Conflict-Aware Memory for Embodied Agents: Enhancing Vector Data Quality via Detection Rules

Kexin Ma, Haotian Wang, Shenglin Chen, Yishuai Cai, Yu Huang, Ruochun Jin*
College of Computer Science and Technology,
National University of Defense Technology, Changsha, China
Correspondence: {makexin, jinrc}@nudt.edu.cn

Abstract

Embodied agents have successfully leveraged large language models (LLMs) to better transform human instructions and images into executable task plans. Furthermore, memories of agents can be leveraged to achieve continual self-learning and optimization. However, vector data quality problems emerge in memories when they are projected into vector space, especially in discerning contextually similar but semantically conflicting sentences and highly similar images. This is particularly detrimental to embodied AI as it potentially distorts the robot's actions. To address this challenge, we propose Conflict Detection Rules (CDRs) to identify and manage data quality issues in vector knowledge bases, which assist in correcting the index structure and further improving the answer quality. Experimental results show that planners with CDRs exceed the basic LLM planner by 15.25% and 14.25% in grammatical accuracy (GA) and interpretation accuracy (IA) on average, respectively. Moreover, the entire workflow has been successfully integrated into various scenarios, demonstrating its practical applicability and robustness in the real world[1].

1 Introduction

Recent advancements in artificial intelligence have led to the rapid evolution of agents, which are increasingly capable of performing complex, real-world tasks with minimal human intervention(Zhang et al., 2026b; Hu et al., 2026; Ma et al., 2026a; Lin et al., 2025; Ma et al., 2026b; Cheng et al., 2026). A prominent manifestation of this trend is the embodied agent, which extends cognitive capabilities into the physical realm. Embodied agents have successfully employed LLMs as planners to better follow human commands and accomplish corresponding tasks(Ahn et al., 2022; Song et al., 2022; Liu et al., 2023a; Lykov and Tsetserukou, 2023; Lykov et al., 2023; Chen et al., 2024; Vemprala et al., 2024; Fang et al., 2025). These models translate natural language into formal representations, such as the planning domain definition language (PDDL)(Liu et al., 2023a), first-order logic(Chen et al., 2024), or XML(Lykov and Tsetserukou, 2023; Lykov et al., 2023), etc., and leverage low-level controllers(Song et al., 2022) to execute the resulting subtasks, which demonstrates significant potential for automating complex reasoning and action planning(Huang et al., 2025). Meanwhile, recent embodied agents can explore environments without human-labeled data, store the encountered states and action outcomes in agent memories, and later retrieve the most informative episodes to generate the next response(Xu et al., 2024; Xie et al., 2024; Yoo et al., 2024; Zare et al., 2024). These varied, multimodal memories can be projected into a vector space for efficient storage and retrieval(Yamanaka and Kido, 2024; Zeng et al., 2026; Zhang et al., 2026c; Liu et al., 2025b).

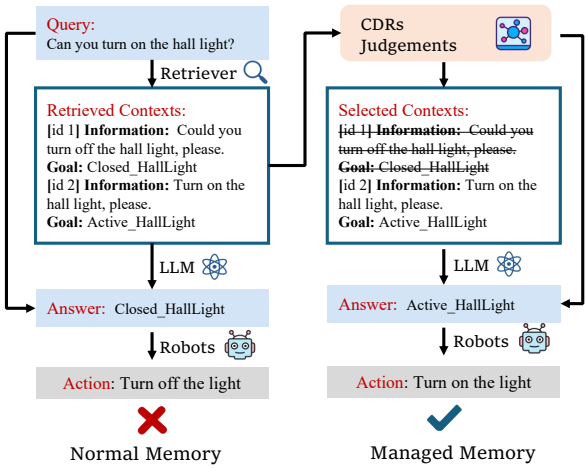

Figure 1: Example of potential error in memory-augmented-LLM planner. (a) represents the conventional memory-augmented-LLM planner, (b) illustrates the planner after enhancement of data quality.

*The corresponding author.
[1]Our code are available at https://github.com/makexine/CDRs

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28328–28347

However, the inherent limitations of vector databases(Liu et al., 2023b; Wang et al., 2023; Yang et al., 2026; Liu et al., 2025a) impedes its applications to the memory of embodied AI. The existing flaws in embedding generation and vector search methods can lead to incorrect similar vectors. For example, vectors of contextually similar but semantically opposing sentences are extremely close in vector space. Moreover, vector querying is predicated on approximate search algorithms for vectors, yet extant approximate search algorithms are incapable of addressing this issue(Ma et al., 2024). As shown in Figure 1, although the vector embeddings of the sentences "Can you turn on the hall light?" and "Can you turn off the hall light?" are closely located in the high-dimensional semantic space, the actions expressed by these sentences are conflicting. This results in poor data quality in the knowledge base of robots. Such conflicting sentences may disrupt the model's generation when fed into LLMs as backgrounds(Liu et al., 2023c), which is particularly detrimental to embodied AI as it potentially distorts the robot's actions. In fact, about 1.8 conflicting commands are found among the top 10 retrieved relevant commands on average for each question after constructing the knowledge base in our experiments.

In order to mitigate the impact of such data inconsistencies, we propose **Conflict Detection Rules (CDRs)** to detect the inconsistencies and manage the data quality in vector databases. CDRs are utilized to evaluate inconsistencies by aligning the subject, predicate, and other clausal components of sentences. This check are then propagated across multi-sentence passages, any inconsistent sentences are pruned to lift overall corpus quality. In general, we first evaluate the retrieved contexts leveraging CDRs after retrieval. If inconsistencies are detected, the corresponding contexts are removed and the indexes in the vector database are trimmed. Otherwise, the retrieved contexts will be reserved and returned directly. Experiments have shown that the planner with CDRs outperforms the basic planner on average by 15.25% in GA and 14.25% in IA, and also exceeds normal memory enhanced planner by 14.00% and 11.25% respectively. Moreover, we have successfully integrated the entire workflow into various scenarios such as **Café, Canteen, Kitchen** and **Lounge**, showing its practical applicability and robustness in real world.

Our work differs from previous work as follows. 1) We find the problem that occurrences of vector misalignment with original data have negative impacts on memory, which has been typically ignored by LLM planners. 2) CDRs are proposed to enhance the quality of vector data, which can be easily extended to suit special cases in other contexts, allowing it to integrate with and improve any LLM planners with high flexibility.

2 Related Work

LLMs as Planners. Researchers have employed LLMs as planners in robotic task planning (Vemprala et al., 2024; Singh et al., 2023; Huang et al., 2023; Zheng et al., 2022; Lu et al., 2022; Yamanaka and Kido, 2024), including 1) fine-tuning a LLM with numerous annotations (Lykov and Tsetserukou, 2023; Lykov et al., 2023) and 2) prompt engineering with few-shot examples, which either directly present a selection of fixed and typical examples(Ahn et al., 2022; Huang et al., 2022; Liu et al., 2023a; Chen et al., 2024) or combine a naive memory component similar to RAG (Retrieval Augmented Generation) without more attention on precise retrieval(Song et al., 2022; Yamanaka and Kido, 2024; Xu et al., 2024; Zhang et al., 2025; Jafaripour et al.; Zhang et al., 2026a). Different from previous work, we employ data quality management with LLM planners and update robot experiences into the knowledge base, thereby realizing a real-time and efficient augmentation on both small and large scale models.

Logical Rules for Data Quality Management. Logic-based rules such as data dependencies have been proposed for data quality management. Functional Dependencies (FDs)(Codd, 1971) have been pivotal in representing data integrity constraints and interconnections(Fan and Geerts, 2011; Raju and Majumdar, 1988; Chang et al., 2007; Abiteboul et al., 1994). Conditional Functional Dependencies (CFDs) were built upon FDs, specifically aimed at data cleaning (Fan et al., 2008). Further, Matching Dependencies (MDs)(Fan et al., 2011) are proposed to pinpoint records that correspond to identical real-world entities, and Approximate Functional Dependencies (AFDs) for addressing the challenges posed by noisy data(Karegar et al., 2021). However, they are tailored to relational data, presenting limitations regarding the vector data quality management essential for vector memory. Instead, our work attempts to utilized data dependencies to manage vector data, thereby optimizing the data quality of the robotic memory.

3 Conflict Detection Rules

This section introduces Conflict Detection Rules (CDRs). We first define the preliminaries, followed by the definition of CDRs. Additionally, we explain how to implement CDR predicates and use CDRs to detect conflicts in vector data . We start with basic notations and concepts.

Natural language actions. A natural language action $s[\phi]$ can be decomposed as a tuple $(s[o], s[p])$, where $s[o]$ denotes the object involved in $s[\phi]$, and $s[p]$ is the specific operation performed on $s[o]$.

