski et al., 2016) and MathQA(Amini et al., 2019), as well as three Chinese datasets: Math23K(Wang et al., 2017a), Ape210K(Zhao et al., 2020) and CM17K(Qin et al., 2021). We rank the difficulty levels of these datasets: MWAPS and Math23k are the easiest, Ape210K is next, and MathQA and CM17k are the most difficult. In order to compare with the SOTA models conveniently, we select the test sets in these datasets for experiments. Following previous work(Zhu et al., 2023), we employ accuracy as the evaluation metric for all datasets.

KG For the Math23k and Ape210k datasets, we employed a Chinese mathematical knowledge

Table 1: Results for different datasets. The SOTA models are: (Toshniwal et al., 2024) for MAWPS, (Zhang and Moshfeghi, 2022) for MathQA, (Tan et al., 2024) for Math23k, (Liang et al., 2023a) for Ape210k and (Liang et al., 2023a) for CM17k.

Method	MAWPS	MathQA	Math23K	Ape210k	CM17k
GPT-3.5-Turbo & IO prompt	0.8414	0.6118	0.7080	0.5938	0.5229
GPT-4o & IO prompt	0.9265	0.8081	0.8770	0.7858	0.7688
GPT-3.5-Turbo & CoT	0.7693	0.3283	0.6190	0.4908	0.3650
GPT-4o & CoT	0.9552	0.7601	0.8970	0.7784	0.7477
Embedding Retrieval	0.9302	0.8131	0.8790	0.8226	0.7940
MindMap	0.9140	0.6124	0.9030	0.8202	0.7230
Prior SOTA	0.9570	0.8300	0.9430	0.7700	0.7880
GPT-4o & Ours	0.9565	0.8154	0.9360	0.8304	**0.8327**
DeepSeek-v3 & Ours	**0.9635**	**0.8318**	**0.9480**	**0.8798**	0.8028

graph from TAL[2], which contains 998 nodes and 1743 edges. We call it 100TAL-KG. As for the MAWPS and MathQA datasets, we constructed a specialized English knowledge graph called MWPEN-KG, comprising 313 nodes and 719 edges. Additionally, for the CM17k dataset, we developed a Chinese knowledge graph named MT700-KG, which has 704 nodes and 724 edges. The details of how we constructed them are presented in the Appendix A.

4.2 Baselines

We employed standard prompting (IO prompt)(Brown et al., 2020), Chain-of-Thought prompting (CoT prompt)(Wei et al., 2023), embedding retrieval, and MindMap(Wen et al., 2023) as baseline models. Notably, both embedding retrieval and MindMap are KG-enhanced LLM methods, similar to our approach. Moreover, for each dataset, we pick previous state-of-the-art (SOTA) works for comparison.

4.3 Implementation Details

Four backbone models were used: DeepSeek-v3, GPT-4o, GPT-3.5-Turbo, and Qwen-Turbo, all accessed through the OpenAI API interface. To optimize data processing efficiency and ensure standardized output, we configured the response format parameter in our API calls to consistently receive responses in JSON format. In all experiments, the iteration count m in the first stage was set to 3. For the Chinese dataset experiments, we manually evaluated the performance of different similarity threshold parameter values, and the final parameter settings were: t_e was set to 1/2, while t_{ne} was set to 2/3.

[2]https://ai.100tal.com/openData/knowledgeGraph

4.4 Main Results

4.4.1 Comparison to Other Methods

We conducted experiments across five datasets using different approaches. As we can see in Table 1, Chain-of-Thought prompting(CoT) does not consistently outperform standard prompting(IO prompt) in MWP-solving tasks. However, KG-enhanced LLM approaches demonstrate superior performance. The Embedding Retrieval and MindMap approaches surpass both IO prompting and CoT on four datasets (MathQA, Math23K, Ape210k, and CM17k), while our method achieves state-of-the-art performance across all five datasets, with particularly notable improvements on Chinese datasets Ape210k and CM17k, which improve by 10.98% and 4.47% respectively. Significantly, our approach demonstrates substantial improvements over conventional prompting methods on more challenging datasets such as MathQA and CM17k, with performance gains of 2.37% and 6.39% respectively. These results substantiate the effectiveness of our method in handling complex MWP scenarios, validating its robustness and generalizability across varying difficulty levels.

