CRMWeaver: Building Powerful Business Agent via Agentic RL and Shared Memories

Yilong Lai[1,2]*, **Yipin Yang**[1], **Jialong Wu**[2], **Zhenglin Wang**[2], **Ting Liang**[1], **Jianguo Lin**[1], **Keping Yang**[1]†
[1] Taobao & Tmall Group of Alibaba [2] Southeast University, China
laiyilong0@gmail.com, {uipin.yang, kuiyu.lt, tengxiao, shaoyao}@alibaba-inc.com

Abstract

Recent years have witnessed the rapid development of LLM-based agents, which shed light on using language agents to solve complex real-world problems. A prominent application lies in business agents, which interact with databases and internal knowledge bases via tool calls to fulfill diverse user requirements. However, this domain is characterized by intricate data relationships and a wide range of heterogeneous tasks, from statistical data queries to knowledge-based question-answering.

To address these challenges, we propose CRMWeaver, a novel approach that enhances business agents in such complex settings. To acclimate the agentic model to intricate business environments, we employ a synthesis data generation and RL-based paradigm during training, which significantly improves the model's ability to handle complex data and varied tasks. During inference, a shared memories mechanism is introduced, prompting the agent to learn from task guidelines in similar problems, thereby further boosting its effectiveness and generalization, especially in unseen scenarios. We validate the efficacy of our approach on the CRMArena-Pro dataset, where our lightweight model achieves competitive results in both B2B and B2C business scenarios, underscoring its practical value for real-world applications.

1 Introduction

With the advanced reasoning and decision-making capabilities, large language model (LLM)-based agents [1, 2], are poised to advance artificial intelligence toward more human-like intelligence by autonomously interacting with external environments to accomplish complex goals. A large number of tasks have proven to be effective using an agent-based framework, including coding [3, 4], web browsing [5, 6], science discovery [7, 8], and deep research [9, 10].

In business applications, agents are increasingly expected to interact with complex environments through multiple tool calls to address diverse user needs. However, deploying such agents in real-world scenarios remains highly challenging. Their effectiveness is often constrained by the intricate nature of domain-specific data dependencies (e.g., large-scale interconnected tables) as well as the wide variability of user demands (e.g., queries ranging from domain-specific QA to complex data analysis). As a result, only proficient but prohibitively costly models have been able to achieve satisfactory performance in these contexts.

*Work done during interned at Alibaba Group.
† Corresponding Author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Multi-Turn Interactions in Large Language Models.

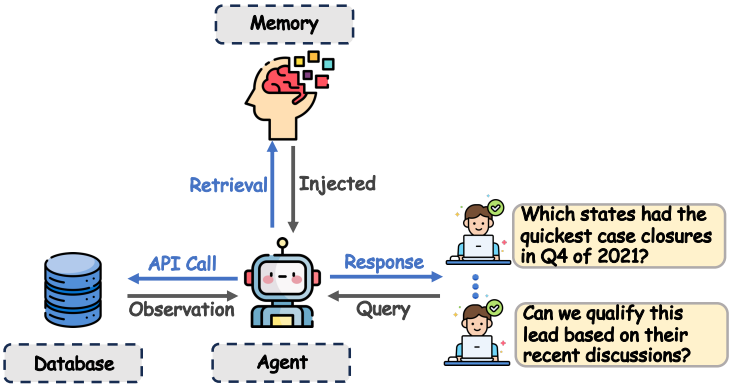

Figure 1: An overview of CRMWeaver.

To address these challenges, we propose **CRMWeaver**, a novel framework for building robust business agents through a two-fold optimization strategy. First, to adapt the model to the intricacies of business environments while mitigating the scarcity of high-quality training data, we employ large language models to synthesize complex and diverse training data. Based on the synthesis data, we introduce a two-stage training paradigm. In the first stage, we utilize rejection sampling and trajectory distillation to bootstrap the model, enabling it to solve the queries through multi-turn interaction. In the second stage, we employ reinforcement learning with Decoupled Clip and Dynamic Sampling Policy Optimization(DAPO) [11] to further enhance the models generalization in complex business environments[3]. Second, to cope with the diversity of user queries and the continual evolution of business schemas, we introduce a long-term memory module to improve the agents stability across multiple tasks, particularly for previously unseen tasks. Specifically, we construct an index of task-specific guidelines built from successful training trajectories. At inference time, when similar tasks are identified, relevant memories are retrieved and injected into the current context, enabling the agent to leverage prior knowledge and improve task performance. Figure 1 is an overview of CRMWeaver.

Our contributions are threefold:

1. We address the scarcity of high-quality training data and the need to adapt to complex business environments by combining synthetic data generation with a two-stage training pipeline.
2. We introduce a long-term memory module that improves robustness and generalization by enabling the agent to learn from solution guidelines produced by a stronger reasoning model on related problems, thereby enhancing the agents capabilities on diverse, complex tasks.
3. A lightweight model built on Qwen3-4B achieves performance comparable to Qwen3-235B-A22B-Instruct and Gemini 2.5-Pro on the business-agent benchmark CRMArena-Pro across B2B and B2C scenarios, demonstrating the effectiveness of CRMWeaver.