Natural language sentences. A natural language sentence s is composed of several actions $s[\phi_i]$, which is denoted by $s = \{s[\phi_1], ..., s[\phi_n]\} = \{(s[o_1], s[p_1])..., (s[o_n], s[p_n])\}$.

Embedding vectors. Given a set of natural language sentences $S = \{s_1, s_2, \cdots, s_k\}$, the embedding vectors V of S is $V = f(S) = \{v_{s_1}, v_{s_2}, \cdots, v_{s_k}\}$, where $f(\cdot)$ is the embedding function. For any sentence $s_i \in S$, its corresponding embedding vector is $v_i \in V$.

3.1 CDRs with function predicates

We define the predicates of CDRs first.

Predicates. Over a natural sentence set $S = \{s_1, s_2, \cdots, s_k\}$, the corresponding vector set $V = f(S) = \{v_{s_1}, v_{s_2}, \cdots, v_{s_k}\}$, $\oplus$ is an operator in $\{=, \neq\}$, a vector similarity function VecSim$(\cdot)$, CDRs can be composed of the following predicates to capture the relationships between sentences and their vector embeddings.

$\underline{s_1[p_i] \oplus s_2[p_i]}$: Consider two sentences s_1 and s_2, predicate $s_1[p_i] \oplus s_2[p_i]$ compares actions of s_1 and s_2. The predicate is true if actions are the same. It is false otherwise.

$\underline{s_1[o_i] \oplus s_2[o_i]}$: Consider two sentences s_1 and s_2, predicate $s_1[o_i] \oplus s_2[o_i]$ compares objects of s_1 and s_2. The predicate is true if objects are the same. It is false otherwise.

$\underline{\text{VecSim}(v_{s_1}, v_{s_2})}$: Consider two embedding vectors v_{s_1} and v_{s_2}, we say that the vector similarity function VecSim(v_{s_1}, v_{s_2}) is true if VecSim$(v_{s_1}, v_{s_2}) \geq \theta$, where θ is a predefined threshold. Otherwise, it is false.

$\underline{\text{C}(s_1, s_2)}$: Consider two sentences s_1 and s_2. $\text{C}(s_1, s_2)$ denotes whether the embeddings generated from sentence generation are precise. We say that C(s_1, s_2) is true if conflicts exists between s_1 and s_2. Otherwise, it is false.

We write VecSim$(v_{s_1}, v_{s_2}) = true$ simply as VecSim(v_{s_1}, v_{s_2}) when it is clear in the context; similarly for C(s_1, s_2). The implementation details of these function predicates are shown in Section 3.2.

Conflict Detection Rules. For a natural sentence set S, the corresponding vector set $V = f(S)$, a CDR φ over S has the form of

$$X \rightarrow p_0,$$

where X is a conjunction of CDR predicates over S. In addition, p_0 is limited to C(s_1, s_2), which is aimed to align with the conflict detection task and further reduce the search space. We define the CDRs to identify conflicts between sentences and their embeddings. Consider the following example to illustrate the CDRs:

Example 1 *Consider the following CDRs.*
$\varphi_1 : s_1[p_i] \neq s_2[p_i] \wedge s_1[o_i] = s_2[o_i] \wedge$ *VecSim*$(v_{s_1}, v_{s_2}) \rightarrow C(s_1, s_2)$. *Given* s_1 *and* s_2 *as below,* s_1*: "Can you turn on the hall light?"* s_2*: "Can you turn off the hall light?"* φ_1 *claims that if the objects of sentences are the same, while their corresponding actions are not, and their corresponding vectors are similar, then they are in conflict. The vector embeddings* v_{s_1} *and* v_{s_2} *have high similarity, while* s_1 *and* s_2 *have the same objects and dissimilar actions. According to the CDR* φ_1*, we conclude that* $C(s_1, s_2)$ *is true.*
$\varphi_2 : s_1[p_i] = s_2[p_i] \wedge s_1[o_i] \neq s_2[o_i] \wedge$ *VecSim*$(v_{s_1}, v_{s_2}) \rightarrow C(s_1, s_2)$. *Given* s_1 *and* s_2 *as below,* s_1*: "Can you pass me an apple?"* s_2*: "Can you pass me a bottle?"* φ_2 *claims that if the actions of sentences are the same, while their corresponding objects are not, and their corresponding vectors are similar, then they are in conflict. The vector embeddings* v_{s_1} *and* v_{s_2} *have high similarity, while* s_1 *and* s_2 *have the same actions and dissimilar objects. According to the CDR* φ_1*, we conclude that* $C(s_1, s_2)$ *is true.*

Semantics of CDRs. Consider a sentence set S and its corresponding embedding vector set $V = f(S)$. A valuation h of sentence variables of CDR φ in S, or simply a valuation h of φ, is a mapping that instantiates $s[\phi]$ in each $s \in S$ with a sentence s in the sentence set S.

We write $h \models p$ for predicate p: (1) if p is $s_1[p_i] \oplus s_2[p_i]$ or $s_1[o_i] \oplus s_2[o_i]$, then $h \models p$ is interpreted as in tuple relational calculus following the standard semantics of first-order logic(Abiteboul et al., 1994). (2) If p is VecSim(v_{s_1}, v_{s_2}) (resp. C(s_1, s_2)), then $h \models p$ if VecSim (resp. C) predicates true on VecSim$(v_{h(s_1)}, v_{h(s_2)})$ (resp.

$\mathrm{C}(h(s_1), h(s_2)))$. For a conjunction X of predicates, we say $h \models X$ if for all predicates p in X, $h \models p$. For a CDR φ, we say $h \models \varphi$ that if $h \models X$, then $h \models p_0$. A sentence set S satisfies φ, denoted by $S \models \varphi$, if for all valuations h of sentence variables of s in S, $h \models \varphi$. We say $S \models \Sigma$ for the CDR set Σ if for all $\varphi \in \Sigma$, $S \models \varphi$.

3.2 Conflict Resolution with CDRs

We primarily elaborate on the concrete implementation of CDR predicates and the resolution of conflicts utilizing CDRs in this section.

Implementation of predicates. For $s_1[p_i] \oplus s_2[p_i]$ and $s_1[o_i] \oplus s_2[o_i]$, we use an LLM to determine their relationships. For predicate $\mathrm{VecSim}(v_{s_1}, v_{s_2})$, we have: $\mathrm{D}(v_{s_1}, v_{s_2}) > \theta \rightarrow \mathrm{VecSim}(v_{s_1}, v_{s_2})$, where $\mathrm{D}(v_{s_1}, v_{s_2})$ denotes the similarity score calculated by the similarity function in vector databases. Conventionally, a higher similarity score correlates with a greater degree of similarity. θ is a constant, which can be either manually defined or learned from samples. We obtain $\theta = 1.34$ through the ROC curve, more details can be seen in Appendix B.2.

Conflict resolution. As shown in Figure 2, for vectors that exhibit a conflict, we resolve the conflict by pruning the edges that connect the conflict vectors in the index structure. For multiple contextually similar sentences, we propose a Trimming Theorem as below. Given sentences s_k, s_1, s_2 which are contextually similar. Among them, s_k and s_1 are semantically similar, whereas those of s_k and s_2 are not. In other words, s_k and s_1 are consistent, s_k and s_2 are in conflict. In that case, there is a substantial likelihood of conflict between s_1 and s_2. Under such circumstances, we cut the edge between v_{s_1} and v_{s_2}. In other words, if the conflict detection result of s_k, s_1 and s_k, s_2 differs, then there exists a contradiction in s_1 and s_2, and s_k witness this, then the similarity linkage between s_1 and s_2 should be cut. The specific process is shown in Figure 2.

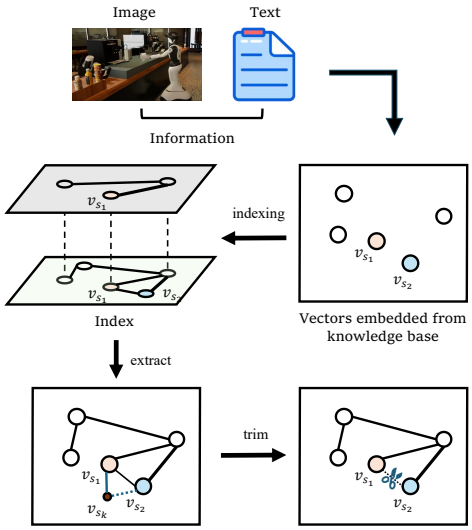

Figure 2: The process of conflict detection and resolution in vector databases.