4.4.2 Comparison of Different Backbone

As a plug-and-play framework, we evaluated our method's performance with different backbone models and compared it with Chain-of-Thought prompting. The results are shown in Table 2.

Table 2: Performance of CoT and our method using different backbones.

Method	MAWPS	MathQA	Math23K	Ape210k	CM17k
Qwen-Turbo & CoT	0.9362	0.5688	0.7520	0.6512	0.5329
GPT-3.5-Turbo & CoT	0.7693	0.3283	0.6190	0.4908	0.3650
GPT-4o & CoT	0.9552	0.7601	0.8970	0.7784	0.7477
DeepSeek-v3 & CoT	0.9542	0.7022	0.9330	0.8334	0.7700
Qwen-Turbo & Ours	0.9357	0.6043	0.8260	0.7112	0.6015
GPT-3.5-Turbo & Ours	0.8262	0.3408	0.6690	0.4374	0.3210
GPT-4o & Ours	0.9565	0.8154	0.9360	0.8304	0.8327
DeepSeek-v3 & Ours	0.9635	0.8318	0.9480	0.8798	0.8028

For simpler datasets(MAWPS and Math23K), our method shows greater improvements with models that have relatively weaker reasoning abilities (such as GPT-3.5-Turbo). However, for more challenging datasets(MathQA and CM17k), our method demonstrates higher improvements when paired with models having stronger reasoning abilities (like DeepSeek-v3 and GPT-4o). Notably, when our method is combined with weaker models on difficult datasets, performance actually deteriorates. We analyze these phenomena as follows:

- **For simple datasets:** Weaker models typically lack basic knowledge foundations. Our method directly fills these knowledge gaps, leading to significant performance improvements. However, stronger models already possess robust reasoning abilities. Since our method introduces information they likely already know, the improvement margin is limited.

- **For difficult datasets:** Strong models inherently possess powerful reasoning abilities, and our method can effectively assist them in their thought process, reducing the difficulty of reasoning. However, weaker models, limited in both abilities and knowledge base, may actually perform worse when presented with knowledge beyond their processing ability.

4.5 Ablation Study

4.5.1 Performance with Only Entities or Only Paths

To investigate the effectiveness of each stage, we conducted ablation studies by removing either entities or paths from the prompt while keeping other components. Using Deepseek-v3 as our backbone model, we performed experiments across five datasets, with results shown in Table 3. Overall, performance declines regardless of whether entities or paths are missing, with a greater decline observed when entities are absent in the difficult datasets MathQA and CM17k. In contrast, on the simpler datasets MAWPS and Math23K, a greater performance decline occurs when paths are missing. One possible reason is that for strong models, introducing "extra" conceptual relations is not important for unlocking their reasoning potential when solving simple mathematical problems. However, when addressing difficult mathematical problems, referencing reasoning paths is more likely to help the model discover hidden solutions.

4.5.2 Performance with Different Iteration Count

To investigate the impact of the iteration count m in the first stage, we selected 1000 MWPs from the MAWPS and CM17k datasets, respectively. We conducted experiments with parameter settings of $m = 1$, $m = 2$, and $m = 3$, using GPT-4o as the backbone. As is shown in Table 4, for simple datasets, increasing the iteration count m has little impact on performance, which may be due to the limited room for effect improvement. In contrast, increasing the iteration count m improves the performance of our method on difficult datasets. This result also demonstrates the potential of our approach in solving difficult MWPs. However, as the iteration count increases, the associated costs also rise, making it essential to select an appropriate value.

4.6 Further Analysis: Why our method works on MWPs?

One notable feature of MWPs is the presence of numerous implicit details, which often complicate the problem-solving process. Recalling the pro-

Table 3: Performance of our method with different components.