2 Preliminaries

To demonstrate the effectiveness of CRMWeaver, we choose to train and test on CRMArena [12] and CRMArena-Pro [13]. Both provide realistic sandbox environments for evaluating LLM agents in a business environment, with a particular emphasis on complex multi-table dependencies and diverse enterprise tasks. CRMArena models 16 interconnected Salesforce objects with latent variables that capture hidden causal relations across thousands of records, enabling nine expert-validated customer service tasks covering service managers, analysts, and agents. Building upon this foundation, CRMArena-Pro substantially expands both the structural and functional scope: it integrates 25 Salesforce objects across Service, Sales, and Configure-Price-Quote schemas, resulting in over

[3]To rigorously assess generalizability while mitigating data leakage, we conduct model training on CRMArena and evaluation on CRMArena-Pro, which encompass entirely distinct environments.

80,000 records across B2B and B2C organizations, and introduces nineteen tasks spanning four core business skills (database querying, textual reasoning, workflow execution, and policy compliance). These benchmarks collectively stress-test agents' abilities to navigate highly interconnected enterprise data, reason across both structured and unstructured records, and handle complex tasks through multi-turn interactions, thereby offering a uniquely challenging and diverse evaluation ground for advancing business-oriented LLM agents.

The task-solving agent in business environments follows the standard agent-environment interaction paradigm. At each time step t, the agent receives an observation $o_t \in \mathcal{O}$ from the business environment and selects an action $a_t \in \mathcal{A}$ based on the context $c_t = (o_1, a_1, \cdots, o_{t-1}, a_{t-1}, o_t)$, where $\mathcal{O}$ and $\mathcal{A}$ denote the observation and action spaces, respectively. This interaction process can be naturally formulated as a Markov Decision Process, characterized by the tuple $\langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R} \rangle$. Here, $\mathcal{U}$ denotes the space of initial user queries, $\mathcal{S}$ the latent state space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the state transition function, and $\mathcal{R}$ denotes the reward function, which indicates task completion. The action space $\mathcal{A}$ corresponds to the set of tools the agent uses to interact with the business environments. In this work, we define three actions: 1) ***Execute***, which employs SQL or Salesforce Object Search Language (SOSL) to retrieve relevant records or articles for problem solving. 2) ***Date Calculation***, which computes precise temporal information given the current date and an interval count. 3) ***Answer***, which allows the agent to finalize and submit a response once it has identified the solution.

The above setup defines the environments, tasks, and interaction paradigm that ground our work. We next introduce CRMWeaver, outlining its methodology for enhancing the problem-solving business-oriented tasks.

3 Methodology

In this section, we outline the methodology of CRMWeaver in four parts. We first introduce our data synthesis approach in § 3.1, designed to support cold-start training by synthesizing challenging query and QA tasks that foster the models ability to reason over complex and heterogeneous data. Then, we introduce the overall paradigm and setup of CRMWeaver in § 3.2. § 3.3 presents a two-stage training pipeline, consisting of supervised fine-tuning (SFT) for initialization, followed by a reinforcement learning phase with DAPO to enhance generalization in complex environments. Lastly, § 3.4 describes the integration of long-term memory at inference, which further strengthens the models robustness and adaptability across diverse tasks, particularly when addressing previously unseen scenarios.

3.1 Synthesis Data Construction

Given that CRMArena provides only a limited set of task-specific test queries targeting particular objects, we aim to enhance the model's generalization and reasoning capabilities over complex, multi-relational, and heterogeneous data. Inspired by recent work leveraging synthetic complex QA datasets to improve deep reasoning agents [9, 10, 14], we construct a graph G, where nodes correspond to records and edges represent inter-table relationships, thereby capturing the relational structure inherent in CRMArena. For each synthesis data sample $X = (Q, A)$, we first extract a path Ψ through a random walk on the graph G:

$$\Psi = (e_1, e_2, \cdots, e_k) \sim \mathrm{RandomWalk}(G) \tag{1}$$

where Q is the synthesis query, A is the corresponding answer, e_i is the sampled record in the path Ψ, and k is the length of the sample path Ψ.

For each pair $\{e_{i-1}, e_i\}$ along the path Ψ, the connection is established through a specific field both in the records e_{i-1} and e_i. Starting from the initial node e_1, we take the unique identifier of e_1 as the final answer A. We then define q_i as a synthetic sub-query, derived from the partial path $(e_1, e_2, \cdots, e_i)$. To construct these sub-queries, we employ a prompt-based LLM strategy, where the LLM generates a more complex query q_i given the preceding query q_{i-1} and the corresponding record pair $\{e_{i-1}, e_i\}$:

$$q_i = \mathrm{LLM}(\{e_{i-1}, e_i\} | q_{i-1}, A) \tag{2}$$

After k iterations, we obtain a high-quality synthetic data $X = (Q, A)$. In addition, to broaden the scope and diversity of training tasks, we further construct synthetic datasets targeting statistical analysis and knowledge-base question answering, by exploiting tabular statistical features and internal

knowledge resources. The detailed prompts used for constructing synthesis data are provided in the Appendix B.

Overall, our training corpus encompasses three categories:

- *Complex synthetic data*: crafted using the method in Eq. (1) and Eq. (2) to strengthen the models ability to search, integrate, and reason over multi-relational queries across interconnected tables. We provide some examples of complex synthesis queries in Appendix B.1.
- *Simple synthetic data*: designed to instill basic business QA and data analysis skills, serving as a foundation for business problem-solving. (*e.g.*How long after purchase can Shoes & Clothings customers claim store credit for an overcharged order?)
- *Task-specific data*: consisting of nine well-defined tasks from CRMArena. (*e.g.*Which states had the quickest case closures in Q4 of 2021?)

