Buffer of Thoughts: Thought-Augmented Reasoning with Large Language Models

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, Bin Cui

NIPS 2024

Yu Hao Chiang

National Cheng Kung University

國立成功大學
National Cheng Kung University
1931

Authors

- Bin Cui教授(北京大學計算機學院)
  - 北京大學數據與智能研究（DAIR）實驗室 https://github.com/PKU-DAIR
  - Ling Yang (北京大學博士生)
    - Diffusion Model、Large Language Models、Representation Learning
  - Zhaochen Yu (新加坡國立大學碩士生) 於PKU-DAIR實習
- Joseph E. Gonzalez教授(加州大學柏克萊分校 EECS 學院)(Sky Computing Lab, RISE Lab)
    - 與外部系統互動的大型語言模型（LLM）（例如工具使用、RAG、LLM 代理、長上下文視窗）
    - 系統支援大型語言模型部署（例如推理、服務、RAG、批次）
    - 大模型微調
    - 邊緣設備上的機器學習
    - 雲端運算的新方法
    - 高解析度電腦視覺的加速深度學習
    - 自動駕駛汽車軟體平台
  - Shiyi Cao (加州大學柏克萊分校 EECS博士生)
  - Tianjun Zhang (加州大學柏克萊分校 EECS博士生)

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

Introduction

- Large Language Models (LLMs):
  - Showcase strong reasoning capabilities (e.g., GPT-4 [3], PaLM [2], LLaMA [6, 7]).
  - Effective prompting methods further enhance their performance.
- Reasoning Methods:
  - Single-query reasoning: Chain of Thought (CoT) [8], Few-shot prompting [11, 12].
  - Multi-query reasoning: Least-to-Most [16], Tree of Thoughts (ToT) [14], Graph of Thoughts (GoT) [17].
- Limitations:
  - Single-query: Requires task-specific exemplars; lacks universality and generalization.
  - Multi-query: Computationally intensive due to recursive reasoning paths.
  - Both methods neglect reusable high-level guidelines from past tasks.

Introduction

Proposed Solution: Buffer of Thoughts (BoT)

- Thought-augmented reasoning framework for accuracy, efficiency, and robustness.
- Meta-buffer: Stores universal high-level thoughts (thought-templates) for cross-task sharing.
- Buffer-manager: Dynamically updates the meta-buffer to enhance scalability and stability.

Advantages:

- Accuracy: Instantiates high-level thoughts, avoiding the need to build reasoning structures from scratch.
- Efficiency: Leverages historical reasoning structures without complex multi-query processes.
- Robustness: Mimics human thought processes, ensuring consistent performance on similar tasks.

Empirical Results:

- Performance improvements on 10 reasoning-intensive tasks
- Game of 24: +11%、Geometric Shapes: +20%、Checkmate-in-One: +51%.

Introduction

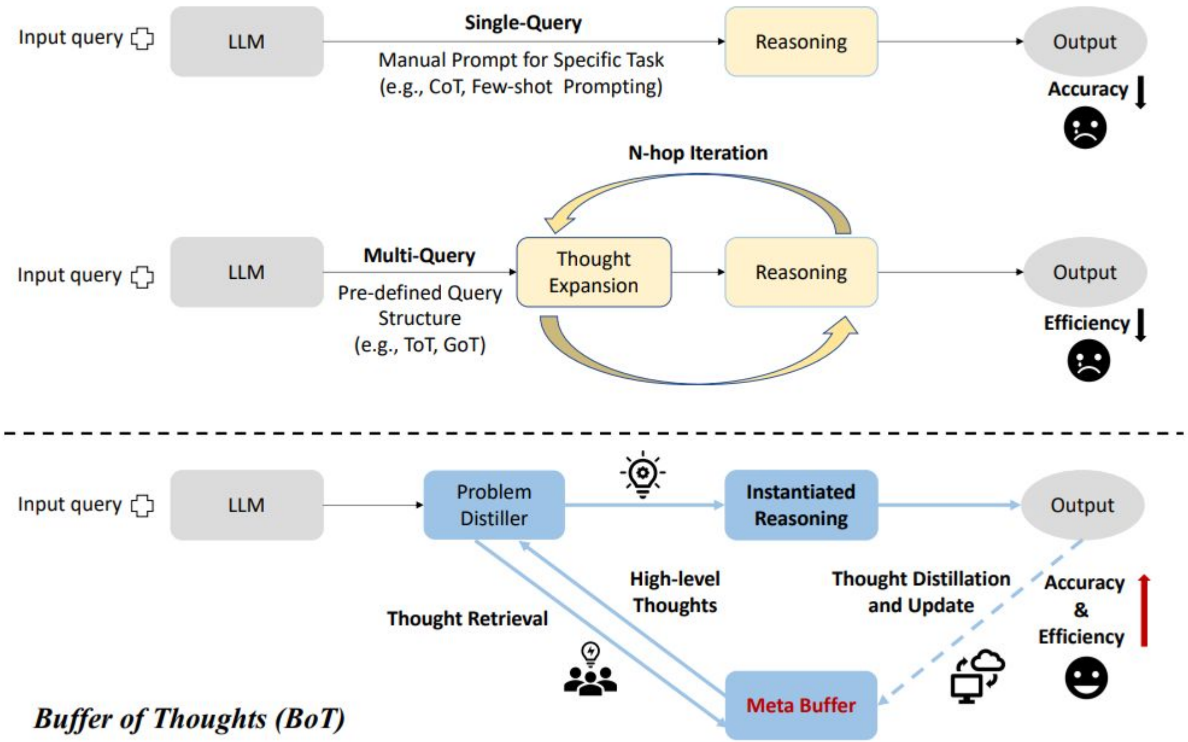


Figure 1: Comparison between single-query [8, 11], multi-query [14, 17], and (c) our BoT methods.

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

Related Work and Disscussions

Retrieval-Augmented Language Models:

- Mitigates hallucination and improves output quality [18–22].
- Queries external databases (billion-level tokens) to retrieve relevant text for better answers [23].
- Applications: multi-modal generation [24, 22, 23, 25], biomedical tasks [26, 27].
- BoT Contribution: Introduces a novel meta-buffer storing high-level thoughts instead of specific instances.

Prompt-based Reasoning with Large Language Models:

- Chain-of-Thought (CoT) [8] and variants [28–30]: Decompose questions into subtasks.
- Tree-of-Thought (ToT) [14] and Graph-of-Thought (GoT) [17]: Dynamic, non-linear reasoning pathways.
- Limitations: These methods have increased resource requirements and time complexity, and rely on manual prompt design, which is usually optimized for a specific task type.

Related Work and Disscussions

Prompt-based Reasoning with Large Language Models:

- Meta Prompting [15]:
    - Task-agnostic form of prompting [15, 40].
    - Recursively guides a single LLM to adaptively address different queries.
- Limitations:
    - Requires a large context window for long prompts.
    - Fails to utilize historical guidelines for similar tasks.

Analogical Reasoning:

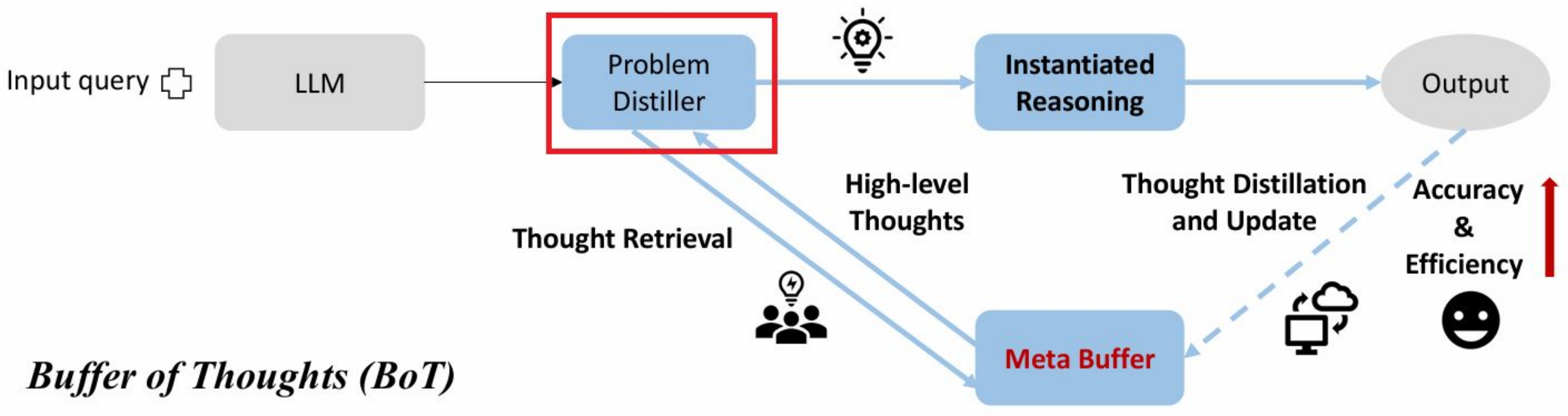
- LLMs perform reasoning like humans via analogous problems [46, 47].
- Analogical Prompting [12], Thought Propagation [48]: Generate and solve analogous problems.
- Thought-Retriever [49]: Uses past intermediate thoughts but limited to textual tasks.
- Limitation: Lacks general approaches for complex reasoning.

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

Buffer of Thoughts - Overview

- Buffer of Thoughts (BoT)
  - A thought-augmented reasoning framework designed to enhance accuracy, efficiency, and robustness.
- Key Components:
  - Problem-Distiller (Section 3.1):
    - Extracts critical task-specific information and relevant constraints.
  - Meta-buffer (Section 3.2):
    - Stores high-level thought templates (thought-templates) for cross-task sharing.
  - Buffer-Manager (Section 3.3):
    - Summarizes the problem-solving process and distills new thought templates to dynamically expand the meta-buffer .