4 Methodology

4.1 Problem Formalization and Overview

Formally, a knowledge base KB is created to store the multimodal memory of robots M. G denotes the index structure of KB, while $\bar{G}$ denotes the index after trimmed. Given a multimodal input q as the query, then $A = [a_1, ..., a_t]$, $C = [c_1, ..., c_k]$, $\bar{C} = [\bar{c_1}, ..., \bar{c_l}]$ denote fixed examples, related contexts and contexts after selection, respectively. y is expected to be the output finally as a logical formal language. The framework is shown in Figure 3.

Overview. The whole process is shown as below. First, the multimodal memory M in the knowledge base KB is embedded and an index structure G is constructed(①). Upon receiving q from users, relevant contexts C are retrieved from the knowledge base(step ②). These contexts are then sent to detect conflicts by CDRs (step ③), and judgments are utilized to trim the index structure of knowledge base G (step ④). Consistent contexts $\bar{C}$ are reserved and combined with q and other fixed examples A, feeding into LLM to produce a response y (step ⑤). Once the robot has executed successfully, we add this experience to KB (step ⑥).

4.2 The Framework

Knowledge Preprocess. The multimodal knowledge M in knowledge base KB is embedded into a unified vector space and the index G is constructed based on vector approximate search methods. M encompasses information from two modalities: images Ψ and text T, i.e., $M = \{\Psi, T\}$. For each ψ in Ψ, we utilize a multimodal LLM (MMLLM) to summarize the main content of the image ψ. The summary of image ψ is denoted as t_ψ. Memory after preprocessed is denoted as $\bar{M}$, $\bar{M} = \{T_\psi, T\}$, where $t_\psi \in T_\psi$. Knowledge in KB is documented in pairs of [Information-Goal], with each context recording a successful instance of natural language command transformation. The indexing structure G is constructed based on $\bar{M}$.

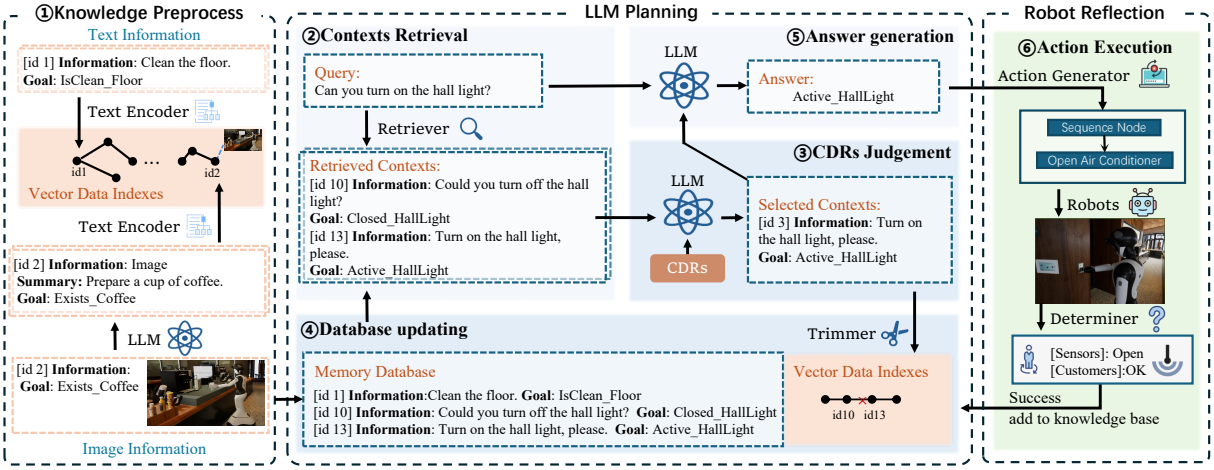

Figure 3: Framework of CDRs LLM planner

Contexts Retrieval. A retriever R is utilized to retrieve top k related contexts from $\bar{M}$ with q. The retriever R then swiftly locates the k contexts most similar to q by leveraging the established index structure G. Retrieved contexts $C = [c_1, ..., c_k]$ will then be sent to detect conflicts.

CDRs Judgement. CDRs and large parameter LLMs are utilized to detec conflicts in retrieved contexts C. We employ GPT-4o as an extractor and comparator for matching, and design a prompt for functioning. First, GPT-4o will extract and compare the actions (o, p) of q and data in C. Further, it will detect the conflict between q and $c_1, ..., c_k$ in C based on CDRs. We explain the meanings of CDRs via natural language to enable the LLM to comprehend rules and request the LLM to extract and compare the action components (o, p) based on CDRs and detect the conflicts. Finally, the reserved contexts $\bar{C}$ will be sent to LLM to generate answers, and the judgment will be passed to a trimmer for data index updating.

Database Updating. We construct the vector graph index G using the Hierarchical Navigable Small World (HNSW)(Malkov and Yashunin, 2016). Further, employing the conflicts judgment between q and set C, we assess the conflict of contexts within C and leverage this assessment to prune index G. First, a graph index G for the vector data stored in vector database is constructed based on the HNSW algorithm to facilitate subsequent search and other operations. IndexHNSWFlat in Faiss(Douze et al., 2024) is utilized for constructing, adopting NSW algorithm(Malkov et al., 2014) and decomposing it into multiple layers. After conflict detection, a trimmer is utilized for conflict resolution between contexts in C. Finally, we obtained a trained index graph $\bar{G}$ after multi-round trimming. The process is shown in Algorithm 1.

Algorithm 1: Index Trimming

Input: Query

for *q in Query* **do**
  $\bar{C}, C \leftarrow$new `// Contexts.`
  add **Retrieve**(q) to C;
  for *c in C* **do**
    if $result \leftarrow$***Judge***(c, q) **then**
      add c to $\bar{C}$; `// Record consistent contexts.`
    else
      Trim$(c, \bar{C})$; `// Record conflict contexts.`
for c_1, c_2 *in* C **do**
  if ***Trim***(c_1, c_2) **then**
    Cutoff(c_1, c_2); `// Cut off the link for` c_1 `and` c_2.

Answer Generation. Selected information is fed into LLM to generate the answer y. The input consists of three parts: query q, fixed examples A, and reserved contexts $\bar{C}$, which are structured orderly as the prompts. Details can be found in Appendix. Finally, LLM generates the answer y, articulated in the form of logical language.

Action Execution. The subsequent specific action sequences are generated by the action generator and then relayed to the robot for execution. We follow the method in (Chen et al., 2024), utilizing LLM to generate the final state of robots and an action generator to deduce the action sequences required to achieve this final state. If the robot

successfully executes the command q, then the pair $q - y$ is recorded in the form of [Information-Goal] and added to the knowledge base KB.

5 EXPERIMENTS

5.1 Settings

Datasets. Our experiments are conducted in four service scenarios, Café, Canteen, Kitchen and Lounge. The four datasets are designed based on the Virtual Home(Puig et al., 2018) and Alfred(Shridhar et al., 2020) benchmarks, and tailored to closely align with the actual scenarios, following the method in (Chen et al., 2024). Each dataset contains 200 data entries. The specific construction method can be found in the Appendix B.1. Each data entry is composed of an [Information-Goal] pair. Among all the data, 50% consists of image information, with the remainder being purely textual information. Goal y consists of literals connected by $\wedge$ or $\vee$, and are first generated by GPT-4o and subsequently refined through human judgment. Notably, our datasets target constrained service environments, where the number of objects and actions is finite. Meanwhile, our method is rule-based and non-parametric, which does not learn from data. In that case, 200 samples per scenario are sufficient to cover the requirement.

Metrics. We use grammatical accuracy (GA) and interpretation accuracy (IA), to evaluate the performance of our CDRs LLM-planner in the interpretation of goal states, referring to the approach outlined in (Chen et al., 2024). GA refers to the percentage of output goal states that are grammatically correct and IA refers to the percentage of goals that match the ground truth, which enables us to precisely evaluate both the grammar accuracy and the content correctness of the generated answers.

Configurations. Contriever-MSMARCO(Izacard et al., 2021) is employed as the retriever and Faiss(Douze et al., 2024) as the utilized vector database. By default, we set t=1, k=5, which means one fixed example is employed as the demonstration and five contexts related to q are retrieved first. In the selection of fixed examples, we employed a method of random sampling from the knowledge base KB. To test the performance of agent memory, we conducted a unified test after the robot had undergone 100 rounds of autonomous exploration and learning. The experimental results are presented as the average of 10 repeated trials, ensuring a robust and reliable assessment.