3.2 Agent Setup of CRMWeaver

We build CRMWeaver on top of the ReAct framework [2], which integrates reasoning and acting through the ThoughtActionObservation paradigm.

Unlike purely reactive agents that rely solely on observed states, or purely deliberative agents that operate only through internal reasoning, ReAct interleaves natural language reasoning traces with environment interactions. Formally, given a query Q, the ReAct agent addresses the task through iterative cycles conditioned on the action space $\mathcal{A}$ and observation space $\mathcal{S}$. At time step t, with historical trajectory $\mathcal{H}_t = (Q, \tau_1, \alpha_1, o_1, \cdots, \tau_{t-1}, \alpha_{t-1}, o_{t-1})$, the agent derives the current thought τ_t and action α_t according to the following equation:

$$(\tau_t, \alpha_t) = \pi_\theta(\tau_t, \alpha_t | \mathcal{H}_t) \tag{3}$$

where π_θ is the policy of the model θ, and $\alpha_t \in \mathcal{A}$. As mentioned in Section 2, the Action Space $\mathcal{A}$ comprises three tools: **execute**, **date calculation**, and **answer**, which enable the agent to interact with external business environments, perform precise temporal computations, and submit the final answers.

For practical supervised fine-tuning(SFT) and reinforcement learning described in the following section, for each time step t, we restrict the thought τ_t, action α_t and observation o_t must be enclosed in <think>...</think>, <tool_call>...</tool_call> and <tool_response>...</tool_response> tags, respectively, while the answer tool content is wrapped within <answer>...</answer> tags. We provide a detailed trajectories example of CRMWeaver in Appendix A.

3.3 Two-stage Training Recipe

3.3.1 SFT for Agentic Model Cold Starts

To equip the model with the fundamental capability to address business tasks and ensure it follows the Thought-Action-Observation paradigm during problem-solving, we first leverage a powerful language model, GPT-4.1. Using the ReAct framework, we apply Reject Sampling [15, 16] to the training data mentioned in Section 3.1, generating multiple candidate execution trajectories. Subsequently, we filter these trajectories based on two primary criteria:

- Correctness of results: To ensure the accuracy of the trajectory data, we first exclude any trajectories where the final answer does not match the gold answer.
- Correctness of process: A portion of the training data has "None" as the answer. To ensure the model does not arrive at the result through simple guesswork, we discard trajectories with a very short number of execution turns.

This process yields a high-quality training dataset, $\mathcal{D}_{\text{SFT}}$, which contains a large number of execution trajectories. Each sample in the dataset is a high-quality trajectory $\mathcal{H} = (Q, \tau_1, \alpha_1, o_1, \cdots, \tau_t, \alpha_t, o_t)$. Following this, we train the model θ using supervised fine-tuning (SFT). It is important to note that, consistent with prior work [17, 9], we mask the loss for the

observation part of the trajectories:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{\sum_{i=1}^{|\mathcal{H}|} \mathbb{I}\left[x_i \neq o\right]} \sum_{i=1}^{|\mathcal{H}|} \mathbb{I}\left[x_i \neq o\right] \cdot \log p_\theta\left(x_i \mid \mathcal{S}, x_{<i}; \theta\right) \tag{4}$$

where $\mathbb{I}(\cdot)$ is the indicator function, x_i is the i-th token of trajectory sample $\mathcal{H}$, p_θ is the output probability of model θ and $\mathcal{S}$ is the system description for solving business tasks.

3.3.2 Reinforcement Learning to Generalization

To further enhance the agentic model's capabilities, particularly its generalization in complex business environments, we employed a training strategy based on Reinforcement Learning, which allows the model to refine its policy through multi-turn interactions with an external environment.

In details, for each sample (Q, A) from our training dataset $\mathcal{D}$, we rollout a group of trajectories, denoted as $\{\mathcal{H}_i\}_{i=1}^{G}$. Subsequently, we optimize the policy of model θ through Decouple Clip and Dynamic Sampling Policy(DAPO) [11]:

$$\begin{aligned}
\mathcal{J}_{\text{RL}}(\theta) = \mathbb{E}_{(Q,A)\sim\mathcal{D},\{\mathcal{H}_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} & \\
\left[\frac{1}{\sum_{i=1}^{G}|\mathcal{H}_i|}\sum_{i=1}^{G}\sum_{t=1}^{|\mathcal{H}_i|}\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right)\right] & \\
\text{s.t.} \quad 0 < \left|\{\mathcal{H}_i \mid \textbf{is_equivalent}(A, \mathcal{H}_i)\}\right| < G &
\end{aligned} \tag{5}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(\mathcal{H}_{i,t} \mid Q, \mathcal{H}_{i,<t})}{\pi_{\theta_{old}}(\mathcal{H}_{i,t} \mid Q, \mathcal{H}_{i,<t})}, \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})} \tag{6}$$

We use a rule-based reward function R, defined as follows:

$$R(\hat{a}_i, a) = 0.1 \times \text{score}_{\text{format}} + 0.9 \times \text{score}_{\text{answer}} \tag{7}$$

where the $\text{score}_{\text{format}}$ is the format score, which is awarded to the model θ only when its output contains complete and valid pairs of <think>...</think>, <tool_call>...</tool_call> tags, and ended with a valid <answer>...</answer> tag. The $\text{format}_{\text{answer}}$ is the answer score. For data analysis tasks, this reward is given only if the final answer $\hat{a}_i$ exactly matches the gold answer a. For QA tasks, we use the F1 score as the answer reward.

