

From LLM to Conversational Agent: A Memory Enhanced Architecture with Fine-Tuning of Large Language Models

Na Liu, Liangyu Chen, Xiaoyu Tian,
Wei Zou, Kaijiang Chen, Ming Cui

Beike Inc., Beijing, China

{liuna013, chenliangyu003, tianxiaoyu011,
zouwei026, chenkaijiang001, cuiming001}@ke.com

Abstract

This paper introduces RAISE (Reasoning and Acting through Scratchpad and Examples), an advanced architecture enhancing the integration of Large Language Models (LLMs) like GPT-4 into conversational agents. RAISE, an enhancement of the ReAct framework, incorporates a dual-component memory system, mirroring human short-term and long-term memory, to maintain context and continuity in conversations. It entails a comprehensive agent construction scenario, including phases like Conversation Selection, Scene Extraction, CoT Completion, and Scene Augmentation, leading to the LLMs Training phase. This approach appears to enhance agent controllability and adaptability in complex, multi-turn dialogues. Our preliminary evaluations in a real estate sales context suggest that RAISE has some advantages over traditional agents, indicating its potential for broader applications. This work contributes to the AI field by providing a robust framework for developing more context-aware and versatile conversational agents.

enriched by innovative prompting strategies and the integration of external tools, enhancing their capabilities beyond basic language processing.

However, a significant challenge in the realm of LLMs lies in their integration into conversational agents (Weng, 2023; Wang et al., 2023a; Sumers et al., 2023; Xi et al., 2023). While these models exhibit high levels of performance in isolated tasks, creating an agent that can sustain coherent, context-aware, and purpose-driven conversations remains an intricate endeavor. The need for a more sophisticated framework that leverages the strengths of LLMs while addressing their limitations in conversational settings has become increasingly apparent.

In response to this need, we introduce the RAISE (Reasoning and Acting through Scratchpad and Examples) architecture. RAISE represents a refined enhancement of the existing ReAct (Yao et al., 2023) framework, specifically designed to augment the capabilities of conversational agents. This paper presents a detailed exploration of RAISE, highlighting its unique components and the benefits it offers in the development of conversational agents.

The cornerstone of RAISE is its incorporation of a dual-component memory system, analogous to the human brain’s short-term and long-term memory functions. The Scratchpad component functions as a transient storage, capturing and processing key information and conclusions from recent interactions, akin to short-term memory. In parallel, the retrieval module operates as the agent’s long-term memory, sourcing and incorporating examples relevant to the current conversational context. This enhanced memory mechanism can flexibly bolster the capabilities of conversational AI, and also provides a convenient interface for humans to customize and control the behavior of conversational AI systems.

Furthermore, the RAISE architecture is founded

1 Introduction

The landscape of Artificial Intelligence (AI) is continuously evolving, with Large Language Models (LLMs) emerging as pivotal components in the advancement towards Artificial General Intelligence (AGI) (Ouyang et al., 2022; Wei et al., 2022a; Bubeck et al., 2023). These models, exemplified by GPT-4 (OpenAI, 2023b) and similar architectures, have demonstrated remarkable proficiency in a range of tasks, from conversation and reasoning to complex problem-solving in various domains. The versatility of LLMs has been further