Buffer of Thoughts (BoT)

Buffer of Thoughts (Problem Distiller)

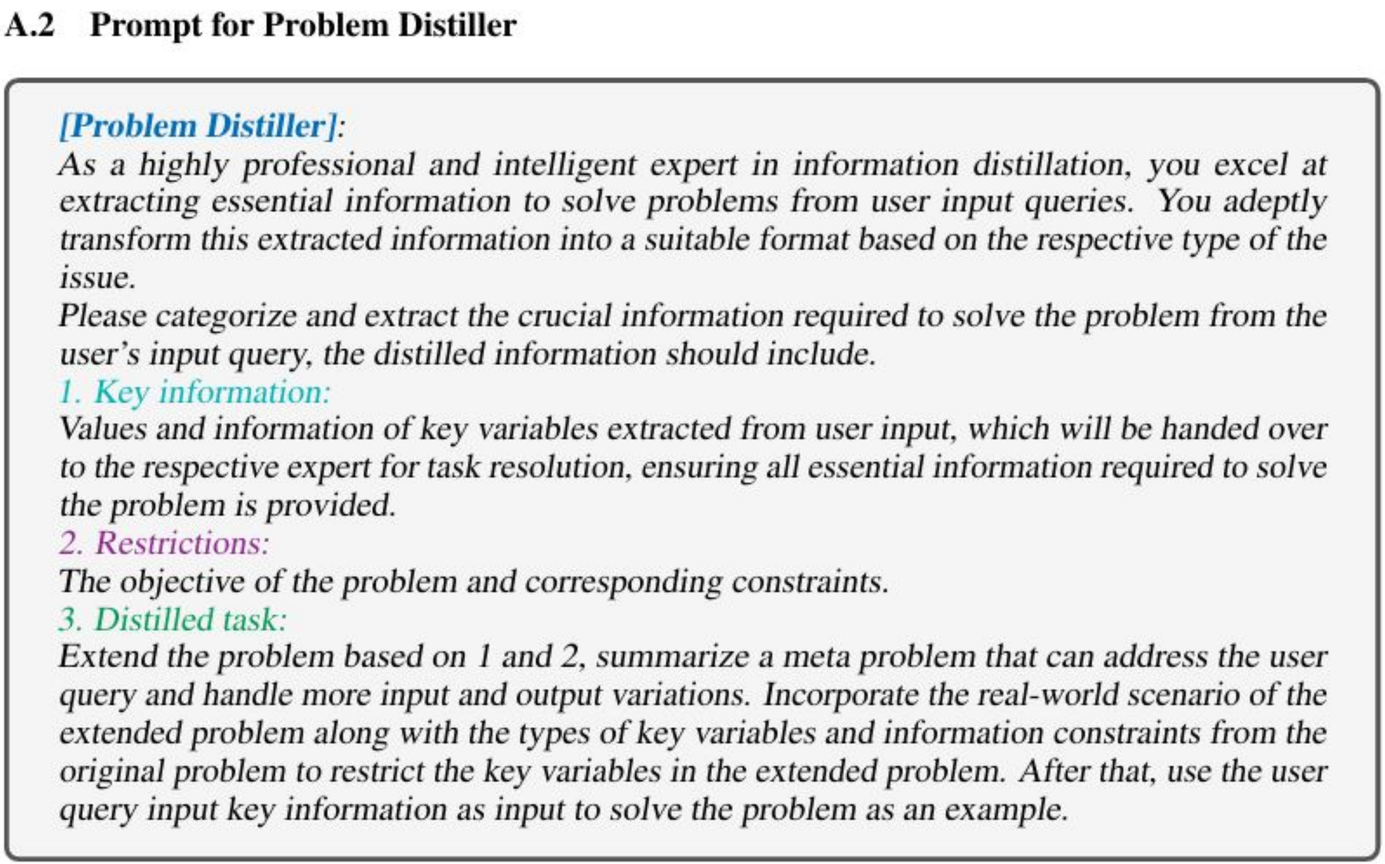
- Complex Task Characteristics:
    - Implicit constraints , Complex object relationships , Intricate variables and parameters.
- Challenges for LLMs:
    - Extracting vital information ,Recognizing potential constraints ,Performing accurate reasoning.
- Problem Condensation and Translation:
    - Problem Distiller
        - Separates task information extraction from reasoning to reduce LLM burden.
        - Distills and formalizes task information using a meta prompt (ϕ).

$$x_d = LLM(\phi(x)), \tag{1}$$

        - Translates problems into high-level concepts for easier reasoning.

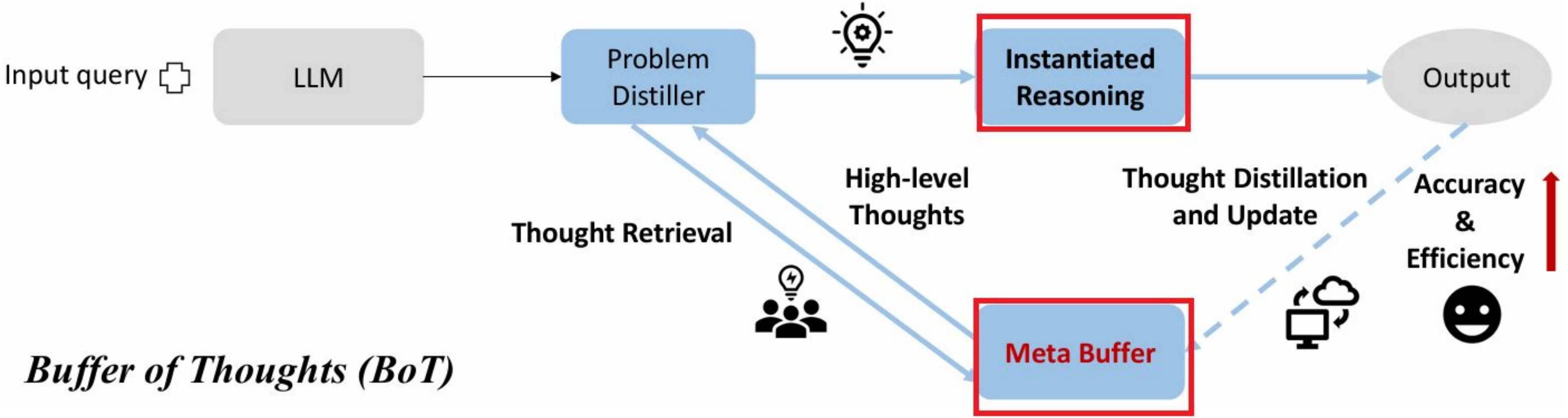
Buffer of Thoughts (Problem Distiller)

1. Extract key variables and their values.
2. Identify objectives and constraints.
3. Extend and structure the problem into a generalized meta problem.



A.2 Prompt for Problem Distiller

[Problem Distiller]:
As a highly professional and intelligent expert in information distillation, you excel at extracting essential information to solve problems from user input queries. You adeptly transform this extracted information into a suitable format based on the respective type of the issue.
Please categorize and extract the crucial information required to solve the problem from the user's input query, the distilled information should include.
1. Key information:
Values and information of key variables extracted from user input, which will be handed over to the respective expert for task resolution, ensuring all essential information required to solve the problem is provided.
2. Restrictions:
The objective of the problem and corresponding constraints.
3. Distilled task:
Extend the problem based on 1 and 2, summarize a meta problem that can address the user query and handle more input and output variations. Incorporate the real-world scenario of the extended problem along with the types of key variables and information constraints from the original problem to restrict the key variables in the extended problem. After that, use the user query input key information as input to solve the problem as an example.

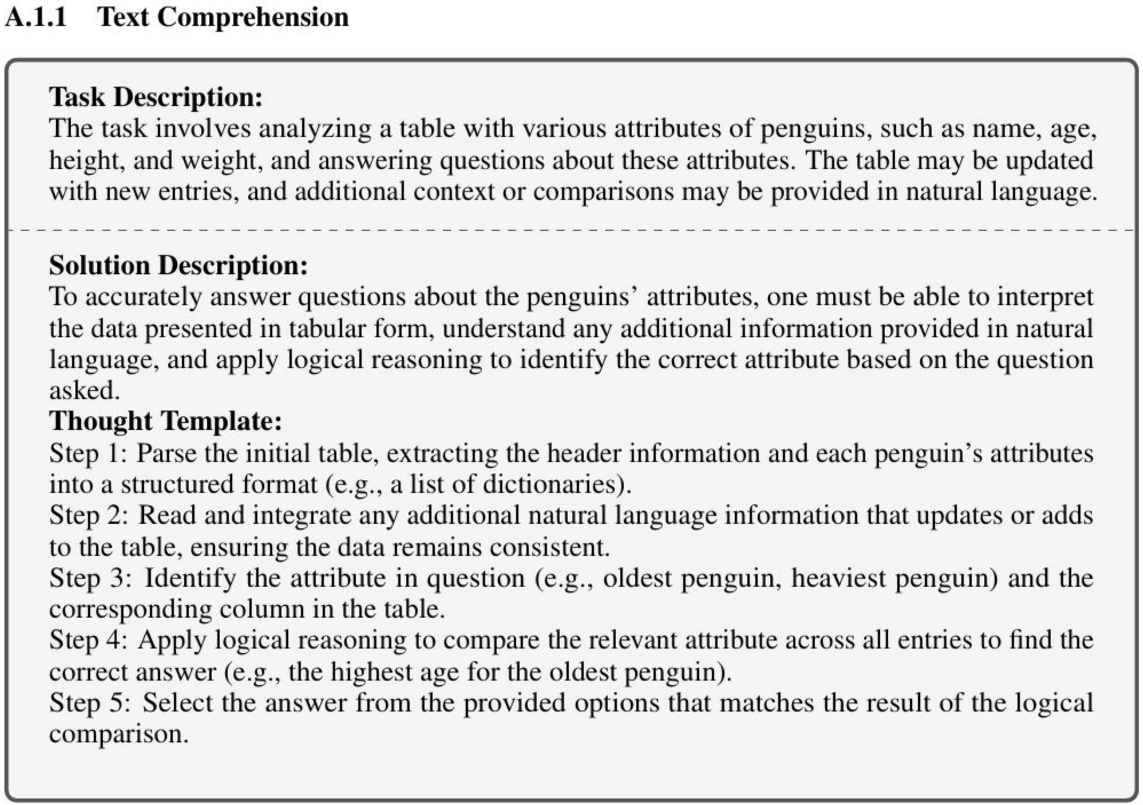

Input query
LLM
Problem Distiller
Instantiated Reasoning
Output
Thought Retrieval
High-level Thoughts
Thought Distillation and Update
Accuracy & Efficiency
Meta Buffer
Buffer of Thoughts (BoT)

Buffer of Thoughts (Thought-Augmented Reasoning with Meta Buffer)

1. Motivation:
    - Inspired by human problem-solving: summarize guidelines and apply them to relevant problems.
    - Meta-buffer: Stores high-level thought templates for adaptability and flexibility.
2. Thought Templates:
    - Serve as general guidelines for solving various tasks. Classified into six categories:
        - Text Comprehension
        - Creative Language Generation
        - Common Sense Reasoning
        - Mathematical Reasoning
        - Code Programming
        - Application Scheduling

Buffer of Thoughts (Thought-Augmented Reasoning with Meta Buffer)

- **Task Description**：分析一個包含企鵝屬性的表格，並回答與屬性相關的問題。
- **Solution Description**：準確解讀表格數據、整合自然語言描述，並進行邏輯推理以生成正確答案。
- **Thought Template**：
  a. 解析表格，提取標題和屬性並轉化為結構化數據。
  b. 整合自然語言上下文，確保數據一致性。
  c. 辨識目標屬性（如最重的企鵝）。
  d. 對屬性進行比較，找出正確答案。
  e. 從選項中選擇匹配結果的答案。



A.1.1 Text Comprehension

Task Description:
The task involves analyzing a table with various attributes of penguins, such as name, age, height, and weight, and answering questions about these attributes. The table may be updated with new entries, and additional context or comparisons may be provided in natural language.