3.4 Improving Reasoning with Long-term Memory

For business-oriented tasks in CRMArena and CRMArena-Pro, we observe that while the surface forms of many problems vary, they often share common underlying solution strategies(*e.g.*"Find the agent with the shortest handle time who managed more than one case in Winter 2021" and "In the past 3 months, which agent achieved the lowest average handle time while handling more than 2 cases?"). Moreover, the tasks in business settings are diverse and cover various tasks, often comprising novel, unseen combinations of tasks.

To improve model performance on such heterogeneous business tasks, particularly in generalizing to unseen task types, we draw inspiration from recent work on long-term memory [18, 19]. We propose augmenting the inference process of our agent with a memory module that stores and retrieves workflow guidelines from similar past tasks. Specifically, our method employs an indexed memory of workflow information to guide inference toward empirically proven solution patterns for analogous problems. For tasks that do not have a direct precedent in the indexed memories, we employ an offline refinement step. In this step, a more powerful reasoning model attempts to solve the query and generates a detailed corresponding workflow guideline. This guideline is then added to the memory index, thereby enhancing the model's capacity to handle future, novel task types. Algorithm 1 details the specific procedure for solving tasks with the long-term memory module.

Key: Can we qualify this lead based on their recent discussions?... **Value:** 1. Always use the execute tool to pull every relevant transcript and pricing/ knowledge record before judging any BANT factor, ensuring no critical data is missed. 2. After data retrieval, perform an explicit cost−vs−budget calculation inside the analysis step to avoid incorrect 'Budget' conclusions. 3. When Authority is not clearly confirmed in the transcript, trigger an additional execute search (e.g., manager titles, purchasing roles) before deciding it is fulfilled or unmet. 4. Parse Timeline details quantitatively (days, dates) rather than qualitatively; if uncertain, run another execute query for scheduling constraints to improve accuracy. 5. Maintain a consistent workflow order −> execute −> analyze −> respond to increase reliability and reproducibility across similar tasks.

Table 1: An example of Memory Unit $\mathcal{E}$, where the **Key** is the indexed query used to match similar queries, and **Value** is the corresponding guideline.

Algorithm 1: Procedural Steps of the Long-term Memory Module

Input: Query Q, Indexed Memories $\mathcal{M}$, Threshold Φ, System Prompt $\mathcal{P}$
Output: Execution trajectory S
function INFERENCEWITHMEMORIES(Q, $\mathcal{M}$)
  $\mathcal{E}, \phi \leftarrow$ `Top1-Retrieval`$(Q, \mathcal{M})$ ▷ Obtain guideline information and corresponding similarity.
  if $\phi \geq \Phi$ **then**
    $\mathcal{P} \leftarrow$ `Concat`$(\mathcal{P}, \mathcal{E})$ ▷ Append guideline in the system prompt.
  else
    `Update`$(Q, \mathcal{M})$ ▷ Offline update indexed memories using advanced reasoning model.
  $\mathcal{S} \leftarrow$ `Solve`$(Q \mid \mathcal{P}; \theta)$ ▷ Solve the query based on the agentic model θ.
return S
end function

It is noteworthy that during the update operation, we employ an advanced reasoning model twice. The final output's consistency determines whether the indexed memories $\mathcal{M}$ are updated. When storing information in memories, we use an LLM to convert specific trajectories into high-level guidelines, which then constitute the memory unit $\mathcal{E}$:

$$\mathcal{E} = \mathrm{LLM}(\mathcal{H}_{\text{adv}}, Q) \tag{8}$$

where the $\mathcal{H}_{\text{adv}}$ is the trajectories of advanced reasoning model. Table 1 shows an example of the memory unit $\mathcal{E}$, and we also provide the details for generating and indexing for the memory unit in Appendix C.

4 Experiments

4.1 Experiments Setup

Implementation Details The backbone for our model θ is `Qwen3-4B-Instruct-2507`. Separately, the retrieval model discussed in § 3.4 is `BGE-small-en-v1.5`. We used the llama-factory framework [20] for Supervised Fine-Tuning (SFT) on nearly 3k trajectories. After this, we proceeded with reinforcement learning, employing the verl[4] framework and 3.5k question-answer pairs. In our RL training, we set the rollout size $G = 16$, with ε_{low} at 0.2 and $\varepsilon_{\text{high}}$ at 0.28. Moreover, to support stable model training during RL, we employ the official, locally deployable SQLite database[5]

[4] `https://github.com/volcengine/verl`
[5] `https://github.com/SalesforceAIResearch/CRMArena/tree/main/local_data`

Model	B2B Workflow	Policy	Text	Database	Avg	B2C Workflow	Policy	Text	Database	Avg
OpenAI-o1	67.0	43.3	23.4	56.3	47.5	74.5	34.5	29.5	59.6	49.5
GPT-4o	26.0	27.5	22.1	31.3	26.7	29.0	32.5	22.2	33.1	29.2
GPT-4o-mini	19.5	33.8	12.6	19.3	21.3	13.5	25.6	11.7	23.8	18.6
Gemini-2.5-Pro	83.0	41.0	34.7	57.6	54.1	90.0	42.0	36.2	64.9	**58.3**
Gemini-2.5-Flash	67.5	33.5	25.1	41.6	41.9	80.0	31.0	26.3	49.3	46.7
Qwen3-235B-A22B-Instruct	90.0	37.7	30.1	49.8	51.9	87.5	42.0	33.5	50.0	53.3
Kimi-K2	92.0	42.5	34.5	48.0	54.3	93.0	41.8	36.2	52.2	55.8
CRMWeaver	90.0	27.3	32.1	73.0	**55.6**	90.5	31.3	33.8	72.8	<u>57.1</u>
w.o Shared Memory	90.5	27.5	27.8	72.0	<u>54.5</u>	89.5	27.5	32.2	72.0	55.3