5.2 Baselines.

Language models. CDRs LLM-planner is evaluated in both pure text and mixed image-text scenarios. Meanwhile, various baseline language models serving as the answer generator, including large-parameter LLMs such as GPT-3.5-turbo, GPT-4o, Claude 3.5 Sonnet, and smaller LLMs such as Llama2-7b(Touvron et al., 2023), Llama3-8b(Dubey et al., 2024), bloomz-7b1(Muennighoff et al., 2022), falcon-7b(Almazrouei et al., 2023), gemma-2-9b-it(Team, 2024a) and Qwen2.5-7b-Instruct(Team, 2024b) in consideration of their convenience, accessibility, and versatility. Among them, bloomz-7b1 is the multitask-prompting fine-tuned version of the BLOOM(Le Scao et al., 2023). Multimodal LLMs includes large-parameter LLMs such as GPT-4o, Claude 3.5 Sonnet, and smaller LLMs such as BLIP-2(Li et al., 2023), LLaVA-Llama-3-8B(Contributors, 2023), MiniGPT-4(Llama2 version)(Zhu et al., 2023).

LLM planners. Our evaluation encompassed the assessments of three LLM planners: basic LLM planner as agent without memory, RAG or mRAG LLM planner as agent with simple memory, CDRs LLM planner as agent with managed memory, which generate responses by engaging with an LLM. The distinguishing feature among them is the specific prompts they employ: basic LLM planner uses fixed examples A, RAG LLM planner utilizes both A and retrieved contexts C, and CDRs planner employs A and contexts after trimming $\bar{C}$.

5.3 Main Results

Main results are shown in Table 1, and conclusions can be deduced as below.

CDRs LLM planner works with various language models. In the case of exclusively text, our work surpasses the basic LLM planner, the most significant increases of GA and IA both reach up to 45.00%. Meanwhile, on smaller scale LLMs, where the inherent memory and learning capabilities are comparatively weaker than those of larger-parameter LLMs, the gains achieved through our high-quality RAG method are even more pronounced.

Our work outperforms normal RAG LLM planner. CDRs LLM planner has on average boosted the GA and IA accuracy by 15.25%, 14.25% and 14.00%, 11.25% over basic LLM planner and normal mRAG LLM planner, respectively, which proves its effectiveness compared to regular re-

Language Model	LLM Planner	Café		Canteen		Kitchen		Lounge	
		GA	IA	GA	IA	GA	IA	GA	IA
Large-parameter LLM									
GPT-4o	base	85	83	72	70	79	73	80	78
	mRAG	86 (↑1)	70 (↓13)	73 (↑1)	73 (↑3)	84 (↑5)	74 (↑1)	83 (↑3)	79 (↑1)
	CDRs	100 (↑15)	89 (↑6)	92 (↑20)	89 (↑19)	92 (↑13)	84 (↑11)	95 (↑15)	83 (↑5)
Claude 3.5 Sonnet	base	89	79	78	73	89	80	84	79
	mRAG	80 (↓9)	76 (↓3)	83 (↑5)	78 (↑5)	85 (↓4)	82 (↑2)	85 (↑1)	82 (↑3)
	CDRs	97 (↑8)	87 (↑8)	95 (↑17)	83 (↑10)	100 (↑11)	92 (↑12)	93 (↑9)	89 (↑10)
Small-parameter LLM									
BLIP-2	base	16	13	28	25	26	23	18	12
	mRAG	35 (↑19)	28 (↑15)	46 (↑18)	43 (↑18)	42 (↑16)	37 (↑14)	38 (↑20)	24 (↑12)
	CDRs	50 (↑34)	47 (↑34)	73 (↑45)	58 (↑33)	68 (↑42)	53 (↑30)	58 (↑40)	48 (↑36)
LLaVA-Llama3-8b	base	63	24	58	43	62	42	63	45
	mRAG	58 (↓5)	36 (↑12)	52 (↓6)	42 (↓1)	59 (↓3)	46 (↑4)	63 (↑0)	56 (↑11)
	CDRs	80 (↑17)	69 (↑45)	83 (↑25)	54 (↑11)	79 (↑17)	56 (↑14)	80 (↑17)	68 (↑23)
MiniGPT-4	base	52	28	57	34	55	37	49	26
	mRAG	48 (↓4)	37 (↑9)	58 (↑1)	42 (↑8)	46 (↓9)	42 (↑5)	48 (↓1)	39 (↑13)
	CDRs	79 (↑27)	68 (↑40)	82 (↑25)	70 (↑36)	69 (↑14)	60 (↑23)	74 (↑25)	63 (↑37)

Table 1: Experiment results on different language models and methods. Values in parentheses respectively represent the performance enhancement of the CDRs planner over the mRAG planner and the performance improvement of the CDRs planner over the basic planner. **Basic, mRAG and CDRs** refer to the basic LLM planner, mRAG LLM planner, and CDRs LLM planner, respectively.

Figure 4: An use example of the robot executing tasks following the customer's instruction. In (a), the RAG LLM planner is used, and the robot incorrectly informed the guest, "The hall light is already off." which was a misinterpretation of the intended instruction. In (b), the CDRs LLM planner is used and the robot turns on the hall light correctly.

trieval. Moreover, the RAG planner exhibited unexpectedly poor performance than the basic planner on GPT-4o, which significantly demonstrates that the erroneous retrieved contexts due to low data quality mislead LLM generations.

5.4 Analysis

We conduct the ablation analysis on: 1) RAG component of our work; 2) The number t of fixed examples A; 3) RAG method; 4) CDRs and LLM

planner. More can be found in Appendix.

Impact of RAG component. We compared the results of the RAG component with retrieved contexts and the basic demonstration component with fixed examples. With both t and k set to 5, we conducted experiments on Café dataset, taking the average as the result. To eliminate the influence of other factors, we conducted separate tests using only A and only C, results are shown in Figure 6. It can be observed that simple retrieval augmentation performs mediocrely in embodied AI, and sometimes even falls short of directly employing fixed examples as demonstrations. Consequently, ensuring the data quality before retrieval is imperative.

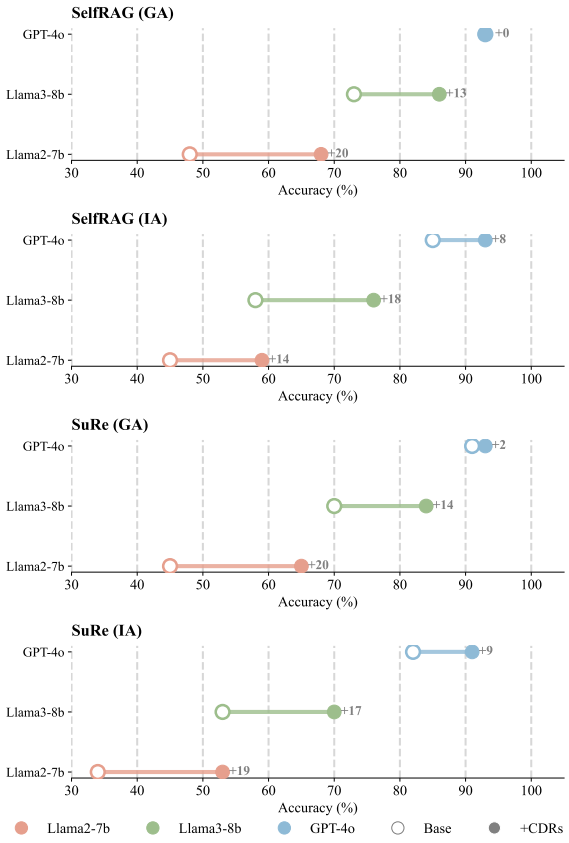

Figure 7: Comparison on SOTA methods.

Ablating fixed examples. t denotes the number of utilized fixed examples, to explore the impact of varying t on the overall model performance, we set t from 1 to 5 across the three planners and conducted repeated experiments to obtain the results, which are shown in Figure 5. More results can be found in Appendix B.6. The findings reveal that an increase in t enhances performance when the overall number of input documents is sparse. Conversely, when the total number is high, elevating t yields minimal performance gains and could potentially result in a degradation. This implies that an overabundance of contexts might engender conflicting information, thus misdirecting the LLM.

RAG method. We have included additional experiments comparing our method with three recent SOTA methods: SelfRAG(Asai et al., 2024) and SuRe(Kim, 2024) on Café_t dataset. The other settings are the same as that in Configurations. As shown in Figure 7, CDRs consistently improves accuracy across all models and frameworks, demonstrating its robustness and generalizability. Smaller models like Llama2-7b benefit significantly more (up to +20%) compared to stronger models like GPT-4o, which indicates CDRs effectively compensates for lower baseline capabilities.

LLM and CDRs planner. We have added a new baseline LLM Filter where a LLM(GPT-4o) is prompted to directly judge whether conflicts exist in retrieve passages. Index are trimmed based on the judgment of LLM Filter. As Table 2 shows, LLM filter has better performance than those planners without conflict detection, which confirms that conflict detection is necessary and critical. Meanwhile, it still performs worse than CDRs planner, demonstrating that explicit rule-based conflict resolution is more effective than simply prompting an LLM to directly judge conflicts.