Solution Description:
To accurately answer questions about the penguins' attributes, one must be able to interpret the data presented in tabular form, understand any additional information provided in natural language, and apply logical reasoning to identify the correct attribute based on the question asked.

Thought Template:
Step 1: Parse the initial table, extracting the header information and each penguin's attributes into a structured format (e.g., a list of dictionaries).
Step 2: Read and integrate any additional natural language information that updates or adds to the table, ensuring the data remains consistent.
Step 3: Identify the attribute in question (e.g., oldest penguin, heaviest penguin) and the corresponding column in the table.
Step 4: Apply logical reasoning to compare the relevant attribute across all entries to find the correct answer (e.g., the highest age for the oldest penguin).
Step 5: Select the answer from the provided options that matches the result of the logical comparison.

Buffer of Thoughts (Thought-Augmented Reasoning with Meta Buffer)

Template Retrieval

- Goal: Retrieve the most relevant **thought-template *T_i*** from Meta-buffer based on **problem $\mathbf{x_d}$**.
- Calculate embedding similarity:

$$j = \text{argmax}_i(\text{Sim}(f(x_d), \{f(D_{T_i})\}_{i=1}^{N})), \quad \text{where} \quad \text{Sim}(f(x_d), \{f(D_{T_i})\}_{i=0}^{n}) >= \delta, \qquad (2)$$

- Threshold δ (recommended: 0.5–0.7) if all similarities are below δ, classify the task as a new one.
- where $f(\cdot)$ is a text embedding model.

Instantiated Reasoning

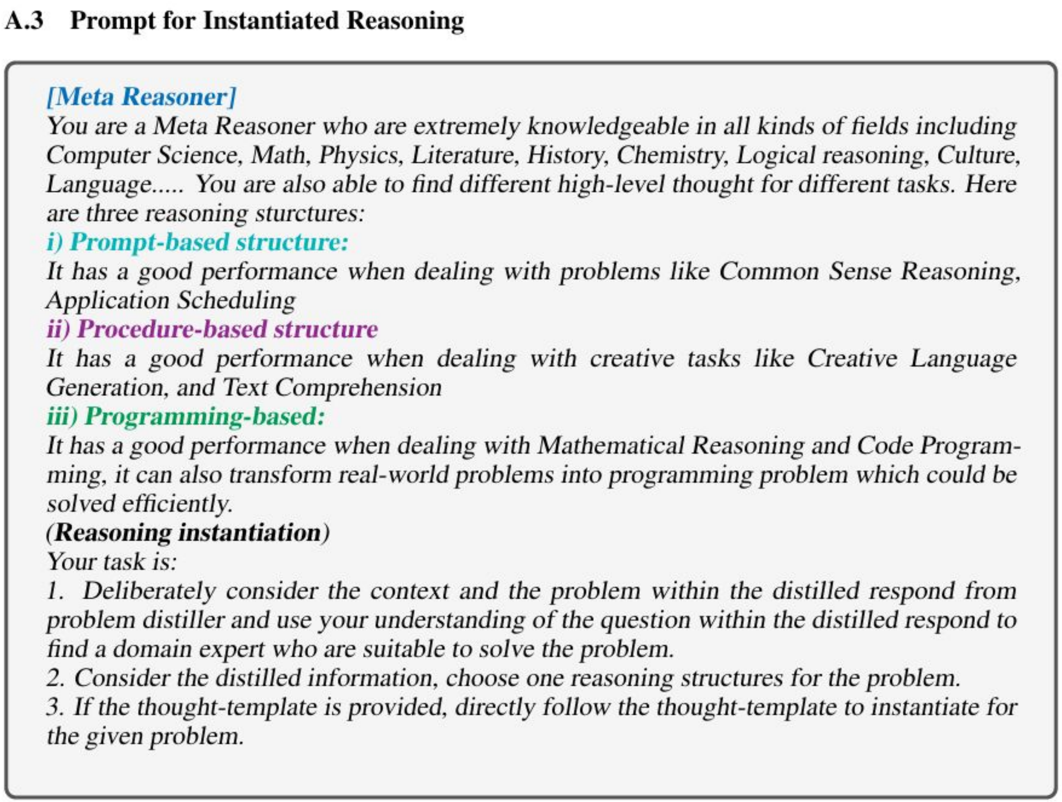
- Matched Template:
  - Use the retrieved template T_j and distilled task information x_d .
  - Generate solution: $$S_x = LLM_{\text{instantiation}}(x_d, T_j), \qquad (3)$$
  - where LLMinstantiation denotes the instantiated reasoner with a LLM.
- New Task: Employ coarse-grained general templates for reasoning.

Buffer of Thoughts (Thought-Augmented Reasoning with Meta Buffer)

- **Prompt-based structure**：
    - **處理**常識推理、應用調度。
- **Procedure-based structure**：
    - 處理創意生成、文本理解。
- **Programming-based structure**：
    - 處理數學推理、程式設計。

任務描述：

- **理解問題並選擇領域專家**　：
    - 從 **Problem Distiller** 提煉的上下文中，提取核心問題並找到適合解決該問題的領域專家
- **選擇適合的推理結構**　：
    - 根據提煉的信息，為該問題選擇一種適合的推理結構
- **應用思維模板進行推理**　：
    - 如果提供了思維模板（thought-template），則直接依據模板進行實例化以解決給定問題。



A.3 Prompt for Instantiated Reasoning

[Meta Reasoner]
You are a Meta Reasoner who are extremely knowledgeable in all kinds of fields including Computer Science, Math, Physics, Literature, History, Chemistry, Logical reasoning, Culture, Language..... You are also able to find different high-level thought for different tasks. Here are three reasoning sturctures:
i) Prompt-based structure:
It has a good performance when dealing with problems like Common Sense Reasoning, Application Scheduling
ii) Procedure-based structure
It has a good performance when dealing with creative tasks like Creative Language Generation, and Text Comprehension
iii) Programming-based:
It has a good performance when dealing with Mathematical Reasoning and Code Programming, it can also transform real-world problems into programming problem which could be solved efficiently.
*(**Reasoning instantiation**)*
Your task is:
1. Deliberately consider the context and the problem within the distilled respond from problem distiller and use your understanding of the question within the distilled respond to find a domain expert who are suitable to solve the problem.
2. Consider the distilled information, choose one reasoning structures for the problem.
3. If the thought-template is provided, directly follow the thought-template to instantiate for the given problem.

Buffer of Thoughts (Buffer Manager)

Purpose:

- Summarizes high-level guidelines and thoughts from problem-solving processes.
- Generalizes specific solutions into thought-templates for reuse across tasks.
- Ensures permanent improvements in accuracy, efficiency, and robustness.

Template Distillation:

- Three-step approach:

$$T_{new} = LLM_{\text{distill}}(x_d, S_x), \quad (4)$$

    a. Core task summarization: Identify problem types and core challenges.

    b. Solution steps description: Outline general steps for solving problems.

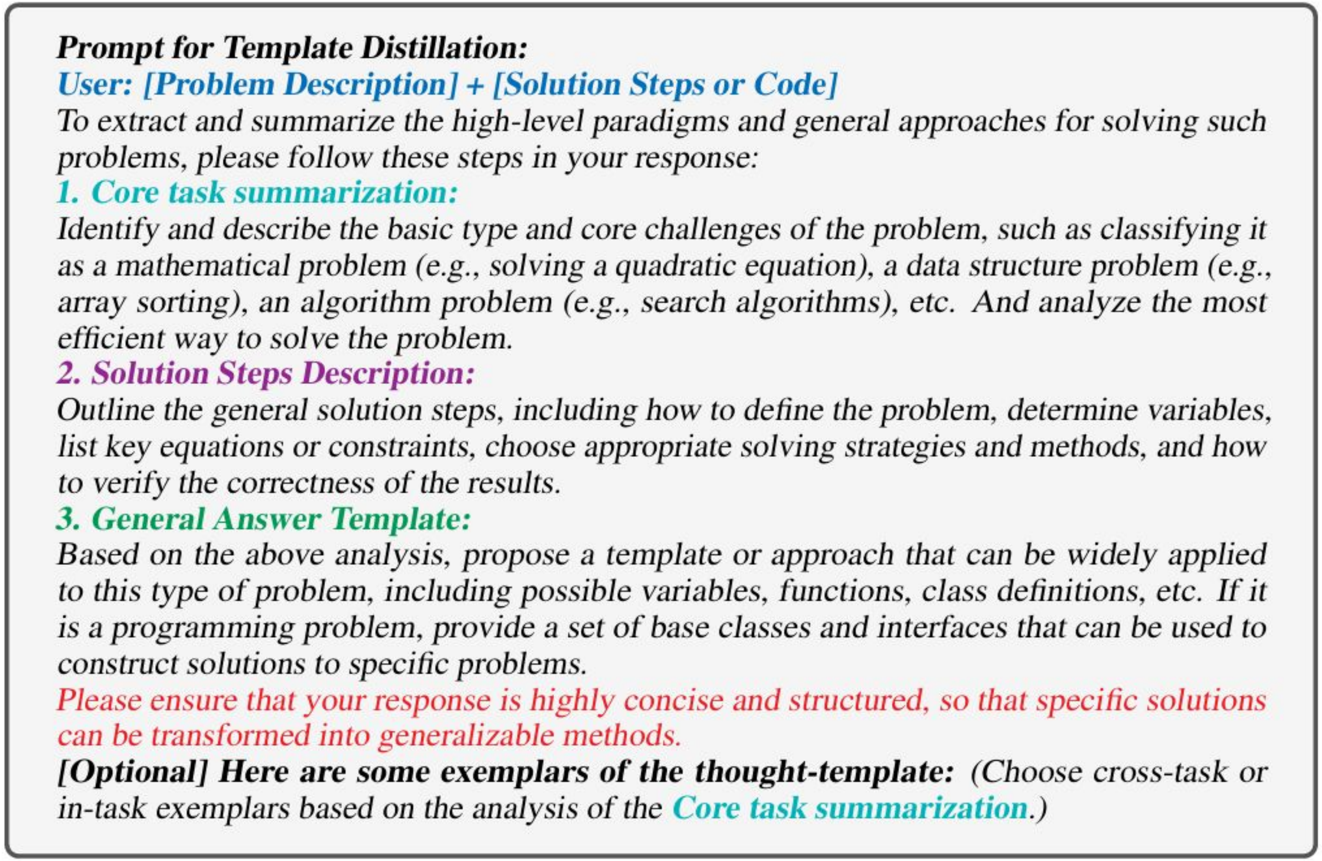
    c. General answering template: Propose reusable templates applicable to similar problems.