Table 2: Experiments result on CRMArena-Pro across the B2B and B2C domains. The best average results among all backbones are **bolded**, and the second-best results are <u>underlined</u>.

for the SQL tool. In the case of the SOSL tool, we follow the original configuration, which relies on invoking the Salesforce API. The threshold Φ in Algorithm 1 is set to 0.7. During inference, we set the temperature to 0.2 and limited the maximum number of interaction turns to 20. For the subsequent main experiments, the results of the OpenAI and Gemini series models are taken directly from the original paper [13]. We have also included results from the Qwen3-235B-A22B-Instruct model, using the ReAct paradigm, for comparison. All experiments are conducted on 2 nodes equipped with 8 x Nvidia H20.

Datasets Details The benchmark is composed of nineteen distinct, expert-validated tasks categorized into four core business skills:

- **Database Querying & Numerical Computation**: This skill assesses an agent's ability to formulate structured queries and perform numerical calculations on the retrieved data.
- **Information Retrieval & Textual Reasoning**: This involves processing and reasoning over unstructured text from sources such as knowledge bases or call transcripts to extract insights.
- **Workflow Execution**: This evaluates the agent's capacity to follow pre-defined business processes and execute actions based on specific rules.
- **Policy Compliance**: This tests an agent's ability to verify whether business operations adhere to established company policies and regulations.

The dataset encompasses both Business-to-Business (B2B) and Business-to-Consumer (B2C) scenarios. We adopted the single-turn setting, *i.e.*the user provides the task at once, resulting in a total of 3.8k test data samples. Evaluation in CRMArena-Pro is multifaceted. For sub-tasks such as knowledge QA and sales insight mining, we employed the F1 score as the evaluation metric, following standard settings. For other tasks, we used exact match as the metric.

4.2 Main Experiments

As depicted in Table 2, we present a comprehensive evaluation of our proposed method, CRMWeaver, against a suite of robust baseline models. The performance assessment was conducted on the CRMArena-Pro dataset, encompassing both B2B and B2C domains.

Our model demonstrates superior performance, achieving an average score of 55.6 in the B2B domain and a highly competitive score of 57.1 in the B2C domain. Notably, CRMWeaver exhibits a pronounced advantage in database-related tasks, where it outperforms all baselines with scores of 73.0 (B2B) and 72.8 (B2C). These results underscore the model's enhanced capability in handling structured data analysis.

To further investigate the contribution of the shared memory component, we conducted an ablation study by removing this module from the agentic model, a variant denoted as "w.o. Shared Memory". The experimental results reveal a discernible degradation in performance, with the average scores declining to 54.5 in the B2B domain and 55.3 in the B2C domain. This outcome indicates that the shared memory module plays a crucial role in facilitating the model's overall performance. However, it is also important to note that even without the aid of shared memories, the agentic model still yielded competitive results, suggesting that the training-time optimization of CRMWeaver remains robust.

5 Analysis

5.1 Impact of Reinforcement Learning with Agents

To study the impact of reinforcement learning on a model's capacity, we compare a model trained with only supervised fine-tuning (SFT) to one trained with both SFT and reinforcement learning. The experimental results are shown in Figure 2, which demonstrate that the application of RL consistently enhances model performance and generalization across diverse tasks. Notably, the RL-based model demonstrated superior outcomes in both B2B and B2C scenarios, suggesting that RL facilitates a more robust understanding of underlying task dynamics.

The most substantial gains are observed in the Workflow tasks, with the model's performance increasing from 67.5% to 90.5% in the B2B domain and from 64.5% to 89.5% in the B2C domain. This improvement is attributed to the sequential nature of these tasks, as the RL approach effectively trained the model to follow a correct action sequence, thereby instilling greater confidence in its reasoning and reducing errors.

Conversely, a minor performance decrease was observed on Policy tasks, which was traced to the Solution Violation Identification sub-task. Further analysis revealed that the SFT model frequently outputs 'None' after only a few simple attempts, leading to quick, albeit unreasoned, correct predictions. In contrast, the RL-trained model pursued a more semantically grounded, step-by-step reasoning process. While this behavior demonstrates superior reasoning, it resulted in a lower score on a metric that inadvertently rewards such shortcuts. Potential solutions to these challenges could include synthesizing a more diverse set of similar tasks or designing more robust evaluation mechanisms that reward nuanced reasoning over quick, shortcut-based predictions.

Figure 2: Performance comparison with SFT and RL on CRMArena-Pro.

5.2 The Frequency of Updates to Long-Term Memory

Our analysis indicates that updates to the Long-term Memory module are relatively infrequent, whereas its indexed memories exhibit consistently high utilization across a broad range of tasks.