Method	MAWPS	MathQA	Math23K	Ape210k	CM17k	Average
without entities	0.9524	0.8237	0.9300	0.8188	0.7905	0.8630
without paths	0.9538	0.8181	0.9290	0.8240	0.7799	0.8610
KEMRP	0.9635	0.8318	0.9480	0.8798	0.8028	0.8850

Problem: To beautify the campus, the school plans to purchase a total of 200 plants of types A and B for landscaping. Each type A plant costs 80, and each type B plant costs 100. If the total expenditure on types A and B plants is 17,600, how many plants of each type does the school purchase?

Entities: system of linear equations, equations, algebra, budget and cost, variables, elimination method, substitution method, solution set of equations

Paths: [word problem → variables → equations → system of linear equations → solving systems of equations], [equations → solving systems of equations → elimination method → solution set of equations]

Final reasoning path: [word problem → budget and cost → variables → equations → system of linear equations → solving systems of equations → elimination method → solution set of equations → verify solution]

Answer: 120, 80

Figure 4: An illustration of the application of our method on a sample problem.

Table 4: Performance with different iteration count m.

DataSet	MAWPS	CM17k
m = 1	0.953	0.741
m = 2	0.956	0.772
m = 3	0.957	0.780

cess that we as humans solve MWPs, we first extract given conditions and unknown quantities from the information provided, and then establish relationships between those known conditions and the unknowns. For simple problems, one or two reasoning steps may suffice to reach a solution. For instance, a basic arithmetic problem might only involve addition or subtraction to find the unknown. However, complex problems often require multiple reasoning steps and involve intermediate concepts that may initially seem unrelated to the problem statement.

As illustrated in Figure 4, in the first stage of our method, we extract the explicit known conditions and unknown quantities from the problem and then expand upon them. As the number of iterations increases, more and more hidden related entities are discovered, expanding the reasoning scope for the LLMs. In the second stage, we simulate human thought processes, firstly identify the start and end entities directly related to the problem, and then seek potential paths to connect these entities, providing a more direct scaffold for the LLMs. In the third stage, we guide the LLMs through prompts to utilize the information gathered in the previous stages, constructing a logical reasoning path step by step to arrive at the final solution. By systematically guiding the LLMs through these stages, we enhance their ability to handle MWPs, ensuring they can navigate the complex relationships and concepts inherent in these problems.

5 Conclusion

In this paper, we present KEMRP, a novel framework that pioneers the integration of LLMs with KGs to enhance MWP-solving abilities. Our approach establishes explicit reasoning paths through KG integration, thereby improving both performance and reliability while addressing key challenges in MWP solving. To validate the effectiveness of our approach, we conducted extensive experiments across five different benchmarks using four LLMs as backbones. The results demonstrate the framework's robust effectiveness. To accommodate existing MWP datasets, we constructed two mathematical knowledge graphs (MWPEN-KG and MT700-KG). We hope this study will offer new research directions in the field of automated MWP solving and drive further advancement in this domain. Furthermore, we believe that the integration of knowledge graphs with LLMs not only improves problem-solving accuracy but also provides crucial insights for building more reliable and transparent AI systems.

Limitations

Although our framework demonstrates significant advancements in solving MWPs through the integration of LLM and KG, there remain certain limitations. First, the effectiveness of KEMRP relies on the completeness and accuracy of the knowledge graphs. Missing concepts, incomplete relations, or errors within the KGs can lead to incorrect solutions. Second, the performance of the framework depends on the abilities of the backbone LLM. If the language model has biases, produces inaccurate information, or lacks certain knowledge—such as GPT-3.5-Turbo's weaker performance on complex tasks—these issues can affect the final answers, even when guided by KGs. Lastly, The multi-stage framework introduces additional computational latency compared to standalone LLM inference. This could hinder real-time applications, especially in resource-constrained environments. Addressing these limitations will be crucial for advancing KG-enhanced LLM frameworks toward robust, scalable, and truly interpretable MWP solvers.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *Preprint*, arXiv:math/0701393.