- Example-based generalization:

    a. In-task examples: From the same task type.

    b. Cross-task examples: Apply a template from one domain to another.

Buffer of Thoughts (Buffer Manager)



Prompt for Template Distillation:

User: [Problem Description] + [Solution Steps or Code]

To extract and summarize the high-level paradigms and general approaches for solving such problems, please follow these steps in your response:

1. Core task summarization:

Identify and describe the basic type and core challenges of the problem, such as classifying it as a mathematical problem (e.g., solving a quadratic equation), a data structure problem (e.g., array sorting), an algorithm problem (e.g., search algorithms), etc. And analyze the most efficient way to solve the problem.

2. Solution Steps Description:

Outline the general solution steps, including how to define the problem, determine variables, list key equations or constraints, choose appropriate solving strategies and methods, and how to verify the correctness of the results.

3. General Answer Template:

Based on the above analysis, propose a template or approach that can be widely applied to this type of problem, including possible variables, functions, class definitions, etc. If it is a programming problem, provide a set of base classes and interfaces that can be used to construct solutions to specific problems.

Please ensure that your response is highly concise and structured, so that specific solutions can be transformed into generalizable methods.

[Optional] Here are some exemplars of the thought-template: *(Choose cross-task or in-task exemplars based on the analysis of the **Core task summarization**.)*

$$T_{new} = LLM_{\text{distill}}(x_d, S_x), \tag{4}$$

Buffer of Thoughts (Buffer Manager)

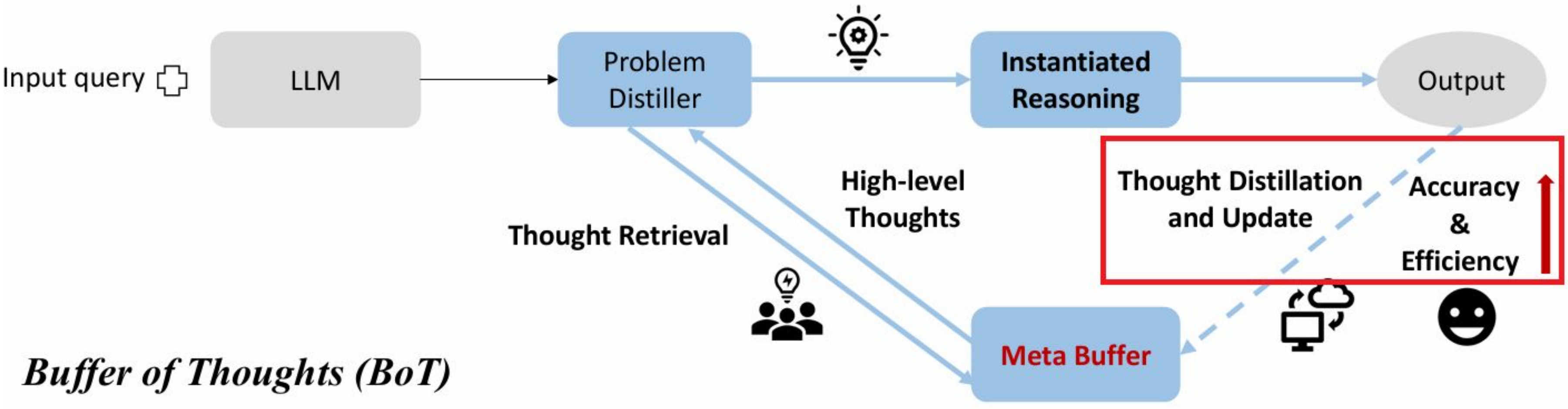
Dynamic Update of Meta-Buffer:

- When to update:
  - For new template: Directly store distilled templates.
  - For template matched: Update only if new insights significantly differ from existing templates

$$\text{Max}(\text{Sim}(f(D_{T_{new}}), \{f(D_{T_i})\}_{i=0}^{n})) < \delta. \tag{5}$$

Benefits:

- Reduces redundancy.
- Ensures the meta-buffer remains lightweight and efficient.



Input query

LLM

Problem Distiller

Instantiated Reasoning

Output

Thought Retrieval

High-level Thoughts

Meta Buffer

Thought Distillation and Update

Accuracy & Efficiency

Buffer of Thoughts (BoT)

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

Experiments (Datasets and Tasks)

1. Game of 24: Form an arithmetic expression using four numbers to equal 24 [14].
2. BIG-Bench Hard (BBH) Tasks [35]:
    - Geometric Shapes
    - Multi-Step Arithmetic Two
    - Word Sorting
3. Reasoning Tasks [50]:
    - Checkmate-in-One: Solve chess problems.
    - Penguins: Answer questions about penguin attributes.
    - Date Understanding: Infer and calculate dates from natural language.
4. Python Programming Puzzles (P3) [51, 52]: Solve challenging Python problems.
5. Multilingual Grade School Math (MGSM) [33]: Math tasks in 10 languages (e.g., Japanese, Swahili).
6. Shakespearean Sonnet Writing [15]: Create sonnets with a fixed rhyme scheme and mandatory keywords.

Experiments (Implementation and Baselines)

- Base Models: GPT-4, Llama3-8B, and Llama3-70B on NVIDIA A100 GPUs.
- Comparison Methods:
  a. Standard Prompting: Direct input query without guidance.
  b. Single-query Methods:
     - Zero-shot CoT [8], PAL [10].
     - Expert Prompting [9]: Integrates expert profiles into queries.
  c. Multi-query Methods:
     - ToT [14], GoT [17]: Explore multiple reasoning paths.
     - Meta-Prompting [15]: Scaffold-based prompting for task adaptability.

Experiments (BoT)

Reasoning Accuracy

- BoT significantly outperforms previous methods across challenging benchmarks:
- Game of 24: +79.4% accuracy compared to GPT-4, +8.4% vs ToT.
- Geometric Shapes: +20% vs Meta-Prompting [15].
- Checkmate-in-One: +51% vs Meta-Prompting [15].
- Leverages thought-templates for optimal reasoning structures, avoiding case-by-case heuristic strategies.

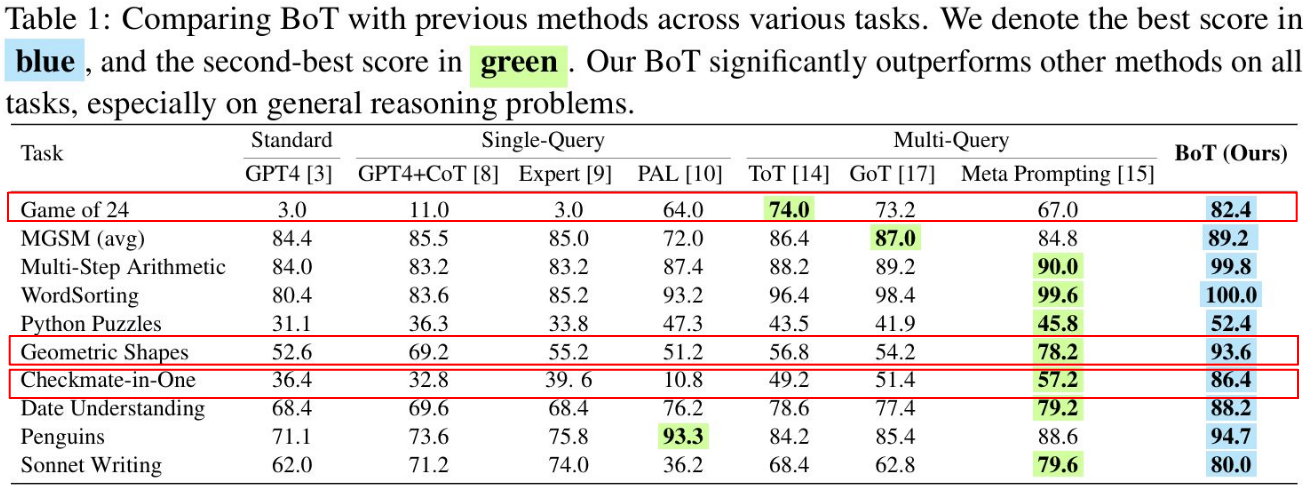


Table 1: Comparing BoT with previous methods across various tasks. We denote the best score in **blue**, and the second-best score in **green**. Our BoT significantly outperforms other methods on all tasks, especially on general reasoning problems.