In the CRMArena-Pro B2B scenario, the average update frequency is close to 3%, indicating that only 3 out of 100 business tasks, on average, trigger an indexed memories update. We do observe that for a small subset of tasks, such as Knowledge QA, the low similarity among questions results in a low probability of hitting the Long-term Memory. This can lead to the issue of frequent updates. To mitigate this, we employ a few-shot prompting approach with an LLM to classify whether a given query constitutes a knowledge-based question. This mechanism effectively prevents memory updates for QA tasks, thereby avoiding an excessive update frequency.

6 Related Works

Business Agents & Benchmarks The evaluation of business agents has evolved from simplified, synthetic benchmarks to more realistic, high-fidelity platforms that simulate real-world professional environments. Initial efforts, such as Webshop [21], were constrained by their reliance on simplified webpages. A shift toward more realistic enterprise software was observed with WorkArena [22], which utilized platforms like ServiceNow to assess agents on knowledge work tasks. Similarly,

WorkBench [23] introduced an outcome-centric evaluation approach for tasks like email drafting and meeting scheduling within a controlled, sandboxed setting. Further developments include Taubench [24], which created a user-agent environment where both the agent and a simulated user could interact to complete tasks such as retail and flight booking. More recently, CRMArena [12] was introduced, a benchmark constructed on a realistic Salesforce CRM system. This dataset was carefully built using a data synthesis and expert verification methodology to generate complex, high-quality tasks that reflect real-world business scenarios. CRMArena includes 16 Salesforce objects and 9 subtasks, integrating textual information from various sources, including data records, knowledge bases, and transcripts. Building on this foundation, CRMArena-Pro [13] was subsequently proposed. This enhanced version expands the number of Salesforce objects to 25 and includes 19 sub-tasks, which are further categorized into four distinct types: Workflow Execution, Policy Compliance, Database Querying, and Information Retrieval. Compared to previous benchmarks, both CRMArena and CRMArena-Pro offer a more comprehensive and realistic range of task scenarios, providing a closer approximation to real-world business tasks.

Long-term Memory of Language Agent The integration of long-term memory has emerged as a critical approach to enhancing the performance of agents by enabling them to learn from past tasks and historical trajectories within specific domains [25]. Several recent studies have explored various aspects of this concept. Inspired by the Ebbinghaus Forgetting Curve, MemoryBank [26] introduced a long-term memory mechanism that enables large language models (LLMs) to selectively store, retrieve, and update information, fostering more human-like interactions and personalized outputs. Similarly, MemoryLLM [27] advanced this area by representing knowledge in a latent space, which has been shown to improve model performance on knowledge-editing tasks. Later research has focused on leveraging different knowledge structures. For instance, Agent Workflow Memory(AWM) [28] enhances model capabilities through the automatic induction and reuse of sub-workflows, while KnowAgent improves performance by augmenting prompts with action-knowledge bases. A notable recent approach is Agent KB [18], which addresses the challenge of knowledge transfer across diverse tasks and frameworks by proposing a shared knowledge base. This work leverages a unique "teacher-student" dual-phase retrieval mechanism: a "student" agent first retrieves high-level workflow patterns for strategic guidance, while a "teacher" agent subsequently identifies execution-level patterns for detailed refinement. Furthermore, Mem0 [29] addresses the limitations of fixed context windows in LLMs by dynamically extracting and leveraging graph-based memory representations to capture complex relational structures.

Agentic Reinforcement Learning Agentic RL aims to enhance model capabilities through multi-turn interactions within complex, real-world environments. With the advent of Deepseek-R1 [30], rule-based RL has emerged as a dominant paradigm. A representative example is Search-R1 [31], which leverages end-to-end optimization of a large language model to improve its performance in multi-turn Retrieval-Augmented Generation (RAG) systems. Subsequently, ReTool [32] adopted a "Think-Code-Observation" approach, utilizing Python program code to enable multi-turn interactions with external code runtimes, thereby enhancing the model's mathematical reasoning abilities. The efficacy of Agentic RL is further demonstrated by landmark works in the field, such as WebDancer [9], Websailor [10], and WebShaper [33]. These projects utilize data synthesis and end-to-end agentic RL methods to build powerful deep research agents. This is accomplished through multi-turn interactions with web pages and search engines, thereby confirming the effectiveness of Agentic RL techniques.

7 Conclusion

We presented CRMWeaver, a framework for building powerful business agents capable of operating in complex business environments and handling diverse domain-specified tasks.

CRMWeaver addresses two core challenges: data scarcity and domain complexity, through a two-fold optimization strategy. First, we synthesize complex, diverse training data and adopt a two-stage learning paradigm: (i) rejection sampling and trajectories distillation to bootstrap problem-solving competency, followed by (ii) reinforcement learning based on DAPO to strengthen generalization and decision-making in diverse environments. Second, we introduce a long-term memory module that indexes task-specific guidelines distilled from successful trajectories; at inference time, retrieval-augmented conditioning enables the agent to reuse prior knowledge and maintain stability across unseen tasks and evolving schemas. Together, these components yield a practical recipe for deploying

business agents that reduce reliance on prohibitively expensive models while improving reliability in real-world, complex business settings.

References

[1] Lilian Weng. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023.

[2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[3] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[4] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

[6] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

[7] Yilun Zhao, Kaiyan Zhang, Tiansheng Hu, Sihong Wu, Ronan Le Bras, Taira Anderson, Jonathan Bragg, Joseph Chee Chang, Jesse Dodge, Matt Latzke, et al. Sciarena: An open evaluation platform for foundation models in scientific literature tasks. *arXiv preprint arXiv:2507.01001*, 2025.