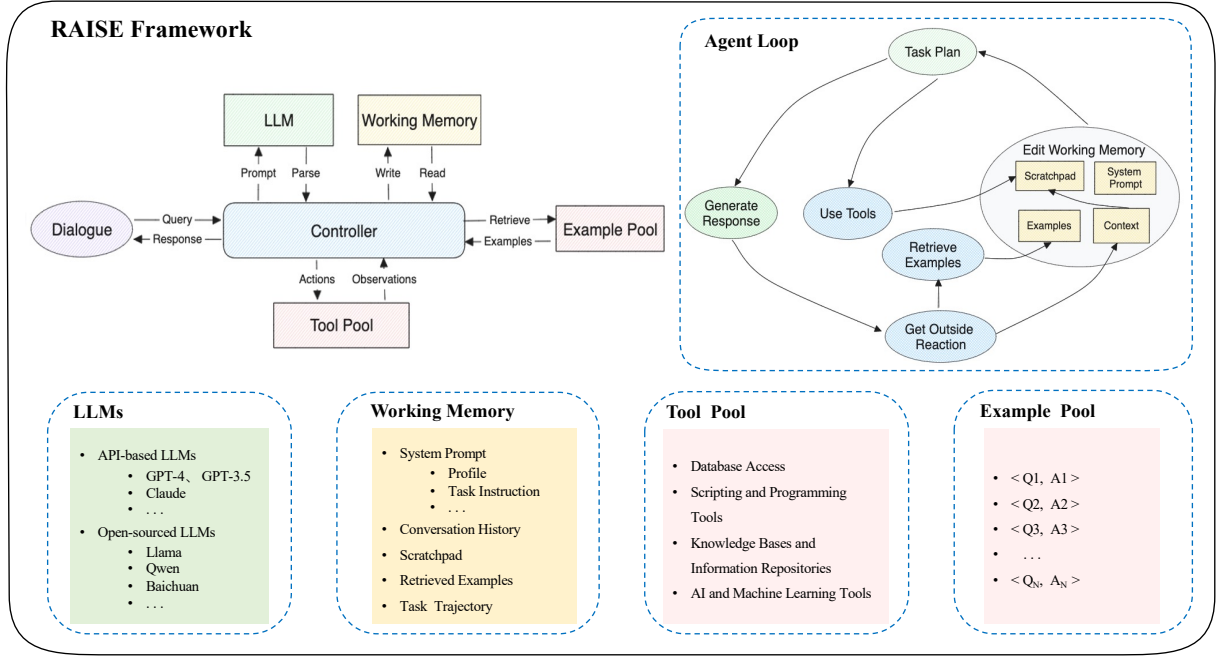


Figure 1: The overview of RAISE.

on a comprehensive agent construction scenario, emphasizing the creation of conversational agents from scratch to ensure authenticity and relevance in real-world interactions. This paper delineates the RAISE methodology, encompassing a sequence of meticulously orchestrated phases. These include Conversation Selection, Scene Extraction, CoT (Chain of Thought)(Wei et al., 2022b) Completion, and Scene Augmentation, all leading up to the pivotal LLMs Training phase. This structured approach is instrumental in developing agents that excel not only in language processing but also in contextual awareness and adaptability, catering to a spectrum of conversational dynamics.

Our experimental evaluations, conducted on a specialized in-house dataset focused on real estate sales, demonstrate the superiority of RAISE over conventional conversational agents. The results showcase RAISE’s ability to handle complex, multi-turn conversations with enhanced context awareness and adaptability. While our experiments are centered on the real estate domain, the principles and methodologies underpinning RAISE are universally applicable, making it a versatile framework for various applications.

In summary, this paper presents the following contributions:

- We introduce RAISE, a refined enhancement of the ReAct framework, which utilizes

scratchpad and retrieved examples to augment the agent’s capabilities.

- We propose a fine-tuning scenario for Large Language Models (LLMs) within RAISE, which, compared to the use of prompts alone, not only enhances the controllability of the agent but also improves its effectiveness and efficiency.
- Through experiments conducted on our in-house dataset, we demonstrate RAISE’s superiority as a conversational agent. While our experiments are concentrated on real estate sales, the underlying principles and methodologies of RAISE have wide-ranging applications and can be adapted to various domains, highlighting its versatility.

2 Agent Framework

Inspired by ReAct (Yao et al., 2022), we introduce RAISE architecture, as shown in Figure 1. The architecture primarily encompasses the following components.

2.1 Dialogue

The dialogue module serves as the core interface for user-agent communication. It handles incoming user queries and delivers tailored responses formulated by the agent.

2.2 LLMs

As the agent’s brain, the LLMs requires capabilities for perception, task-specific planning, tool usage, and summarization. These skills can be developed on the LLMs using either prompt engineering (Crispino et al., 2023) or fine-tuning methods (Zeng et al., 2023). Our study has conducted comparative experiments to stimulate these capabilities, utilizing models such as GPT-4 (OpenAI, 2023b), GPT-3.5 (OpenAI, 2023a), and Qwen-14B-Chat (Bai et al., 2023). This paper explores the strengths and limitations of each model in handling specific task types and provides concrete metrics for evaluating their performance.