Task	Standard	Single-Query			Multi-Query			**BoT (Ours)**
	GPT4 [3]	GPT4+CoT [8]	Expert [9]	PAL [10]	ToT [14]	GoT [17]	Meta Prompting [15]	
Game of 24	3.0	11.0	3.0	64.0	**74.0**	73.2	67.0	**82.4**
MGSM (avg)	84.4	85.5	85.0	72.0	86.4	**87.0**	84.8	**89.2**
Multi-Step Arithmetic	84.0	83.2	83.2	87.4	88.2	89.2	**90.0**	**99.8**
WordSorting	80.4	83.6	85.2	93.2	96.4	98.4	**99.6**	**100.0**
Python Puzzles	31.1	36.3	33.8	47.3	43.5	41.9	**45.8**	**52.4**
Geometric Shapes	52.6	69.2	55.2	51.2	56.8	54.2	**78.2**	**93.6**
Checkmate-in-One	36.4	32.8	39. 6	10.8	49.2	51.4	**57.2**	**86.4**
Date Understanding	68.4	69.6	68.4	76.2	78.6	77.4	**79.2**	**88.2**
Penguins	71.1	73.6	75.8	**93.3**	84.2	85.4	88.6	**94.7**
Sonnet Writing	62.0	71.2	74.0	36.2	68.4	62.8	**79.6**	**80.0**

Experiments (BoT)

Reasoning Efficiency

- Comparable reasoning time to single-query methods, significantly faster than multi-query methods like ToT.
- Eliminates iterative searches by directly retrieving and instantiating thought-templates.
- Reduces costs: BoT requires only 12% of the cost of multi-query methods on average.

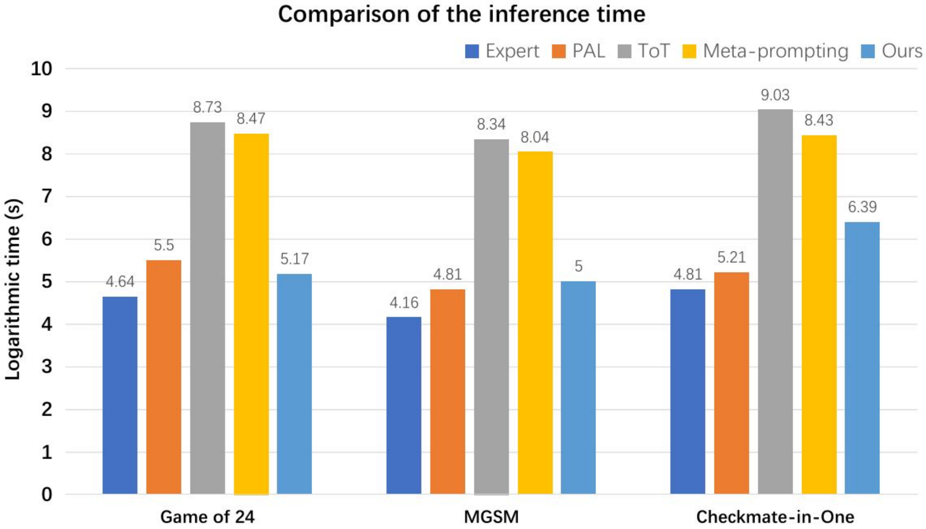


Figure 3: Comparison of **logarithmic inference time** between our Buffer of Thoughts and GPT4 [3], GPT4+CoT [8], Expert-prompting [9], PAL [10], ToT [14] across different benchmarks.

Experiments (BoT)

Reasoning Robustness

- New metric: Success Rate (average accuracy across 1000 examples).
- BoT surpasses the second-best method by 10% in success rate on various tasks.
- Strength lies in the generalization ability of distilled thought-templates, providing stability across diverse tasks.

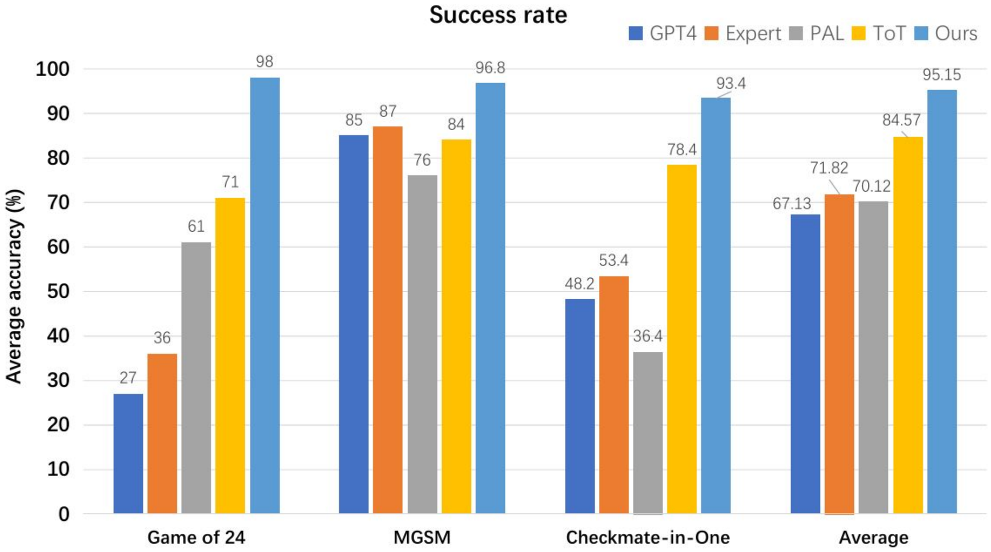


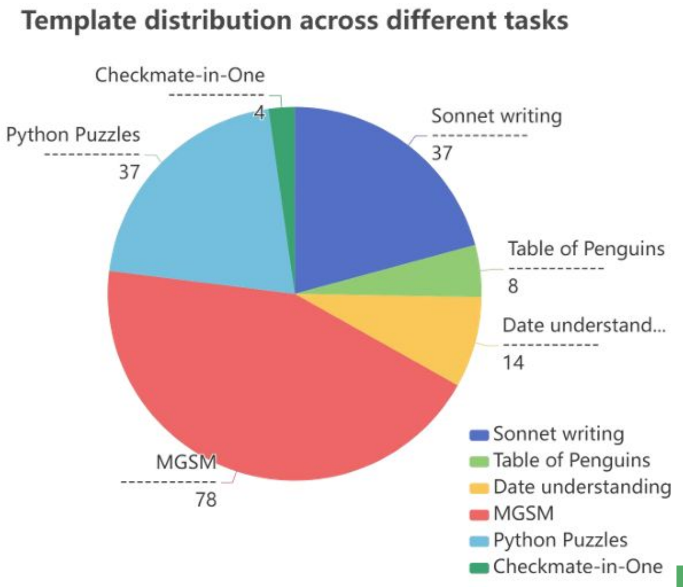
Figure 4: Comparison of reasoning robustness between our Buffer of Thoughts and GPT4 [3], GPT4+CoT [8], Expert-prompting [9], PAL [10], ToT [14] across different benchmarks.

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

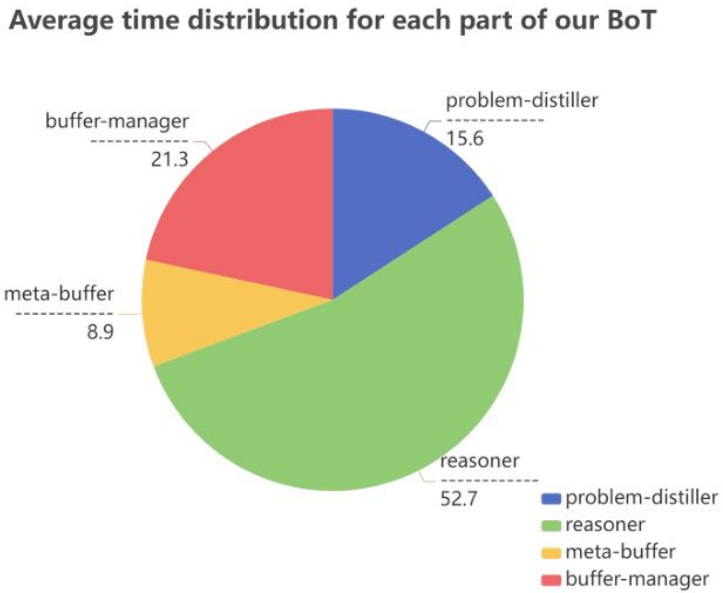
Model Analysis

- Distribution Analysis of Thought-Templates
  - Benchmarks: Six benchmarks with 100 distinct tasks each.
  - More templates in diverse scenarios like MGSM.
  - Fewer, fixed templates in simpler tasks like Checkmate-in-One and Penguins.
- Conclusion: BoT effectively discovers and tailors templates to suit different benchmarks.



Model Analysis

- Distribution Analysis of Time Cost
  - Task distillation and template retrieval: Relatively short.
  - Instantiated reasoning: Longer, depending on task complexity.
- Conclusion: BoT achieves a balanced time distribution, highlighting its efficiency.



Model Analysis

- Model Size vs. Performance Trade-off
  - Without BoT: Smaller models (e.g., Llama3-8B) perform poorly.
  - With BoT: Smaller models (Llama3-8B) can match or surpass larger models (Llama3-70B).
- Impact: Reduces inference cost for tackling complex problems.

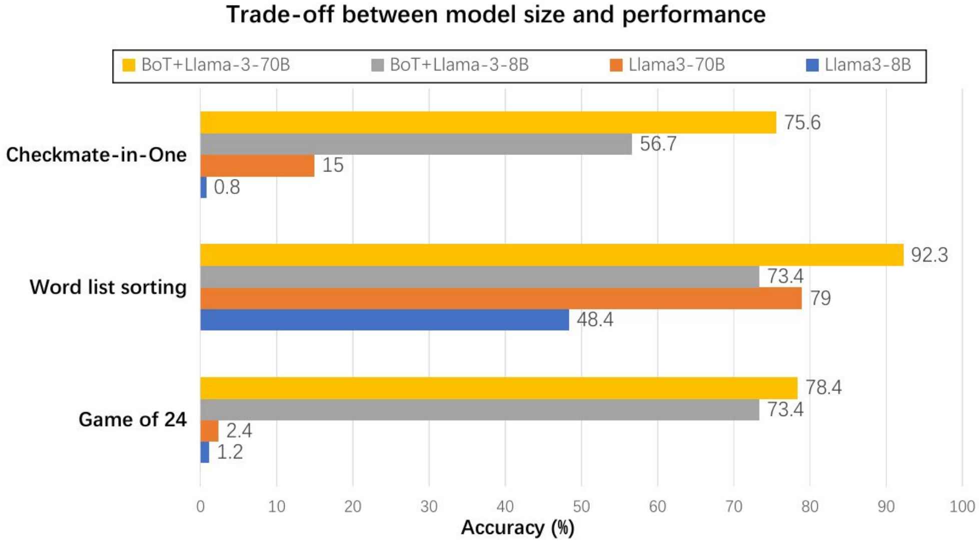


Figure 6: We evaluate the trade-off between model size and performance with Llama3-8B and Llama3-70B models on three challenging benchmarks.