[8] Yanzheng Xiang, Hanqi Yan, Shuyin Ouyang, Lin Gui, and Yulan He. Scireplicate-bench: Benchmarking llms in agent-driven algorithmic reproduction from research papers. *arXiv preprint arXiv:2504.00255*, 2025.

[9] Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, et al. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025.

[10] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.

[11] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[12] Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the capacity of LLM agents to perform professional CRM tasks in realistic environments. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3830–3850, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[13] Kung-Hsiang Huang, Akshara Prabhakar, Onkar Thorat, Divyansh Agarwal, Prafulla Kumar Choubey, Yixin Mao, Silvio Savarese, Caiming Xiong, and Chien-Sheng Wu. Crmarena-pro: Holistic assessment of llm agents across diverse business scenarios and interactions. *arXiv preprint arXiv:2505.18878*, 2025.

[14] Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, et al. Taskcraft: Automated generation of agentic tasks. *arXiv preprint arXiv:2506.10055*, 2025.

[15] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023.

[16] Aaron Grattafiori et.al. The llama 3 herd of models, 2024.

[17] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9354–9366, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[18] Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

[19] Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

[20] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[21] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[22] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: how capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, pages 11642–11662, 2024.

[23] Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. Workbench: a benchmark dataset for agents in a realistic workplace setting. In *First Conference on Language Modeling*.

[24] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024.

[25] Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, and Junxiao Shen. Human-inspired perspectives: A survey on ai long-term memory. *arXiv preprint arXiv:2411.00489*, 2024.

[26] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.

[27] Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. In *Forty-first International Conference on Machine Learning*.

[28] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*.

[29] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

[30] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[31] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[32] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

[33] Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, et al. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025.

A Case Study of CRMWeaver

Trajectories Example on CRMArena-Pro

Question:

Which agent ought to be designated to handle this lead? Return only the ID of the agent.
Lead Routing Policy
The lead routing policy determines the best agent to assign the given new lead based on the following criteria:
– Territory Match: First, find the territory that matches the lead's region by comparing it with the territory description (comma–separated list of states/regions).
– Quote Success: Among agents associated with the matching territory, select those with the highest number of quotes with status "Accepted" (find the agent ID through the associated opportunity ID).
– Workload Balance: If there's still a tie, select the agent with the fewest unconverted (open) leads
Lead's region: WI

Step 1

<think>
I need to find the territory that matches the lead's region (WI) by checking the Territory2 object. I will look for a territory with "WI" in its Description field.
</think>

<tool_call>
```
{"name": "execute", "arguments": {"query": "SELECT Id, Name FROM Territory2 WHERE Description LIKE '%WI%'"}}
```
</tool_call>

<tool_response>
```
Observation: [('0MIWt0000007xzROAQ', 'US-Great Lakes')]
```
</tool_response>

Step 2

<think>
I have found the territory that matches the lead's region (WI), which is "US-Great Lakes" with ID `"0MIWt0000007xzROAQ"`. Now, I need to find the agents associated with this territory by checking the UserTerritory2Association object.
</think>

<tool_call>
```
{"name": "execute", "arguments": {"query": "SELECT UserId FROM UserTerritory2Association WHERE Territory2Id = '0MIWt0000007xzROAQ'"}}
```
</tool_call>

<tool_response>
```
Observation: [('005Wt000003NHpdIAG',), ...]
```
</tool_response>

Step 3

<think>
I have found the list of agents associated with the territory "US-Great Lakes" (WI). Now, I need to determine which of these agents has the highest number of quotes with status "Accepted" by checking the Quote and QuoteLineItem objects.

</think>

<tool_call>
{"name": "execute", "arguments": {"query": "SELECT OwnerId, COUNT(*) AS accepted_quotes FROM Quote WHERE Status = 'Accepted' AND OwnerId IN ('005Wt000003NHpdIAG', ···, '005Wt000003NJmcIAG') GROUP BY OwnerId ORDER BY accepted_quotes DESC"}}
</tool_call>

<tool_response>
Observation: Error: no such column: OwnerId
</tool_response>

Step 4

<think>
The error occurred because the OwnerId field does not exist directly on the Quote object. Instead, the agent (OwnerId) is associated with the Opportunity object, and the Quote is linked to that Opportunity via OpportunityId. I need to check the Opportunity object to find the OwnerId (agent) associated with each Quote.
</think>

<tool_call>
{"name": "execute", "arguments": {"query": "SELECT o.OwnerId, COUNT(*) AS accepted_quotes FROM Quote q JOIN Opportunity o ON q.OpportunityId = o.Id WHERE q.Status = 'Accepted' AND o.OwnerId IN ('005Wt000003NHpdIAG', ···, '005Wt000003NJmcIAG') GROUP BY o.OwnerId ORDER BY accepted_quotes DESC"}}
</tool_call>

<tool_response>
Observation: [('005Wt000003NIowIAG', 5), ('005Wt000003NJmcIAG', 4),···, ('005Wt000003NDXaIAO', 1)]
</tool_response>

Step 5

<think>
The agent with ID '005Wt000003NIowIAG' has the highest number of accepted quotes (5), which is more than any other agent in the list. Since this agent has the highest quote success, they should be allocated to the lead according to the routing policy.
</think>

<answer>
005Wt000003NIowIAG
</answer>

Table 3: A Trajectories Example on CRMArena-Pro.