Yi Bin, Mengqun Han, Wenhao Shi, Lei Wang, Yang Yang, See-Kiong Ng, and Heng Shen. 2023. Non-Autoregressive Math Word Problem Solver with Unified Tree Structure. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3290–3301, Singapore. Association for Computational Linguistics. EMNLP 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Oishik Chatterjee, Isha Pandey, Aashish Waikar, Vishwajeet Kumar, and Ganesh Ramakrishnan. 2023. WARM: A Weakly (+Semi) Supervised Model for Solving Math word Problems. *arXiv preprint*. COLING 2022.

Stefan Dernbach, Khushbu Agarwal, Alejandro Zuniga, Michael Henry, and Sutanay Choudhury. 2024. Glam: Fine-tuning large language models for domain knowledge graph alignment via neighborhood partitioning and generative subgraph encoding. In *Proceedings of the AAAI Symposium Series*, volume 3, pages 82–89.

Zixuan Dong, Baoyun Peng, Yufei Wang, Jia Fu, Xiaodong Wang, Xin Zhou, Yongxue Shan, Kangchen Zhu, and Weiguo Chen. 2025. EffiQA: Efficient question-answering with strategic multi-model collaboration on knowledge graphs. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7180–7194, Abu Dhabi, UAE. Association for Computational Linguistics.

John Guare. 2016. Six degrees of separation. In *The Contemporary Monologue: Men*, pages 89–93. Routledge.

Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. 2024. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1):45–74.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Yuxuan Hu, Jing Zhang, Haoyang Li, Cuiping Li, and Hong Chen. 2023. A Generation-based Deductive Method for Math Word Problems. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1737–1750, Singapore. Association for Computational Linguistics. EMNLP 2023.

Mingyi Jia, Junwen Duan, Yan Song, and Jianxin Wang. 2025. medIKAL: Integrating knowledge graphs as assistants of LLMs for enhanced clinical diagnosis on EMRs. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9278–9298, Abu Dhabi, UAE. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157.

Zhenwen Liang, Dian Yu, Xiaoman Pan, Wenlin Yao, Qingkai Zeng, Xiangliang Zhang, and Dong Yu. 2023a. Mint: Boosting generalization in mathematical reasoning via multi-view fine-tuning. *arXiv preprint arXiv:2307.07951*.

Zhenwen Liang, Wenhao Yu, Tanmay Rajpurohit, Peter Clark, Xiangliang Zhang, and Ashwin Kalyan. 2023b. Let GPT be a Math Tutor: Teaching Math Word Problem Solvers with Customized Exercise Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14384–14396, Singapore. Association for Computational Linguistics. EMNLP 2023.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: Numeracy-Augmented Pre-training for Math Word Problem Solving. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 997–1009, Seattle, United States. Association for Computational Linguistics. NAACL 2022.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6418–6425.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153, Berlin, Germany. Association for Computational Linguistics.

Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. *Preprint*, arXiv:2107.01431.

Syed Rifat Raiyan, Md Nafis Faiyaz, Shah Md. Jawad Kabir, Mohsinul Kabir, Hasan Mahmud, and Md Kamrul Hasan. 2023. Math Word Problem Solving by Generating Linguistic Variants of Problem Statements. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 362–378, Toronto, Canada. Association for Computational Linguistics. ACL 2023.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2924–2934.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

Wenting Tan, Dongxiao Chen, Jieting Xue, Zihao Wang, and Taijie Chen. 2024. Teaching-inspired integrated prompting framework: A novel approach for enhancing reasoning in large language models. *Preprint*, arXiv:2410.08068.

Shiyu Tian, Yangyang Luo, Tianze Xu, Caixia Yuan, Huixing Jiang, Chen Wei, and Xiaojie Wang. 2024. Kg-adapter: Enabling knowledge graph integration in large language models through parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3813–3828.

Armin Toroghi, Willis Guo, Mohammad Mahdi Abdollah Pour, and Scott Sanner. 2024. Right for right reasons: Large language models for verifiable commonsense knowledge graph question answering. *arXiv preprint arXiv:2403.01390*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Preprint*, arXiv:2402.10176.