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

Ablation Study (Impact of Problem-Distiller)

- Disabling the problem-distiller results in accuracy decline:
  - Significant for complex problems: Game of 24, Checkmate-in-One.
  - Minor for simpler tasks: Word list sorting, MGSM.
- Conclusion: Problem-distiller is critical for extracting key information and constraints in complex problems.

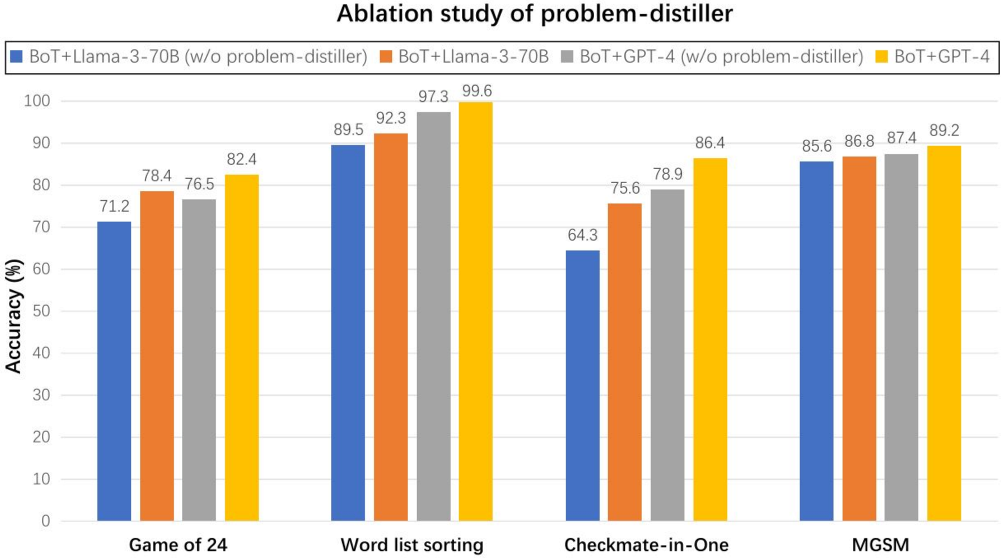


Figure 7: We conduct ablation study on problem-distiller across four benchmarks, employing Llama3-70B and GPT-4 as the base models.

Ablation Study (Impact of Meta-Buffer)

- Removing the meta-buffer causes noticeable performance drops:
  - Particularly in complex reasoning tasks: Game of 24, Checkmate-in-One.
- Conclusion: Meta-buffer is essential for addressing complex problems.

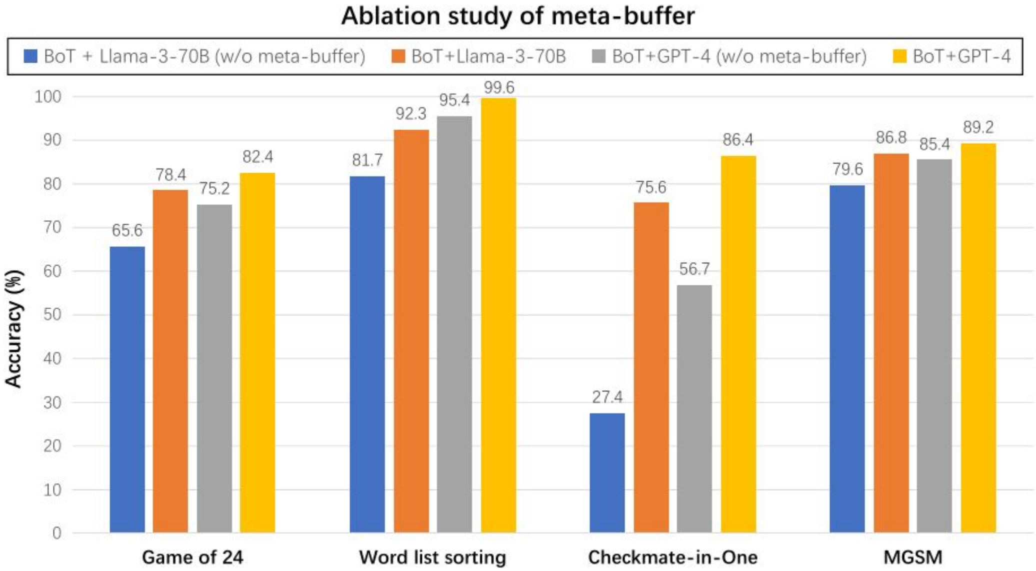


Figure 8: We conduct ablation study on meta-buffer across four benchmarks, employing Llama3-70B and GPT-4 as the base models.

Ablation Study (Impact of Buffer-Manager)

- Process: Four rounds of evaluation with 50 questions per round.
  - With buffer-manager: Accuracy steadily improves across rounds due to meta-buffer expansion.
  - Without buffer-manager: No significant accuracy improvement over rounds.

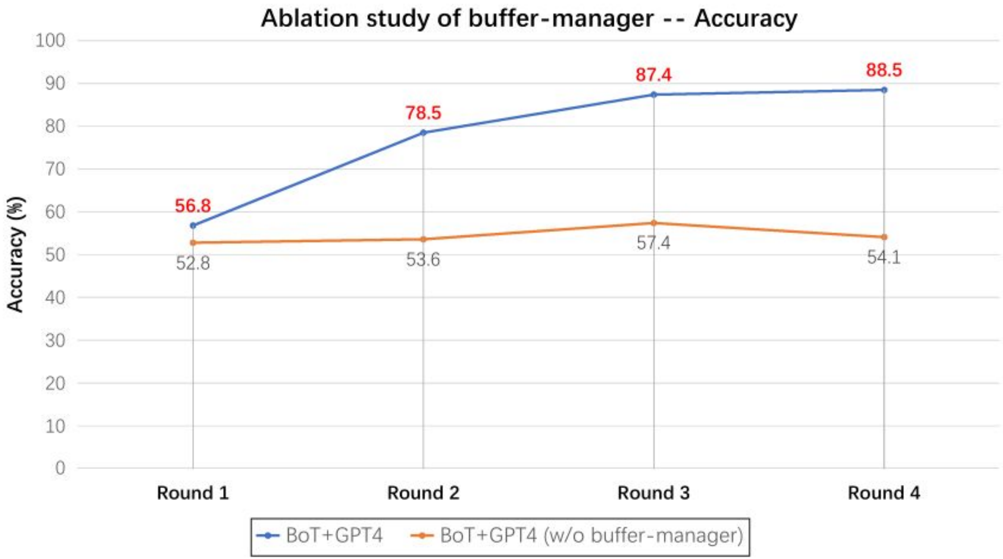


Figure 9: We conduct ablation study on buffer-manager regarding reasoning accuracy across four tasks, employing Llama3-70B and GPT-4 as the base models.

Ablation Study (Impact of Buffer-Manager)

- Process: Four rounds of evaluation with 50 questions per round.
  - With buffer-manager: Reasoning efficiency increases as suitable templates are retrieved.
  - Without buffer-manager: Inefficient due to repetitive reasoning structure reconstruction.

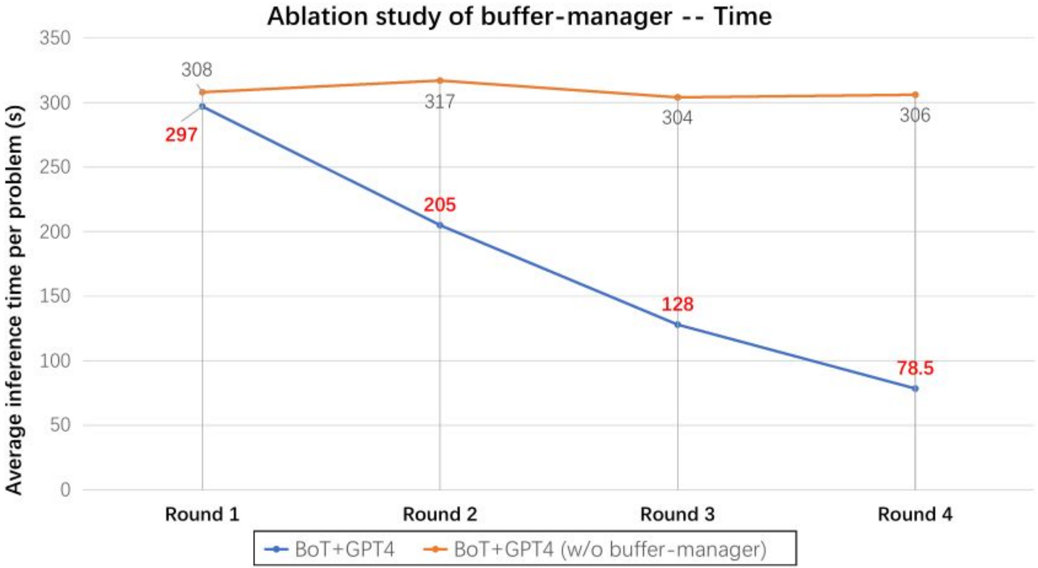


Figure 10: We conduct ablation study on buffer-manager regarding reasoning efficiency across four tasks, employing Llama3-70B and GPT-4 as the base models.

Outline

- Introduction
- Related Work and Disscussions
- Buffer of Thoughts
- Experiments
- Model Analysis
- Ablation Study
- Discussion

Conclusion

- Limitations
  - Limited performance on tasks requiring human-like creativity due to the absence of structured thought-templates.
  - Quality of thought-templates depends on the initial model's reasoning and instruction-following capabilities.
- Future Directions
  - Open-domain integration:
    - Leverage external resources to develop open-domain systems, e.g., agent models [54, 55].
  - Optimized template distillation:
    - Enhance template quality for handling more complex tasks.