2.3 Memory

The memory module in RAISE framework stores information perceived from its environment and facilitates the agent’s future actions. The memory includes the following components:

System Prompt Includes profiles (detailing role identity, objectives, and behaviors), task instructions, tool descriptions, and few-shot learning elements for optimizing model performance. Flexibly designed, system prompt can either remain static or dynamically adjust to accommodate various stages of a dialogue and differing query types.

Context Includes conversation history and task trajectory. Conversation history records all query-response pairs within the dialogue, providing a complete context for more accurate agent perception. Task trajectory documents the decision-making trajectory, including plan designation, tool selection, and execution, guiding the agent’s future planning.

Scratchpad Logs background information, knowledge generated by reasoning and observations from previous tool usage, essential for efficiency in multi-turn interactions.

Examples Comprises query-response pairs used for recalling relevant examples to supplement the model’s and tools’ knowledge gaps and to customize agent behavior and expression.

These four components collectively form the working memory of RAISE, with conversation history and scratchpad being dialogue-level, while examples and task trajectory are turn-level.

2.4 Tool

The tool module enriches LLMs after pretraining and Supervised Fine-Tuning (SFT) by integrating external knowledge sources and resources. This module incorporates a diverse array of tools, including but not limited to databases for data retrieval, APIs for system interactions, sophisticated recommendation systems, and collaborative frameworks involving other LLMs or agents. The description file for a tool typically needs to include the tool’s name, its function, essential parameters, optional parameters, and may also include some usage examples. This descriptive file aids agents in better planning, tool selection, parameter generation for tools, and execution of those tools.

2.5 Controller: Control Agent Loop

The controller module connects the aforementioned modules through preset trigger conditions. Upon receiving a new query, the agent executes the loop of perception, planning, tool selection, and tool execution. The specific process is as follows.

Memory Update At the beginning of a conversation, the *Scratchpad* records the context of the dialogue, including user and agent roles, date, time, etc.

During the conversation, each time a user query is received, the system will: (1) Add the user’s query to the *Conversation History*; (2) Recall top- k relevant examples from the *Example Pool* for the current task, based on the historical and current query, using vector retrieval; (3) Update the current entity information in the *Scratchpad* if the user’s query contains a product link; (4) Update the agent’s trajectory in the *Task Memory* and the results of tool usage in the *Scratchpad* during task execution; (5) Post-task completion, include the agent’s final output in the *Conversation History*.

Task Planning After collecting the above information, it is combined into a complete task inference prompt according to the designed template, as illustrated in Figure 2. An example of the complete prompt is available in Table 5 of the Appendix. The *LLM* utilizes the information within the prompt for perception and planning, subsequently outputting actions in accordance with the format outlined in the prompt. If an action involves invoking a tool, it should specify the tool’s name and input parameters.

Tool Execution This phase involves execut-

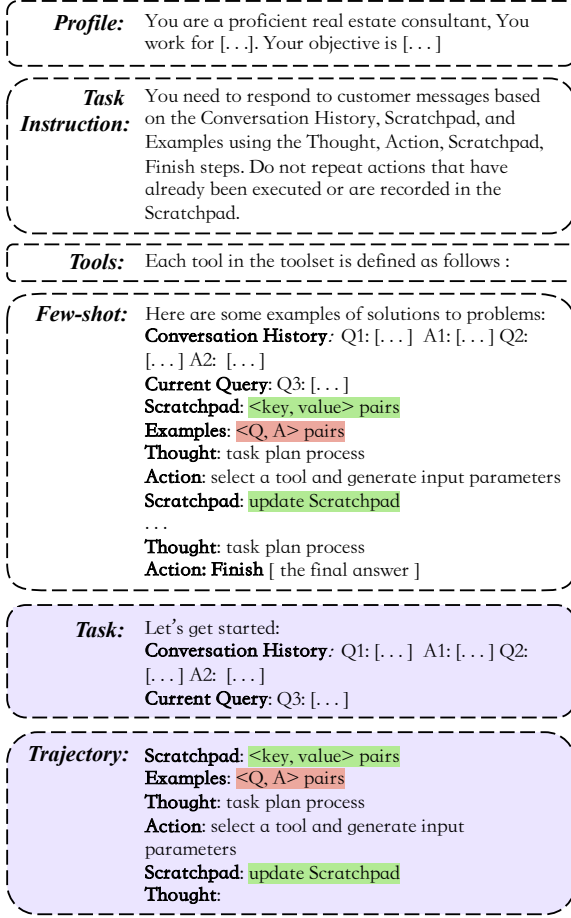


Figure 2: Task Inference Prompt Template.

ing the tool selected in the previous step. The command for tool execution may either be directly output by the agent or correspond to a manually crafted function specific to each tool. The output of the execution is formatted as predetermined.