Planner	GPT-4o		BLIP-2		MiniGPT-4	
	GA	IA	GA	IA	GA	IA
LLM Filter	92	84	42	35	57	46
CDRs	100	89	50	47	79	68

Table 2: Comparison of CDRs and LLM filter planners.

Robustness analysis. To better assess robustness and simulate the real world, we have conducted additional noise injection experiments on Café_t dataset and gpt-3.5-turbo model with other settings remaining unchanged. 10% irrelevant, 10% contradictory sentences and the mixture of the two situations are treated as data noise, injected into the test data, repectively, simulating the real-world situation. As results showed in Table 3, CDRs demonstrate good resilience, maintaining 92% GA under mixed noise (only an 8% drop) compared to RAG's significant collapse to 63% (-15%). This confirms that our explicit conflict detection effectively filters misleading information, preserving planning reliability even when the data is imperfect.

Use Example. We implemented CDRs LLM planner with OBTEA(Chen et al., 2024) to generate

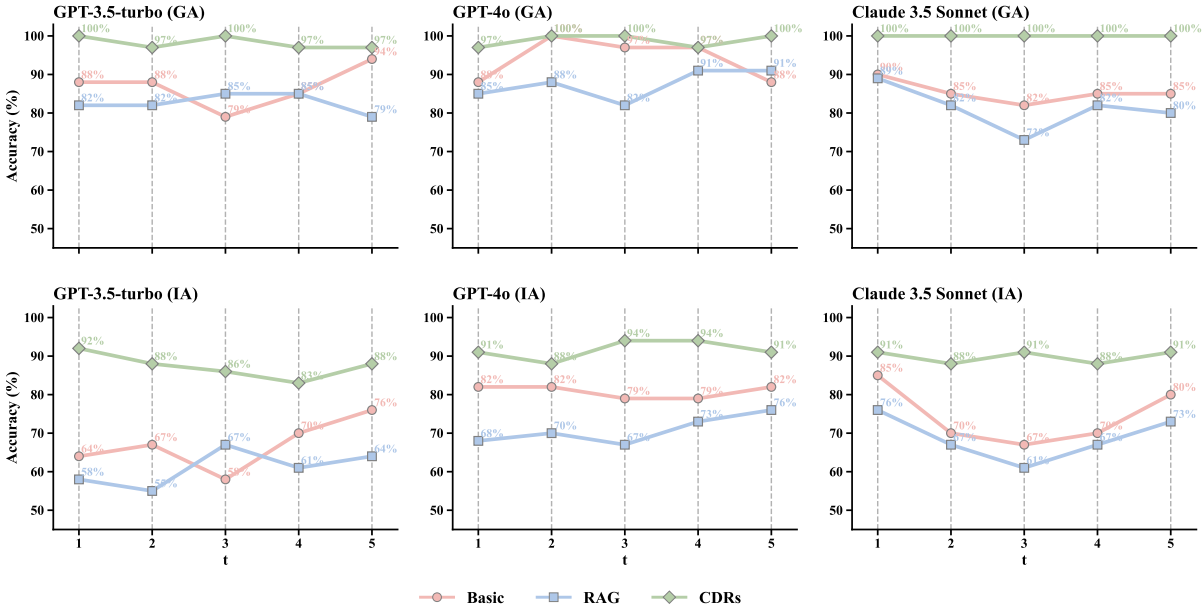

Figure 5: Fixed examples ablation on Larger LLMs.

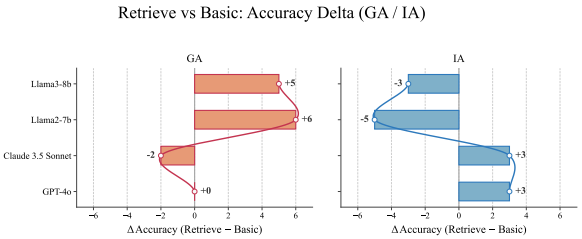

Figure 6: Comparison between fixed examples A and naive retrieved contexts C on Café.

Noise Condition	Method	GA (%)	IA (%)
Clean (0%)	RAG	78	71
	CDRs	**100**	**92**
10% Irrelevant	RAG	74 (↓ 4)	63 (↓ 8)
	CDRs	**98** (↓ 2)	**89** (↓ 3)
10% Contradictory	RAG	68 (↓ 10)	54 (↓ 17)
	CDRs	**95** (↓ 5)	**85** (↓ 7)
10% + 10% Mixed	RAG	63 (↓ 15)	48 (↓ 23)
	CDRs	**92** (↓ 8)	**60** (↓ 11)

Table 3: Performance Comparison with Delta Values

action sequences in a Café environment, achieving a comprehensive and reliable robot system as shown in Figure 4. The environment is constructed digitally using the simulator in MO-vln(Liang et al., 2023), where a humanoid robot serves as a waiter. The robot waiter conducts visual segmentation and detection to obtain information about objects from various perspectives.

6 Conclusions

CDRs significantly advance the capability of embodied AI to accurately interpret and execute natural language commands in a cross-modal manner. Its innovative integration of data quality management with RAG enhances both the reliability and accuracy of task planning. Meanwhile, the successful deployment of the CDRs LLM planner demonstrates the practicality and adaptability of our work, showing its potential for future applications.

Acknowledgment

This work is supported by NSFC No.62302503, Innovation Research Foundation of NUDT No.ZK23-15, the Open Research Fund from State Key Laboratory of High Performance Computing of China Grant No.202401-09, and Young Elite Scientists Sponsorship Program by CAST No.YESS20240767.

Limitations

Although CDR has achieved considerable improvements, it still falls short in overall accuracy for small LLMs. This may be due to the inferior reasoning capabilities of small-parameter models, which require higher data quality. Further refinement of logical rules may help address this issue. Additionally, the need to invoke LLMs for conflict detection during training incurs extra costs, which may increase latency in time-sensitive applications.

References

Serge Abiteboul, Richard Hull, and Victor Vianu. 1994. Foundations of databases.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario M Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, and 24 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Shi-Kuo Chang, Vincenzo Deufemia, Giuseppe Polese, and Mario Vacca. 2007. A normalization framework for multimedia databases. *IEEE Transactions on Knowledge and Data Engineering*, 19:1666–1679.

Xinglin Chen, Yishuai Cai, Yunxin Mao, Minglong Li, Wenjing Yang, Weixia Xu, and Ji Wang. 2024. Integrating intent understanding and optimal behavior planning for behavior tree generation from human instructions. *ArXiv*, abs/2405.07474.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. 2026. Mem^{2}evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation. *Preprint*, arXiv:2604.10923.

E. F. Codd. 1971. Further normalization of the data base relational model. *Research Report / RJ / IBM / San Jose, California*, RJ909.

XTuner Contributors. 2023. Xtuner: A toolkit for efficiently fine-tuning llm. `https://github.com/InternLM/xtuner`.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, and Shuai Ma. 2011. Dynamic constraints for record matching. *The VLDB Journal*, 20:495–520.

Wenfei Fan and Floris Geerts. 2011. Uniform dependency language for improving data quality. *IEEE Data Eng. Bull.*, 34:34–42.

Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33:6:1–6:48.

Zhen Fang, Zhuoyang Liu, Jiaming Liu, Hao Chen, Yu Zeng, Shiting Huang, Zehui Chen, Lin Chen, Shanghang Zhang, and Feng Zhao. 2025. Dualvla: Building a generalizable embodied agent via partial decoupling of reasoning and action. *arXiv preprint arXiv:2511.22134*.

Junan Hu, Shudan Guo, Wenqi Liu, Jianhua Yin, and Yinwei Wei. 2026. Context-agent: Dynamic discourse trees for non-linear dialogue. *arXiv preprint arXiv:2604.05552*.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. 2023. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR.

Yu Huang, Ziji Wu, Kexin Ma, and Ji Wang. 2025. Differentiable synthesis of behavior tree architectures and execution nodes. In *International Conference on Neuro-symbolic Systems, 28-30 May 2025, University of Pennsylvania, Philadelphia, Pennsylvania, USA*, Proceedings of Machine Learning Research, pages 231–259. PMLR.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Masoud Jafaripour, Shadan Golestan, Shotaro Miwa, Yoshihiro Mitsuka, and Osmar Zaiane. 2025. Adaptive iterative feedback prompting for obstacle-aware path planning via llms. In *AAAI 2025 Workshop LM4Plan*.

Masoud Jafaripour, Shadan Golestan, Shotaro Miwa, Yoshihiro Mitsuka, and Osmar R. Zaiane. Adaptive iterative feedback prompting for obstacle-aware path planning via llms.

Reza Karegar, Parke Godfrey, Lukasz Golab, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. 2021. Efficient discovery of approximate order dependencies. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 427–432. OpenProceedings.org.

Jaehyung Kim. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, and 1 others. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR.