Lieven Verschaffel, Stanislaw Schukajlow, Jon Star, and Wim Van Dooren. 2020. Word problems in mathematics education: A survey. *Zdm*, 52:1–16.

Jianing Wang, Wenkang Huang, Qiuhui Shi, Hongbin Wang, Minghui Qiu, Xiang Li, and Ming Gao. 2022. Knowledge prompting in pre-trained language model for natural language understanding. *arXiv preprint arXiv:2210.08536*.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017a. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017b. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Yujie Wang, Hu Zhang, Jiye Liang, and Ru Li. 2023. Dynamic heterogeneous-graph reasoning with language models and knowledge representation learning for commonsense question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14048–14063.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. *arXiv preprint arXiv:1912.09637*.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *arXiv preprint arXiv:2404.14741*.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. 2022. Ontology-enhanced prompt-tuning for few-shot learning. In *Proceedings of the ACM Web Conference 2022*, pages 778–787.

Jiaxin Zhang and Yashar Moshfeghi. 2022. Elastic: Numerical reasoning with adaptive symbolic compiler. *Preprint*, arXiv:2210.10105.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822, Lisbon, Portugal. Association for Computational Linguistics.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2023. Solving Math Word Problems via Cooperative Reasoning induced Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4471–4485, Toronto, Canada. Association for Computational Linguistics. ACL 2023.

A KG

For MT700-KG, we manually extracted mathematical concepts from the Chinese mathematics curriculum standards and textbooks and established connections between them. For MWPEN-KG, we used DeepSeek-v3 to extract keywords from problems in MAWPS and MathQA, then rigorously filtered out irrelevant keywords and established relationships among the remaining ones. We defined five types of relationships: composition, alias, prerequisite, and related. Finally, we used DeepSeek-v3 to verify all generated triples and artificially add important entities and relationships that are not covered. All relevant prompts are shown in Figure 5-7.

B Prompts

The prompts in our method are presented in Figure 8-14.

```
Role: Mathematics Education Expert, Language Expert
Task: Determine whether the given word is a math keyword.
Output format: JSON
Example:
Input: "Pythagorean Theorem"
Output:
{
     "result": true
}
Your task:
```

Figure 5: The prompt to determine whether a word is a mathematical keyword.

```
Role: Mathematics Education Expert, Language Expert
Task: Determine if there is a relationship between the two given mathematical keywords, and if so, what the relationship is.
Note: There are four types of relationships: composition, alias, prerequisite, and related.
    Assume there are concepts A and B:
    Composition means A is a component of B, or B is a component of A, such as the sine function and trigonometric functions.
    Alias means B is another term for A, such as Pythagorean Theorem and Right Triangle Theorem.
    Prerequisite means A is prior knowledge for B, or B is prior knowledge for A, such as triangles and the area of a triangle.
    Related means A and B have a relationship but do not fall into the above three categories.
    You have to strictly determine.
Output format: JSON
Example1:
Input: ["Pythagorean Theorem", "Right Triangle Theorem"]
Output:
{
     "relationship": "Alias"
}
Example2:
Input: [Pythagorean Theorem], [equation]
Output:
{
     "relationship": null
}
Your task:
```

Figure 6: The prompt to generate relationships between keywords.

```
Role: Mathematics Education Expert, Language Expert
Task: Determine whether two mathematical keywords have link. You must be very very strict. That is to say, if two keywords have no
direct relationship, you should say no.
Output format: JSON
Example:
Input: [Pythagorean Theorem], [Right Triangle Theorem]
Output: {
     "has_link": true
}
Again, you must be very strict.
Your task:
```

Figure 7: The prompt to check the correctness of triples.

```
Role: Mathematics Education Expert
Style: Direct, Intuitive, Concise
Task: Extract mathematical concepts from math problems.
Output Format: JSON
Example:
Input:
Children from XX School's second grade are planting trees along one side of a path. They plant a tree every 2 meters
(including trees at both ends of the path). They find that a total of 11 trees are planted. How long is the path?
Output:
{
    "keywords": ["combined work rates", "linear equations", "solving for variables"]
}
Your task:
```