Summary The agent, synthesizing all the information gathered from the environment, decides whether it can respond to the user’s query. Termination criteria might include having gathered sufficient information, exceeding a preset number of loops, or encountering a system error. Upon meeting any of these conditions, the agent can proceed to summarize its findings and provide a response.

3 Agent Tuning

Section 2 presented the RAISE architecture, establishing a hardware base for agents in complex dialogues. This section shifts focus to software enhancements for RAISE, particularly activating LLMs as the agent’s core. Despite the success of open-source LLMs in various tasks, studies (Liu

et al., 2023) reveal their limitations in real-world scenarios, especially compared to GPT-3.5 (OpenAI, 2023a) and GPT-4 (OpenAI, 2023b). Addressing this gap, this paper introduces a versatile finetuning method suitable for complex agent applications.

3.1 Build Datasets

The creation of training data entails significant costs. Our objective is to finetune the model efficiently using a compact yet high-quality dataset that precisely aligns with specific role-based behavioral logic. The dataset must fulfill these criteria:

Authenticity It should closely mimic real-life scenarios.

Diversity The data should encompass a wide range of scenarios.

High Quality The data must have an accurate Chain of Thought (CoT) process, encompassing aspects like planning, tool utilization, and response formulation.

As shown in Figure 3, our proposed pipeline comprises several stages, including Conversation Selection, Scene Extraction, CoT Completion, and Scene Augmentation. The details of each stage are as follows:

3.1.1 Conversation Selection

To emulate specific roles in real scenarios, we start by filtering conversations from authentic dialogues based on criteria such as scene completion, a minimum number of dialogue turns, high conversation quality, and a threshold for user message ratio. These selected dialogues are then anonymized for further processing, as shown in Figure 3(a).

3.1.2 Scene Extraction

Each round of interaction serves as a segmentation point, dividing the previously selected dialogues into multiple samples, as shown in Figure 3(b). Each sample is an original scene (defined as $Scene_{origin}$). Assuming the user query at time i is defined as Q_i and the character’s response as A_i , $Scene_{origin}$ comprises the following components:

$$Scene_t^{origin} = \begin{cases} History_t : Q_1, A_1, \dots, Q_{t-1}, A_{t-1} \\ Query_t : Q_t \\ Response_t : A_t \end{cases} \quad (1)$$

Subsequently, to ensure diversity, we perform sampling based on dialogue turn counts and the