Xiwen Liang, Liang Ma, Shanshan Guo, Jianhua Han, Hang Xu, Shikui Ma, and Xiaodan Liang. 2023. Mo-vln: A multi-task benchmark for open-set zero-shot vision-and-language navigation. *arXiv preprint arXiv:2306.10322*.

Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, and 1 others. 2025. Se-agent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. *arXiv preprint arXiv:2508.02085*.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Peiyang Liu, Ziqiang Cui, Di Liang, and Wei Ye. 2025a. Who stole your data? a method for detecting unauthorized rag theft. *arXiv preprint arXiv:2510.07728*.

Peiyang Liu, Xi Wang, Ziqiang Cui, and Wei Ye. 2025b. Queries are not alone: Clustering text embeddings for video search. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 874–883.

Peiyang Liu, Jinyu Yang, Lin Wang, Sen Wang, Yunlai Hao, and Huihui Bai. 2023c. Retrieval-based unsupervised noisy label detection on text data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4099–4104.

Yujie Lu, Weixi Feng, Wanrong Zhu, Wenda Xu, Xin Eric Wang, Miguel Eckstein, and William Yang Wang. 2022. Neuro-symbolic procedural planning with commonsense prompting. *arXiv preprint arXiv:2206.02928*.

Artem Lykov, Maria Dronova, Nikolay Naglov, Mikhail Litvinov, Sergei Satsevich, Artem Bazhenov, Vladimir Berman, Aleksei Shcherbak, and Dzmitry Tsetserukou. 2023. Llm-mars: Large language model for behavior tree generation and nlp-enhanced dialogue in multi-agent robot systems. *ArXiv*, abs/2312.09348.

Artem Lykov and Dzmitry Tsetserukou. 2023. Llm-brain: Ai-driven fast generation of robot behaviour tree based on large language model. *ArXiv*, abs/2305.19352.

Kexin Ma, Ruochun Jin, Xi Wang, Huan Chen, Jing Ren, and Yuhua Tang. 2024. Context-driven index trimming: A data quality perspective to enhancing precision of ralms.

Kexin Ma, Bojun Li, Yuhua Tang, Liting Sun, and Ruochun Jin. 2026a. Cast: Character-and-scene episodic memory for agents. *Preprint*, arXiv:2602.06051.

Xiaoxu Ma, Xiangbo Zhang, and Zhenyu Weng. 2026b. Stable and explainable personality trait evaluation in large language models with internal activations. *Preprint*, arXiv:2601.09833.

Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68.

Yury Malkov and Dmitry A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:824–836.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, and 1 others. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502.

K. Raju and Arun Kumar Majumdar. 1988. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Trans. Database Syst.*, 13:129–166.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE.

Chan Hee Song, Jiaman Wu, Clay Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2986–2997.

Gemma Team. 2024a. Gemma.

Qwen Team. 2024b. Qwen2.5: A party of foundation models.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Sai H Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2024. Chatgpt for robotics: Design principles and model abilities. *IEEE Access*.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-knowledge guided retrieval augmentation for large language models. *ArXiv*, abs/2310.05002.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. 2024. Embodied-rag: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*.

Weiye Xu, Min Wang, Wengang Zhou, and Houqiang Li. 2024. P-rag: Progressive retrieval augmented generation for planning on embodied everyday task. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6969–6978.

Jin Yamanaka and Takashi Kido. 2024. Evaluating large language models with rag capability: A perspective from robot behavior planning and execution. In *Proceedings of the AAAI Symposium Series*, volume 3, pages 452–456.

Qianyun Yang, Zhiwei Chen, Yupeng Hu, Zixu Li, Zhiheng Fu, and Liqiang Nie. 2026. Stable: Efficient hybrid nearest neighbor search via magnitude-uniformity and cardinality-robustness. *arXiv preprint arXiv:2604.01617*.

Minjong Yoo, Jinwoo Jang, Wei-Jin Park, and Honguk Woo. 2024. Exploratory retrieval-augmented planning for continual embodied instruction following. *Advances in Neural Information Processing Systems*, 37:67034–67060.

Ali Zare, Yulei Niu, Hammad Ayyubi, and Shih-fu Chang. 2024. Rap: Retrieval-augmented planner for adaptive procedure planning in instructional videos. In *European Conference on Computer Vision*, pages 410–426. Springer.

Yu Zeng, Wenxuan Huang, Zhen Fang, Shuang Chen, Yufan Shen, Yishuo Cai, Xiaoman Wang, Zhenfei Yin, Lin Chen, Zehui Chen, and 1 others. 2026. Vision-deepresearch benchmark: Rethinking visual and textual search for multimodal large language models. *arXiv preprint arXiv:2602.02185*.

Qianchi Zhang, Hainan Zhang, Liang Pang, Yongxin Tong, Hongwei Zheng, and Zhiming Zheng. 2025. Less is more: Compact clue selection for efficient retrieval-augmented generation reasoning. *arXiv preprint arXiv:2502.11811*.

Qianchi Zhang, Hainan Zhang, Liang Pang, Hongwei Zheng, and Zhiming Zheng. 2026a. Stable-rag: Mitigating retrieval-permutation-induced hallucinations in retrieval-augmented generation. *arXiv preprint arXiv:2601.02993*.

Wenyuan Zhang, Xinghua Zhang, Haiyang Yu, Shuaiyi Nie, Bingli Wu, Juwei Yue, Tingwen Liu, and Yongbin Li. 2026b. Expseek: Self-triggered experience seeking for web agents. *Preprint*, arXiv:2601.08605.

Yuanjun Zhang, Fuzel Ahamed Shaik, Suvojit Acharjee, Fahad Khalid, and Mourad Oussalah. 2026c. Towards reliable multimodal disaster severity assessment through preference optimization and explainable vision-language reasoning. *Reliability Engineering & System Safety*, page 112674.

Kaizhi Zheng, Kaiwen Zhou, Jing Gu, Yue Fan, Jialu Wang, Zonglin Di, Xuehai He, and Xin Eric Wang. 2022. Jarvis: A neuro-symbolic commonsense reasoning framework for conversational embodied agents. *arXiv preprint arXiv:2208.13266*.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

A Appendix

B Experiment

B.1 Datasets Construction

Based on scenario characteristics and predicate features, objects are categorized as illustrated in Table 6. Specific details can be found in Table 7,8,9,10, respectively. Moreover, the four distinct scenarios datasets, including **Café**, **Canteen**, **Kitchen**, and **Lounge**, are constructed leveraging LLMs integrated with the classification results.

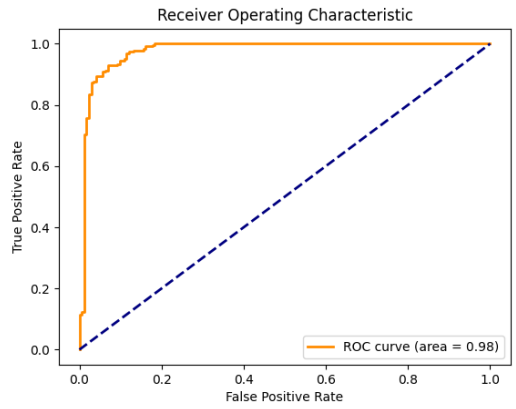

Figure 8: ROC curve of θ.

B.2 Hyperparameters Selection

θ is used as the threshold of selecting similar vectors. Specifically, to determine the threshold of the classifier, We have collected a sample consisting of 400 positive and 400 negative instances. The training data is collected from the exploration process(Section 5.1 Configurations). Since we have conducted exploration and learning for 100 rounds, various labeled data have been generated. We selected positive and negative samples separately and randomly selected 400 as the training set for ROC. Subsequently, the Receiver Operating Characteristic (ROC) curve is constructed based on samples and thereby we obtained the optimal classification threshold θ. The ROC curve is shown in Figure 8.

B.3 Prompting

Based on time and performance considerations, we choose OpenAI[2] GPT-4o API as the employed LLM. We primarily utilize LLM in two parts, with detailed prompts as follows.

B.4 CDRs Judgments

Prompting is used to achieve the functionality of extracting components and making judgments based on CDRs. The specific prompts are shown in Table 4.

B.5 Prompt for Answers

After retrieval, we combine the retrieval contexts and other instructions to prompt the language generator for the final answers. To align with the specific scenario, we follow the method in (Chen et al., 2024), designing task instructions shown in Table 5.

B.6 Results

Table 11,12 shows the full results of CDRs with mRAG and RAG on different language models and methods. Due to space constraints in the main text, we present the complete results here. Table 13,14 and 15 show the results of fixed example ablation on basic LLM planner, RAG LLM planner, and CDRs LLM planner on Café_t dataset, respectively. In this experiment, GPT-3.5-turbo, GPT-4o, CLaude 3.5 Sonnet, Llama2-7b, Llama3-8b and bloomz-7b1 are employed as the language generator. With t ranking from 1 to 5, k set to 5, we conducted 10 repeated experiments and took their average values.