Conclusion

- Buffer of Thoughts (BoT):
  - A novel reasoning framework that stores pre-accumulated experiences as thought-templates in a meta-buffer.
- Buffer-Manager:
  - Continuously refines thought-templates and improves problem-solving processes dynamically.
- Performance:
  - Demonstrated state-of-the-art results on 10 challenging tasks with strong prospects for future applications.

學習重點

從靜態檢索到動態知識儲存：

- 使用 Meta Buffer 儲存高層次思維模板，包括任務描述、解決方案描述及具體步驟。
- 支持動態更新與提煉，適應新任務需求。

從局部解決到通用推理：

- 提供模板化的推理過程（Thought Templates），適用於多種任務，消除逐一手動設計範例的需求
- 支持跨任務模板應用，增強泛化能力。

從單次推理到累積優化：

- 通過 Buffer Manager 提煉過往任務的通用規則和高層次指導原則，逐步提升推理框架的能力
- 確保解決方案的累積性，避免知識重複與冗餘。

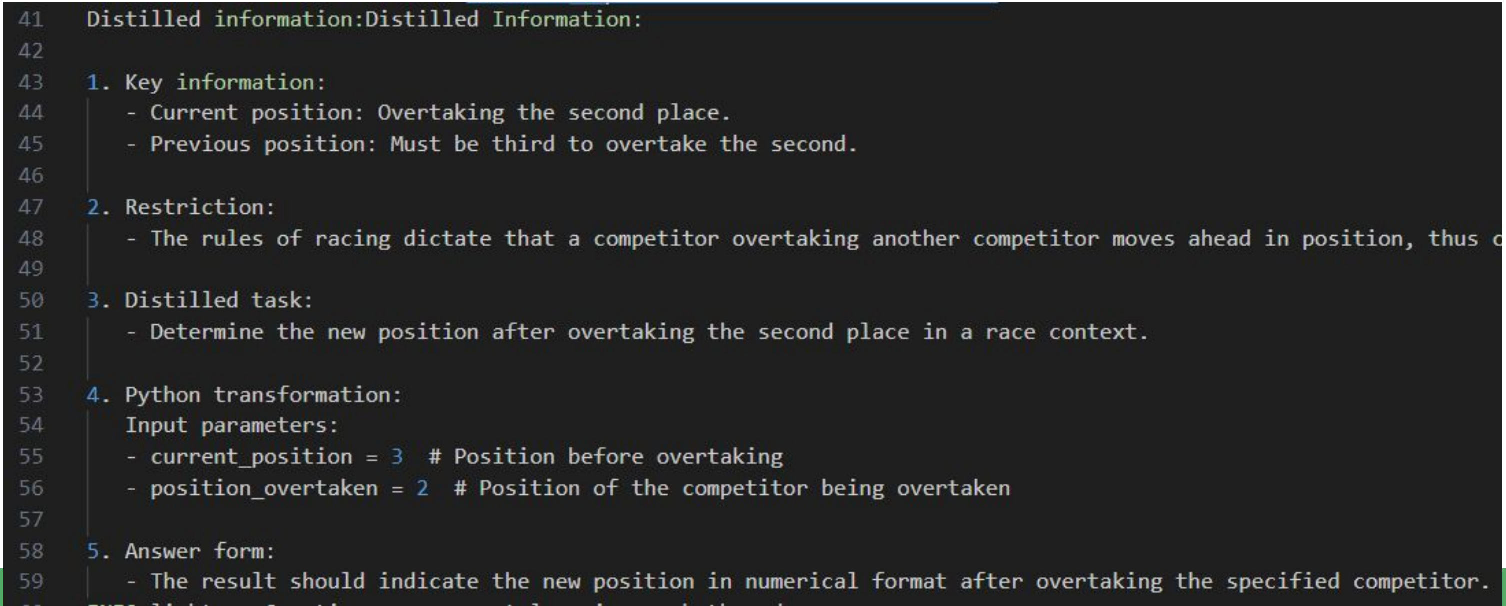
效率與穩健性的提升：

- 單次模板檢索 + 思維增強推理即能完成任務，計算成本僅為多查詢方法的 12%。
- 動態模板更新策略減少冗餘，保持 Meta Buffer 輕量化與高效運行。

解決上下文窗口限制：通過模板化儲存高層次資訊，避免冗長提示對模型上下文窗口的限制。

Problem Distiller (提取關鍵信息)

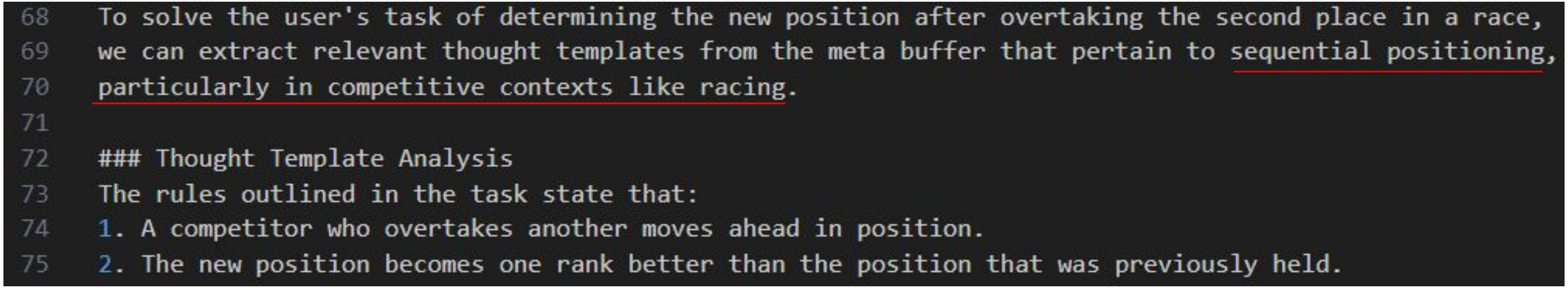
- 限制條件: 根據賽跑規則，超越某位參賽者後，位置將上升一名。
- 摘要後的問題 (Distilled Task): 根據競賽規則，確定超越第二名後的排名
- Python轉換、Answer Form(數字格式，輸出最終排名)
- 當前位置 (current_position): 第三名（才能追過第二名）。被超越者的位置 (position_overtaken): 第二名。



```
Distilled information:Distilled Information:

1. Key information:
   - Current position: Overtaking the second place.
   - Previous position: Must be third to overtake the second.

2. Restriction:
   - The rules of racing dictate that a competitor overtaking another competitor moves ahead in position, thus c

3. Distilled task:
   - Determine the new position after overtaking the second place in a race context.

4. Python transformation:
   Input parameters:
   - current_position = 3  # Position before overtaking
   - position_overtaken = 2  # Position of the competitor being overtaken

5. Answer form:
   - The result should indicate the new position in numerical format after overtaking the specified competitor.
```

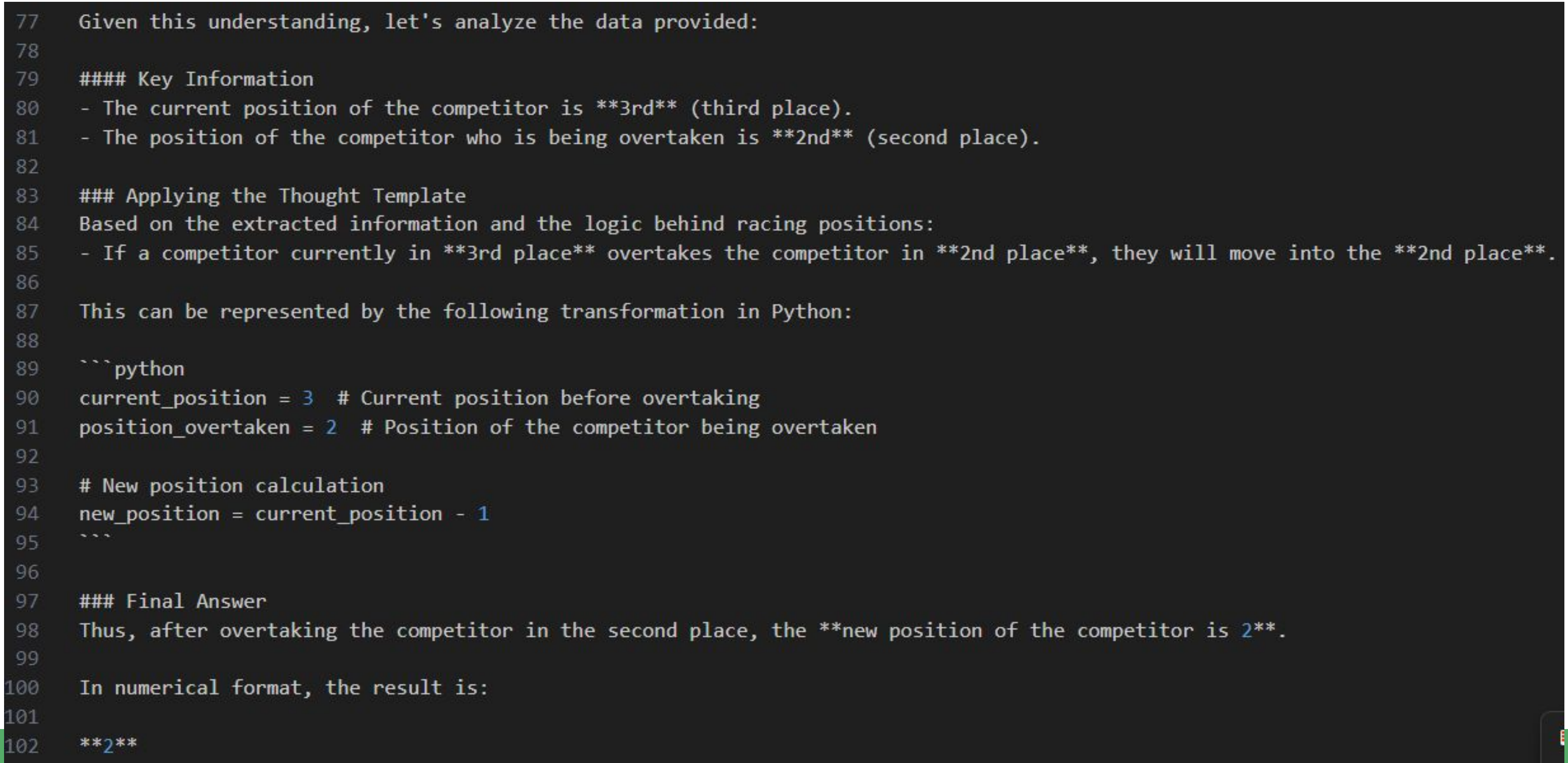
思維模板匹配與選取 (Thought Template Selection)