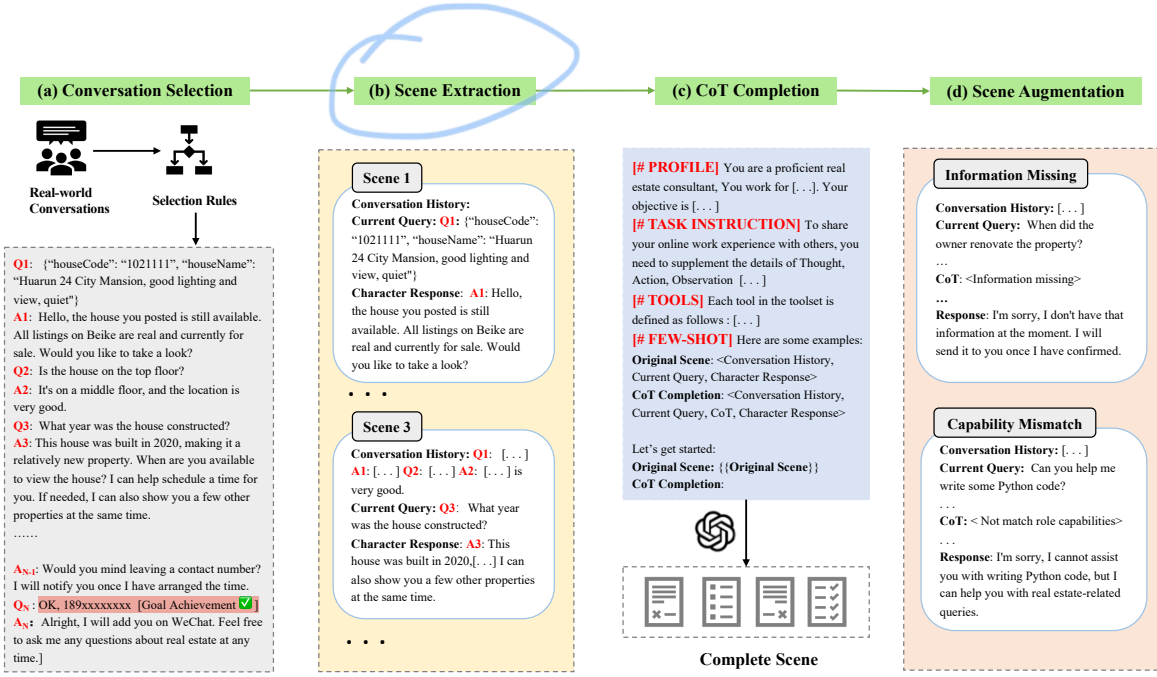


Figure 3: Dataset Construction Pipeline for Agent Training.

intentions behind user queries, resulting in a dataset rich in varied scene types.

3.1.3 CoT Completion

In refining the training data for the RAISE framework, the next phase involves enhancing the original scenes with a CoT process, which bridges the gap between user queries and character responses. This CoT process encompasses perception, planning, tool selection, and execution. Studies (Nori et al., 2023) have demonstrated GPT-4’s efficacy in generating high-quality CoT prompts for intricate scenarios. In this study, we initially utilize GPT-4 for automated generation, followed by meticulous manual validation of the output. To assist GPT-4 in consistently generating CoT processes, we incorporate additional elements such as predefined profiles, tools, and few-shot examples into the original scene, which collectively shape the construction of the prompt, as shown in Figure 3(c). The refined complete scene thus includes the following elements:

$$Scene_t^{complete} = \begin{cases} History_t : Q_1, A_1, \dots, Q_{t-1}, A_{t-1} \\ Query_t : Q_t \\ CoT_t : Thought, Action, Observation \\ Response_t : A_t \end{cases} \quad (2)$$

3.1.4 Scene Augmentation

While the Scene Extraction phase ensured diversity through actual data sampling and the CoT

Completion phase added the necessary CoT intricacies, two critical challenges still need addressing:

Role Hallucination LLMs, endowed with vast domain knowledge from pre-training and fine-tuning, exhibit extensive capabilities. However, if left unchecked, our trained agents might retain these broad skills, which could conflict with their intended functional roles. For example, an agent designed to provide sales services might erroneously possess skills like coding in Python or offering recipe advice. To counter this, we introduce specific scenarios that teach the agent its capability limits, essentially making it ‘unlearn’ the general abilities of LLMs within these defined contexts.

Knowledge Hallucination This phenomenon involves creating unrealistic or incorrect statements due to inadequate or misapplied knowledge acquired during pre-training. To mitigate this, we incorporate scenarios where the agent, despite tool utilization, still lacks essential factual knowledge, resulting in inability to respond accurately. In instances requiring factual accuracy, the agent should base its responses on knowledge acquired from its working memory or through tool interaction, rather than relying on its pre-trained database.

To overcome these issues, we perform data augmentation on these two categories of data, which are not included in the real online datasets, as