B Synthesis Data Construction

B.1 Complex Synthesis Data Examples

Here are some examples of complex synthesis data:

1. Which live chat transcript documents a case where a Portland-based customer, previously known for resolving product and sizing complaints in 2022, reached out on November 30, 2023, to request an immediate exchange after receiving a blue hoodie instead of black running shoes, and successfully initiated the return process with photo documentation through interactions involving three different support agents? Return only the chat transcript's ID. If no such records exist, return 'None'.

2. During the second week of January 2022, a customer with an account connected to a contact named Isabella Adams, who has an email address of isabella.morgan@example.com and a shipping address in Austin, Texas, placed an order that was activated and used a price book featuring the promotion 'Start 2022 with unbeatable deals on sports gear.' Which price book was referenced in this transaction? Return only the price book's id. If no such records exist, return 'None'.
3. Identify the durable and waterproof item, ideal for harsh weather conditions, that was added as a new selection by the customer whose previous order item was related to outdoor hiking, after a warranty issue with defective equipment was resolved in Portland and resulted in a successful transaction activation on March 19, 2022. What is the unique catalog reference for that item within the product listings? Return only the product's ID. If no such records exist, return 'None'.

B.2 Prompt for Complex Queries Generation

We provide the prompt used to generate the complex queries in Table 4 and Table 5.

C Long Term Memory Indexing

To generate the long-term memory unit $\mathcal{E}$, we use o3-mini as the advanced reasoning model. After obtain the trajectories $\mathcal{H}_{\text{adv}}$, we use GPT4.1 with prompt described in Table 6 to generate the guideline for the given task:

Prompt for Initialize Seed Query

You are a synthesis data generator. I will give you two schema and two corresponding data records. The two records are connected by a foreign key relationship. Your task is to generate a complex natural language question that includes the information from the records implicitly. Apart from this, the question's answer should be the id I provided with you. The question should be complex and not directly related to the id.

Input/Output Specifications:
Input:
– Source Schema: A description of the source schema.
– Source Record: A record from the source schema.
– Target Schema: A description of the target schema.
– Target Record: A record from the target schema.
– Connect Key: The connect relationship between the two records.
– Answer: Known correct answer.

Output:
– Must be in strict JSON format: {"Q": "generated question"}
– No explanations or extra fields allowed

Q must satisfy:
1. Be a complete natural language question
2. Allow deriving answer A by using the source and target records and connect key

Question Generation Principles:
1. Exact correspondence – Each question must fully base on the original conclusion, with the answer being its core content.
2. Derivability – The original conclusion must be directly derivable from the question and be the only correct answer.
3. Self–containment – Questions must be complete and independent, not relying on external references or unspecified context.
4. Information hiding – Do not reveal specific sources or data paths, but can include search hints.
5. Specificity and clarity – Questions should include details like specific times to ensure unique answers.
6. Single question – Generate only one question per conclusion.
7. If the conclusion can only be obtained from input content, include hints via data source identifiers in the question.
8. Language consistency – The language of each question must be the same as the conclusion's language.

Table 4: Prompt for Generating Seed Query

Prompt for Initialize Seed Query

You are a synthesis data generator. I will give you an existing question and its question. Apart from this, I will also give you a schema and a record. The schema and record includes a foreign key exactly the same as the answer of the giving question. Your task is to generate a more complex question that merges the information from the record and the current given record implicitly. Apart from this, the question's answer should be the id I provided with you. The question should be complex and not directly related to the ID. And the question should not include explicit information such as id of the record or the name of the user.

Input/Output Specifications:
Input:
– Existing Question: A complex natural language question that has been previously generated.
– Existing Question's Answer: The answer to the existing question.
– Target Schema: A description of the target schema.
– Target Record: A record from the target schema.
– Connect Key: The connect relationship between the two records.
– Answer: Known correct answer.

Output:
– Must be in strict JSON format: {"Q": "generated question"}
– No explanations or extra fields allowed

Q must satisfy:
1. Be a complete natural language question
2. Allow deriving answer A by using the source and target records and the connect key

Question Generation Principles:
1. Exact correspondence – Each question must be fully based on the original conclusion, with the answer being its core content.
2. Derivability – The original conclusion must be directly derivable from the question and be the only correct answer.
3. Self–containment – Questions must be complete and independent, not relying on external references or unspecified context.
4. Information hiding – Do not reveal specific sources or data paths, but can include search hints.
5. Specificity and clarity – Questions should include details like specific times to ensure unique answers.
6. Single question – Generate only one question per conclusion.
7. If the conclusion can only be obtained from input content, include hints via data source identifiers in the question.
8. Language consistency – The language of each question must be the same as the conclusion's language.

Table 5: Prompt for Generating Deeper Challenge Queries

Prompt for Generating Guideline

Please provide a concise guideline for the following execution log.
Break down the guideline into individual steps, and for each step, briefly describe what action was taken (note that it must be report the tool's name explicitly) and its outcome. Ensure the summary is clear, structured, and captures all key actions in the order they occurred.

Key Requirements
1. Focus exclusively on behavioral improvements derived from similar task patterns and experience.
2. Format output strictly as:
    1. [Specific suggestion 1]
    2. [Specific suggestion 2]
    ...
3. No headings, explanations, or markdown.

Detailed Schema Information
{schema_information}

Execution logs:
{successful_trajectories}

Table 6: The prompt used to generating the guideline of memory unit $\mathcal{E}$.