[2] https://platform.openai.com/docs/api-reference

CDR: CDR ϕ_1
Prompt: You are a cautious language assistant. ###[Rules] Here are some language rules: # If the two sentences can be identified as similar, then the subjects, predicates and objects of the two sentences are similar. Be especially mindful of predicate phrases that appear similar but actually have opposite meanings, which make sentences dissimilar. ###[Instructions] Are the following statements similar with the question? Just say True if they are; otherwise just say False. Only output one word.
Sentences: He turned on the radio. He turned off the radio.
Answer: False. ✔

Table 4: Prompts used for CDRs judgments

Task Instruction
[System] [Condition Predicates] Lists all predicates representing conditions and their optional parameter sets. [Objects] Lists all parameter sets. [Few-shot Demonstrations] Provide several examples of Instruction to Goal mapping.
1. Your task is to interpret the input instructions into a goal represented as a well-formed formula in first-order logic. 2. Utilize [Conditions] and [Objects] to construct this goal, and apply logical operators ($\&$, $\|$, $\sim$) appropriately to combine these elements. $\&$ **(AND Operator)**: Combines conditions such that the result is true only if both conditions are true. $\|$ **(OR Operator)**: Combines conditions such that the result is true if at least one of the conditions is true. $\sim$ **(NOT Operator)**: Negates or reverses the truth value of a single condition. 3. The predicate formulas can be converted into disjunctive paradigms (DNFs) using the python package sympy.todnf. 4. Please generate directly interpretable predicate formulas without any additional explanations. 5. Provide only a single line goal formula without specifying any steps.

Table 5: Task instruction for answer generation.

Predicate-condition		
Scenario	Predicate	Category
Café	RobotNear, On, Holding, Exists, Dirty, Active, Closed, Low	<items_place> , <items>, <makable>, <furniture>, <appliance>, <furnishing>, <control>, <place>
Canteen	turn, walk, sit, grab, open, put, close, putin, touch, drink, lookat, switchon, switchoff, walktowards	<direction>, <Food>, <Beverage>, <Cleaning>, <Appliance>, <Furniture>, <Controls>, <Preparation_Tools>, <Dining_Utensils>
Kitchen	turn, walk, sit, grab, open, put, close, putin, touch, drink, lookat, switchon, switchoff, walktowards	<direction>, <Food>, <Controls>, <Beverage>, <Furniture>, <Appliances>, <Furnishings>, <Cooking_tools>, <Dining_utensils>
Lounge	turn, walk, sit, grab, open, put, close, putin, touch, drink, lookat, switchon, switchoff, walktowards	<direction>, <Furniture>, <Appliances>, <Controls>, <Food>, <Beverage>, <Furnishings>, <Daily_Items>, <Miscellaneous>

Table 6: Dataset of Predicate-condition

Object-List (Café)	
Category	**Objects**
<items_place>	<items>+<place>
<makable>	Coffee, Water, Dessert
<Appliance>	AC, TubeLight, HallLight
<Furniture>	Table1, Floor, Chairs
<Controls>	ACTemperature
<Furnishing>	Curtain
<place>	Bar, Bar2, WaterStation, CoffeeStation, Table1, Table2, Table3, WindowTable6, WindowTable4, WindowTable5, QuietTable7, QuietTable8, QuietTable9, ReadingNook, Entrance, Exit, LoungeArea, HighSeats, VIPLounge, MerchZone
<items>	Coffee, Water, Dessert, Softdrink, BottledDrink, Yogurt, ADMilk, MilkDrink, Milk, VacuumCup,Chips, NFCJuice, Bernachon, ADMilk, SpringWater, Apple, Banana, Mangosteen, Orange, Kettle, PaperCup, Bread, LunchBox,Teacup, Chocolate, Sandwiches, Mugs,Watermelon, Tomato, CleansingFoam,CocountMilk,SugarlessGum, MedicalAdhensiveTape, SourMilkDrink, PaperCup,Tissue, YogurtDrink, Newspaper, Box,PaperCupStarbucks, CoffeeMachine, Straw, Cake,Tray, Bread,Glass, Door,Mug, Machine,PackagedCoffee, CubeSugar,Apple, Spoon,Drinks, Drink,Ice, Saucer,TrashBin, Knife,Cube

Table 7: Object-List of Café

Object-List (Canteen)	
Category	**Objects**
<direction>	left, right
<Food>	Apple, Banana, Orange, Peach, Pear, Carrot, Cucumber, Potato, Tomato, Chicken, Salmon, Bread, BellPepper, Bread_Slice, Pancake, Chips, Salad, ChocolateSyrup, Cereal
<Beverage>	Coffee_pot, Milk, Alcohol, Beer, Bottle_water, Wine
<Cleaning>	DishWashing, Dustpan, Washing_sponge, Towel, Towel_rack
<Appliance>	LightSwitch, Power_socket, WallPictureFrame, WallTV, Wall_lamp, Wall_phone
<Furniture>	Chair, Desk, Nightstand, Shelf, Sofa Dining_Table, Coffee_Table, Kitchen_Table
<Controls>	Walllamp, Powersocket, Wallphone, WallTV, Stove, Kettle, OvenTray, Washingmachine, Microwave, LightSwitch, Fridge, CeilingFan, Ceilinglamp, Dishwasher
<Preparation_Tools>	ChefKnife, FryingPan, Stove, Toaster, Kettle, Microwave, Cutting_board, Dishwasher, OvenTray
<Dining_Utensils>	Cutlery_Knife, Cutlery_fork, Spoon, Dish_bowl, Plate, Mug, WaterGlass, WineGlass, Napkin, PaperTowel

Table 8: Object-List of Canteen

Object-List (Kitchen)	
Category	**Objects**
<direction>	left, right
<Food>	Apple, Banana, Orange, Peach, Pear, BellPepper, Carrot, Cucumber, Potato, Tomato, Chicken, Salmon, Bread, Pancake, Chips, Salad, Milk, Alcohol, Beer, Wine, Coffee_pot, Bread_slice, Bottle_water, ChocolateSyrup
<Beverage>	Bottlewater, Beer, Coffeepot, Milk, Alcohol, Wine
<Controls>	Stove, Kettle, Fridge, Dishwasher, Toaster, Microwave, Ceiling_Fan, Ceiling_lamp, Washing_machine, Light_Switch
<Appliance>	Box, Microwave, Fridge, Washingmachine, Dishwasher, Cabinet
<Cooking_tools>	ChefKnife, FryingPan, Stove, Toaster, Kettle, Microwave, Cutting_board
<Furniture>	Chair, Desk, Dining_Table, Kitchen_table, Nightstand, Shelf, Sofa
<Furnishings>	Curtains, WallPictureFrame, WallTV, Wall_lamp, Wall_phone
<Dining_Utensils>	Cutlery_Knife, Cutlery_fork, Spoon, Dish_bowl, Plate, Mug, Wine_Glass, Napkin, Paper_Towel

Table 9: Object-List of Kitchen

Object-List (Lounge)	
Category	**Objects**
<direction>	left, right
<Daily_Items>	Pen, Notebook, Magazine, Book, Boardgame, Cards, TeddyBear, Toy
<Beverage>	WaterGlass, WineGlass, Beer, Wine, Coffee_pot
<Controls>	Dishwasher, Radio, WallTV, Ceilinglamp, CeilingFan, Printer, Projector, Powersocket, LightSwitch, WallPictureFrame, Clock
<Appliance>	HairProduct, BookShelf, Fridge, Door, Microwave, Washingmachine, Box, Curtains, Dishwasher, Cabinet, Drawer, Garbagecan, Window,
<Food>	Apple, Banana, Orange, Peach, Pear, BellPepper, PancakeChips, Salad, Milk, Alcohol, Beer, Wine, Coffee_pot, Bread_slice, Bottle_water, ChocolateSyrup
<Furniture>	BookShelf, Coffeetable, Chair, Nightstand, Sofa, TVstand, Plate
<Furnishings>	WallPictureFrame, Clock, Vase, PhotoFrame, Perfume
<Miscellaneous>	Cellphone, Computer, CPU_screen, Coatack, Window, Faucet, Garbage_can