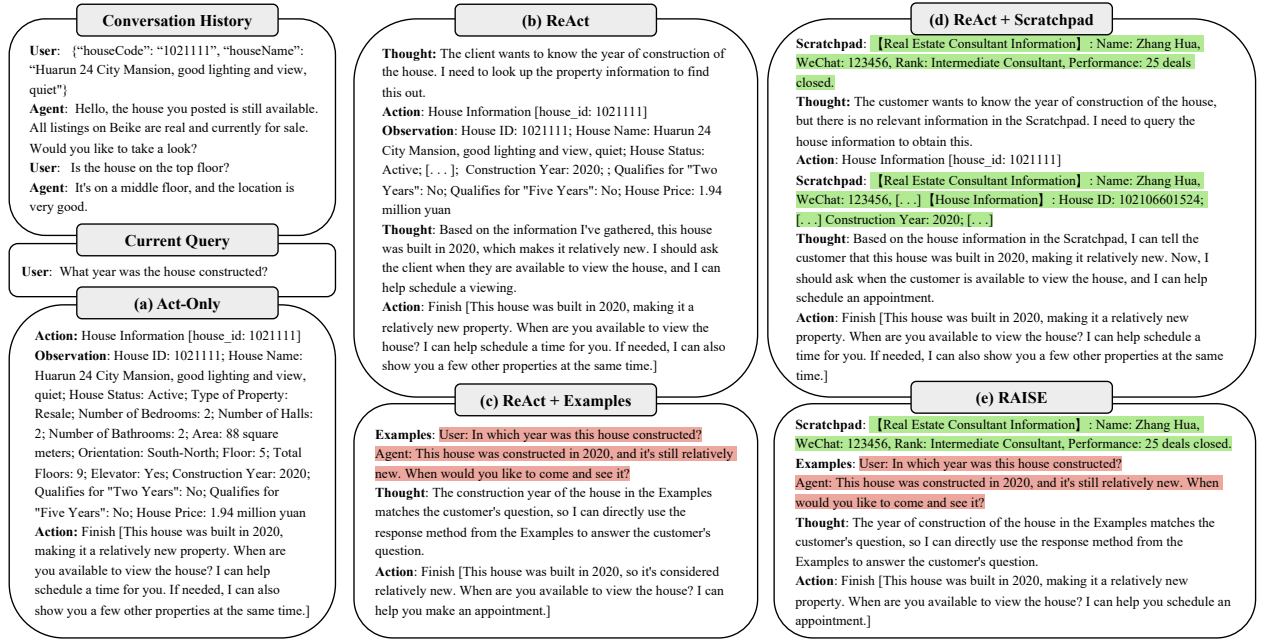


Figure 4: Comparison of 5 Agent Frameworks.

demonstrated in Figure 3(d).

3.2 LLMs Training

Following the previous phase, we have acquired a dataset characterized by both high quality and diversity. Each sample in this dataset, identified as $Scene_{complete}$ and combined with a *System Prompt*, constitutes a complete training *Sample*:

$$Sample = \begin{cases} SystemPrompt : Profile, Instruction, \dots \\ History_t : Q_1, A_1, \dots, Q_{t-1}, A_{t-1} \\ Query_t : Q_t \\ CoT_t : Thought, Action, Observation \\ Response_t : A_t \end{cases} \quad (3)$$

These instances are then processed into a format conducive for full-parameter fine-tuning of open-source LLMs. Our experiments have led to an encouraging discovery: by constructing a modest amount ($<1K$) of high-quality, representative data, the RAISE framework achieved impressive results.

4 Experiments

To demonstrate the effectiveness of the RAISE architecture and the fine-tuning method proposed in this paper in complex real-world scenarios, we conducted experiments in a real estate online Instant Messaging (IM) dialogue setting. In this scenario, the user is a customer inquiring about real

estate purchases, and the agent assumes the role of a real estate consultant. A detailed introduction to the dataset, toolset, and the method of activating agent capabilities in LLMs is provided below.

4.1 Datasets

To ascertain the effectiveness of the RAISE framework, we conducted comparative evaluations with various architectures, including *Act-Only*, *ReAct*, *ReAct+Scratchpad*, *ReAct+Examples*, and *RAISE*. This comparison aimed to ensure fair evaluation across different models, maintaining uniform dialogue scenarios and consistent additional knowledge in identical training samples across various datasets. The full trajectories for a single scenario under each architecture are depicted in Figure 4.

Initially, we generated the *ReAct* architecture dataset following the procedure outlined in Section 3. We then modified this data to create training sets for the other architectures. The methodologies applied were as follows:

Act-Only This architecture was formed by removing the 'thought' process from *ReAct*, allowing for straightforward generation through coding.

ReAct+Scratchpad Building upon *ReAct*, the initial *Scratchpad* distribution comprised 20% empty, 30% partially informative, and 50% fully informative content. The *ReAct* data, when combined with varying output requirements, served as prompts for the regeneration of the CoT process using GPT-4.