Figure 8: The prompt to extract keywords from mathematical problems in the first stage.

```
角色：数学教育专家
风格：一针见血、直观、简练
任务：从数学问题中提取数学概念。
输出格式：JSON
例子：
输入：镇海雅乐学校二年级的小朋友到一条小路的一边植树．小朋友们每隔2米种一棵树（马路两头都种了树），最后发现一共种了11棵，这条小路长多少米．
输出：
{
    "keywords": ["植树问题", "线段长度计算"]
}
你的任务：
```

Figure 9: Chinese prompt to extract keywords from mathematical problems in the first stage.

```
You are a mathematical teacher, please judge whether the keyword is helpful to solve the mathematical problem.
problem: {problem}
keyword: {keyword}
Json is the recommended output format:
{
    "judge": true or false
}
Your task:
```

Figure 10: The prompt to determine whether a keyword is related to a mathematical problem in the first stage.

```
你是一个数学教师，请判断以下关键词是否对于解决该数学问题有帮助：
问题：{problem}
关键词：{keyword}
请用json格式输出：
{
    "judge": true or false
}
你的任务：
```

Figure 11: Chinese prompt to determine whether a keyword is related to a mathematical problem in the first stage.

```
Role: Mathematics Education Expert, Language Expert
Style: Precise, Intuitive, Concise
Task: You will be given a math problem and a set of mathematical concepts. Solving the problem may require the use of
some of these concepts. Mentally work through the problem step by step and select the first and last concepts needed to
solve it.
Output Format: JSON
Example:
Input:
{
"question": "Children from XX School's second grade are planting trees along one side of a path. They plant a tree every
2 meters (including trees at both ends of the path). They find that a total of 11 trees are planted. How long is the
path?",
"concepts": ["work rate problems", "combined work rates", "linear equations", "solving for variables", "time taken to
complete work"]
}
Output:
{
"head": "linear equations",
"tail": "solving for variables"
}
```

Figure 12: The prompt to extract the head entity and the tail entity from mathematical problems in the second stage.

```
角色：数学教育专家、语言专家
风格：精准、直观、简练
任务：你将获得一个数学问题，和一些数学概念，解决这个数学问题可能需要用到其中的概念，请在心中一步一步推演，并选择需要用到的第一个概念和最后一个概
念。
输出格式：JSON
例子：
输入：
{
    "question": "镇海雅乐学校二年级的小朋友到一条小路的一边植树．小朋友们每隔2米种一棵树（马路两头都种了树），最后发现一共种了11棵，这条小路
长多少米．"
    "concepts": ["植树问题", "线段长度计算", "封闭路线植树问题", "两端都不种树问题", "一端种树问题", "两端种树问题"]
}
输出：
{
    "head": "植树问题",
    "tail": "线段长度计算"
}
```

Figure 13: Chinese prompt to extract the head entity and the tail entity from mathematical problems in the second stage.

```
角色：数学教育专家
风格：一针见血、直观、简练
任务：你将获得一个数学问题和一些参考材料，包括一些数学概念和两条概念路径，请试着对这些材料进行分析，选出最适合解决该数学问题的部分，组成一条概念
路径，并一步一步利用这些概念解决问题，给出解决问题的过程。
输出格式：JSON
例子：
输入：
{
    "question": "镇海雅乐学校二年级的小朋友到一条小路的一边植树．小朋友们每隔2米种一棵树（马路两头都种了树），最后发现一共种了11棵，这条小路
长多少米．",
    "entities": "[线段长度计算]、[植树问题]、[两端种树问题]",
    "paths": ["概念1 -> 概念2 -> 概念3", "概念4 -> 概念5 ->概念6"]
}
输出：
{
    "path": "针对问题生成的概念路径，格式：[xxx] -> [xxx] -> [xxx]",
    "interpret": "对这些概念的分析",
    "analyse": "一步一步解决问题的过程",
    "answer": "最终答案，不要有多余的信息，如果有多个答案，用,分隔",
}
你的任务：
```

Figure 14: Chinese prompt template for the third stage.