Table 10: Object-List of Lounge

Language Model	LLM Planner	Café_t		Canteen_t		Kitchen_t		Lounge_t	
		GA	IA	GA	IA	GA	IA	GA	IA
Large-parameter LLM									
GPT-3.5-turbo	base	75	75	71	68	75	60	73	61
	RAG	78 (↑3)	71 (↓4)	72 (↑1)	66 (↓2)	75 (↑0)	66 (↑6)	74 (↑1)	65 (↑4)
	CDRs	100 (↑25)	92 (↑17)	78 (↑7)	72 (↑4)	94 (↑19)	72 (↑12)	89 (↑16)	78 (↑17)
GPT-4o	base	88	82	78	71	77	69	83	67
	RAG	85 (↓3)	68 (↓14)	78 (↑0)	73 (↑2)	88 (↑11)	68 (↓1)	85 (↑2)	72 (↑5)
	CDRs	97 (↑9)	91 (↑9)	89 (↑11)	81 (↑10)	90 (↑13)	88 (↑19)	93 (↑10)	82 (↑15)
Claude 3.5 Sonnet	base	90	85	80	71	82	72	85	73
	RAG	85 (↓5)	68 (↕17)	79 (↓1)	71 (↑0)	80 (↓2)	74 (↑2)	87 (↑2)	72 (↓1)
	CDRs	100 (↑10)	91 (↑6)	90 (↑10)	82 (↑11)	93 (↑11)	84 (↑12)	94 (↑9)	83 (↑10)
Small-parameter LLM									
Llama2-7b	base	17	14	32	21	28	9	22	6
	RAG	33 (↑16)	20 (↑6)	54 (↑22)	46 (↑25)	54 (↑26)	16 (↑7)	42 (↑20)	22 (↑16)
	CDRs	63 (↑46)	51 (↑37)	73 (↑41)	51 (↑30)	68 (↑40)	35 (↑26)	55 (↑33)	33 (↑27)
Llama3-8b	base	64	42	69	40	66	32	22	6
	RAG	61 (↓3)	52 (↑10)	50 (↓19)	41 (↑1)	59 (↓7)	32 (↑0)	50 (↑28)	33 (↑27)
	CDRs	85 (↑21)	72 (↑30)	82 (↑13)	47 (↑7)	75 (↑9)	40 (↑8)	72 (↑50)	41 (↑35)
bloomz-7b1	base	25	5	34	13	39	3	36	3
	RAG	42 (↑17)	20 (↑15)	55 (↑21)	44 (↑31)	46 (↑7)	21 (↑18)	53 (↑17)	20 (↑17)
	CDRs	85 (↑60)	33 (↑28)	79 (↑45)	58 (↑45)	65 (↑26)	33 (↑30)	68 (↑32)	32 (↑29)
falcon-7b	base	34	6	46	4	23	3	16	3
	RAG	46 (↑12)	24 (↑18)	51 (↑5)	21 (↑17)	44 (↑21)	18 (↑15)	35 (↑19)	12 (↑9)
	CDRs	72 (↑38)	37 (↑31)	73 (↑27)	34 (↑30)	77 (↑54)	36 (↑33)	52 (↑36)	31 (↑28)
gemma-2-9b-it	base	47	36	55	39	41	34	40	27
	RAG	63 (↑16)	56 (↑20)	62 (↑7)	58 (↑19)	63 (↑22)	57 (↑23)	64 (↑24)	49 (↑22)
	CDRs	83 (↑36)	72 (↑36)	88 (↑33)	79 (↑40)	81 (↑40)	78 (↑44)	78 (↑38)	59 (↑32)
Qwen2.5-7b-Instruct	base	57	23	53	29	50	27	47	21
	RAG	53 (↓4)	45 (↑22)	59 (↑6)	46 (↑17)	56 (↑6)	40 (↑13)	43 (↓4)	24 (↑3)
	CDRs	78 (↑21)	64 (↑41)	73 (↑20)	69 (↑40)	70 (↑20)	59 (↑32)	72 (↑25)	57 (↑36)

Table 11: Experiment results of RAG on different language models and methods. **Café_t**,**Canteen_t**,**Kitchen_t**,**Lounge_t** denotes the subset of the datasets, which comprises exclusively textual data. Bold numbers indicate the best performance among models. The values in parentheses respectively represent the performance enhancement of the CDRs planner over the RAG planner and the performance improvement of the CDRs planner over the basic planner. **Basic, RAG and CDRs** refer to the basic LLM planner, RAG LLM planner, and CDRs LLM planner, respectively.

Language Model	LLM Planner	Café		Canteen		Kitchen		Lounge	
		GA	IA	GA	IA	GA	IA	GA	IA
Large-parameter LLM									
GPT-4o	base	85	83	72	70	79	73	80	78
	mRAG	86 (↑1)	70 (↓13)	73 (↑1)	73 (↑3)	84 (↑5)	74 (↑1)	83 (↑3)	79 (↑1)
	CDRs	100 (↑15)	89 (↑6)	92 (↑20)	89 (↑19)	92 (↑13)	84 (↑11)	95 (↑15)	83 (↑5)
Claude 3.5 Sonnet	base	89	79	78	73	89	80	84	79
	mRAG	80 (↓9)	76 (↕3)	83 (↑5)	78 (↑5)	85 (↓4)	82 (↑2)	85 (↑1)	82 (↑3)
	CDRs	97 (↑8)	87 (↑8)	95 (↑17)	83 (↑10)	100 (↑11)	92 (↑12)	93 (↑9)	89 (↑10)
Small-parameter LLM									
BLIP-2	base	16	13	28	25	26	23	18	12
	mRAG	35 (↑19)	28 (↑15)	46 (↑18)	43 (↑18)	42 (↑16)	37 (↑14)	38 (↑20)	24 (↑12)
	CDRs	50 (↑34)	47 (↑34)	73 (↑45)	58 (↑33)	68 (↑42)	53 (↑30)	58 (↑40)	48 (↑36)
LLaVA-Llama3-8b	base	63	24	58	43	62	42	63	45
	mRAG	58 (↓5)	36 (↑12)	52 (↓6)	42 (↓1)	59 (↓3)	46 (↑4)	63 (↑0)	56 (↑11)
	CDRs	80 (↑17)	69 (↑45)	83 (↑25)	54 (↑11)	79 (↑17)	56 (↑14)	80 (↑17)	68 (↑23)
MiniGPT-4	base	52	28	57	34	55	37	49	26
	mRAG	48 (↓4)	37 (↑9)	58 (↑1)	42 (↑8)	46 (↓9)	42 (↑5)	48 (↓1)	39 (↑13)
	CDRs	79 (↑27)	68 (↑40)	82 (↑25)	70 (↑36)	69 (↑14)	60 (↑23)	74 (↑25)	63 (↑37)

Table 12: Experiment results of multimodal mRAG on different language models and methods. Bold numbers indicate the best performance among models. The values in parentheses respectively represent the performance enhancement of the CDRs planner over the RAG planner and the performance improvement of the CDRs planner over the basic planner. **Basic, mRAG and CDRs** refer to the basic LLM planner, mRAG LLM planner, and CDRs LLM planner, respectively.

t	1		2		3		4		5	
	GA	IA	GA	IA	GA	IA	GA	IA	GA	IA
Large-parameter LLM										
GPT-3.5-turbo	88	64	88	67	79	58	85	70	94	76
GPT-4o	88	82	100	82	97	79	97	79	88	82
Claude 3.5 Sonnet	90	85	85	70	82	67	85	70	85	80
Small-parameter LLM										
Llama2-7b	7	4	13	8	10	4	10	5	13	10
Llama3-8b	64	42	61	30	58	40	52	36	70	42
bloomz-7b1	25	5	23	8	25	8	39	10	42	12

Table 13: Experiments of Basic LLM planner on different language models.

t	1		2		3		4		5	
	GA	IA	GA	IA	GA	IA	GA	IA	GA	IA
Large-parameter LLM										
GPT-3.5-turbo	82	58	82	55	85	67	85	61	79	64
GPT-4o	85	68	88	70	82	67	91	73	91	76
Claude 3.5 Sonnet	89	76	82	67	73	61	82	67	80	73
Small-parameter LLM										
Llama2-7b	33	20	35	22	32	18	37	24	33	23
Llama3-8b	61	52	67	39	58	39	67	39	58	42
bloomz-7b1	42	20	50	19	47	12	47	7	50	11

Table 14: Experiments of RAG LLM planner on different language models.

t	1		2		3		4		5	
	GA	IA	GA	IA	GA	IA	GA	IA	GA	IA
Large-parameter LLM										
GPT-3.5-turbo	100	92	97	88	100	86	97	83	97	88
GPT-4o	97	91	100	88	100	94	97	94	100	91
Claude 3.5 Sonnet	100	91	100	88	100	91	100	88	100	91
Small-parameter LLM										
Llama2-7b	74	59	68	51	65	52	71	57	74	60
Llama3-8b	85	72	94	70	85	67	91	79	91	67
bloomz-7b1	85	33	84	33	81	35	80	36	82	31

Table 15: Experiments of CDRs LLM planner on different language models.