- 問題類型的識別: 問題被分類為「競賽位置調整問題」, 涉及序列位置的變化。
- 思維模板分析
- 任務中概述的規則規定:
  - 超過另一名選手的選手將會取得領先。
  - 新位置比原位置高一等。



```
To solve the user's task of determining the new position after overtaking the second place in a race,
we can extract relevant thought templates from the meta buffer that pertain to sequential positioning,
particularly in competitive contexts like racing.

### Thought Template Analysis
The rules outlined in the task state that:
1. A competitor who overtakes another moves ahead in position.
2. The new position becomes one rank better than the position that was previously held.
```

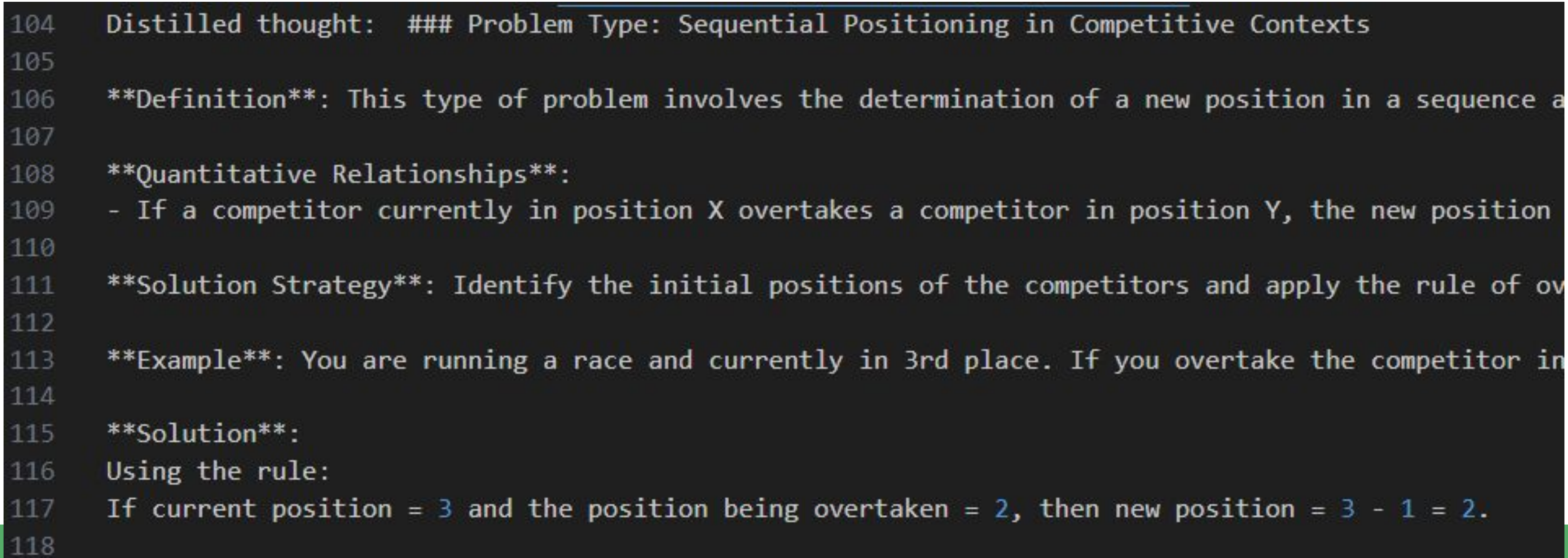
實例化推理 (Instantiated Reasoning):



~~~
Given this understanding, let's analyze the data provided:

#### Key Information
- The current position of the competitor is **3rd** (third place).
- The position of the competitor who is being overtaken is **2nd** (second place).

### Applying the Thought Template
Based on the extracted information and the logic behind racing positions:
- If a competitor currently in **3rd place** overtakes the competitor in **2nd place**, they will move into the **2nd place**.

This can be represented by the following transformation in Python:

```python
current_position = 3  # Current position before overtaking
position_overtaken = 2  # Position of the competitor being overtaken

# New position calculation
new_position = current_position - 1
```


### Final Answer
Thus, after overtaking the competitor in the second place, the **new position of the competitor is 2**.

In numerical format, the result is:

**2**
~~~

Buffer Manager 執行「思維提煉 (Template Distillation)」

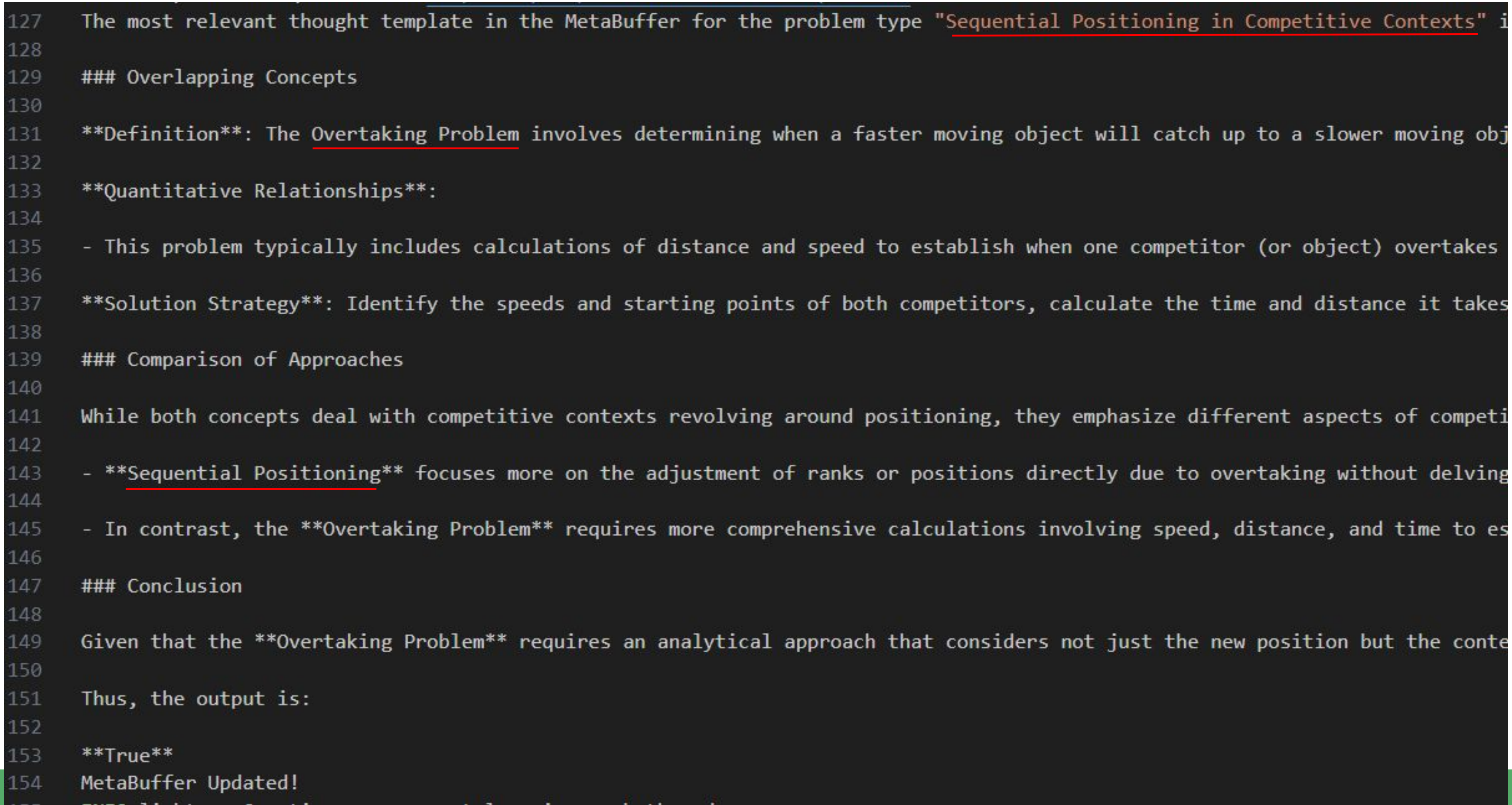
- 將該特定問題(競賽中的位置變化問題)轉化為通用模板，使得該模板可以泛化到類似的「序列位置調整問題」。
- 確保未來在處理類似問題時可以快速檢索並應用這一通用邏輯。



```
Distilled thought:  ### Problem Type: Sequential Positioning in Competitive Contexts

**Definition**: This type of problem involves the determination of a new position in a sequence a

**Quantitative Relationships**:
- If a competitor currently in position X overtakes a competitor in position Y, the new position

**Solution Strategy**: Identify the initial positions of the competitors and apply the rule of ov

**Example**: You are running a race and currently in 3rd place. If you overtake the competitor in

**Solution**:
Using the rule:
If current position = 3 and the position being overtaken = 2, then new position = 3 - 1 = 2.

```

Buffer Manager 執行 模板儲存過程 (Meta Buffer Update)

- 檢查當前問題與 Meta Buffer 中已有的模板是否具有高相似性，確定是否需要更新
- 如果是全新類型的問題，則提煉出新模板並存儲至 Meta Buffer。



```
The most relevant thought template in the MetaBuffer for the problem type "Sequential Positioning in Competitive Contexts" i

### Overlapping Concepts

**Definition**: The Overtaking Problem involves determining when a faster moving object will catch up to a slower moving obj

**Quantitative Relationships**:

- This problem typically includes calculations of distance and speed to establish when one competitor (or object) overtakes

**Solution Strategy**: Identify the speeds and starting points of both competitors, calculate the time and distance it takes

### Comparison of Approaches

While both concepts deal with competitive contexts revolving around positioning, they emphasize different aspects of competi

- **Sequential Positioning** focuses more on the adjustment of ranks or positions directly due to overtaking without delving

- In contrast, the **Overtaking Problem** requires more comprehensive calculations involving speed, distance, and time to es

### Conclusion

Given that the **Overtaking Problem** requires an analytical approach that considers not just the new position but the conte

Thus, the output is:

**True**
MetaBuffer Updated!
```