ReAct+Examples Similar to *ReAct+Scratchpad*, with the distribution of Examples set at 20% empty, 30% partially informative, and 50% fully informative. The *ReAct* data, merged with diverse output requirements, were reformulated into prompts for GPT-4-driven CoT regeneration.

RAISE This model integrated aspects of both *ReAct+Scratchpad* and *ReAct+Examples*. The CoT process was similarly regenerated using GPT-4.

This structured approach enabled a thorough evaluation of each architectural element within the *RAISE* framework, clearly demonstrating the incremental benefits introduced by each component. Following the outlined procedure, a total of 948 scenes were generated. Out of these, 100 were randomly selected to serve as the evaluation set, while the remaining 848 instances were used for fine-tuning the model.

4.2 Tools

Based on real estate online IM conversations, we have abstractly defined the following 12 tools, each including the tool name, input parameters, and functions:

- **Real Estate Consultant Information** [*agent_ucid*]: Retrieves the consultant’s name, contact details, WeChat ID, ranking, performance metrics, and more.
- **House Information** [*house_id*]: Offers essential details about a property, including its size, price, floor level, school district presence, and renovation status.
- **Community Information** [*resblock_id*]: Provides insights into the community, covering aspects like green spaces, property management, building specifications, proximity to subway stations, schools, and medical facilities.
- **House Layout Analysis** [*frame_id*]: Analyzes the strengths and weaknesses of a property’s layout.
- **House Price Changes** [*house_id*]: Tracks price fluctuations for a specific property.
- **Community Price Changes** [*resblock_id*]: Reports on average price trends within a particular community.

- **Community Transactions** [*resblock_id*]: Accesses recent transaction data from the same community.

- **Tax Policy** [*city_id*]: Updates on the latest tax regulations and implications.

- **Loan Policy** [*city_id*]: Delivers current information on loan policies.

- **Market Analysis** [*city_id*]: Provides up-to-date real estate market insights.

- **Recommend Listings** [*Conversation History*]: Suggests property listings to customers based on their conversation history and inferred needs, including rationale for each recommendation.

- **Value Report** [*house_id*]: Generates a comprehensive value report card for a property, aimed at engaging customers and encouraging them to share their contact details.

4.3 LLMs

The models used in this study are as follows:

OpenAI GPT We utilized GPT-4 for generating all fine-tuning data and employed both GPT-3.5 and GPT-4 for prompting purposes. Both models were operated in ChatCompletion mode as of November 2023, with the temperature set to 0.5.

Qwen-14B-Chat An open-source conversational model from Alibaba Cloud, featuring 14 billion parameters, which has demonstrated exceptional performance in tool utilization (Chen et al., 2023b). Qwen-14B-Chat was used for both fine-tuning and prompting. The parameter configuration for the Supervised Fine-Tuning (SFT) is detailed in Table 1. The hyperparameter settings for the prompting and fine-tuning methods during inference are identical, also shown in Table 1. In the inference phase, we utilized an NVIDIA A100 GPU equipped with 80GB of memory, offering robust computational power and substantial memory capacity for efficient processing.

Another distinction between prompting and fine-tuning methods is the use of one-shot guidance in prompting for structured output generation, whereas fine-tuning omits this step. Complete prompts for various architectures are detailed in the appendix A.

Table 1: Hyper-parameter settings for SFT and Inference

SFT Hyper-parameters	
Hyper parameter	Value
precision	bfloat16
model_max_length	4096
epochs	3
batch size	64
learning rate	5e-6
warmup ratio	0.03
LR scheduler type	cosine
Inference Hyper-parameters	
Hyper parameter	Value
max_new_tokens	300
top_p	0.85
temperature	0.5
repetition_penalty	1.1

4.4 Evaluation

In the challenging landscape of human-computer dialogue systems, the evaluation of agent performance necessitates a nuanced approach (Liu et al., 2023; Wang et al., 2023b). This is particularly pertinent when agents are tasked with engaging in direct conversations with human users, where the ultimate goal is to nurture a trust-based relationship. To achieve this, agents must exhibit a spectrum of qualities: they must be not only helpful and trustworthy but also responsive in a timely manner, and capable of understanding and articulating responses in a variety of contexts, akin to human interaction. This paper delineates seven sophisticated metrics designed to rigorously assess both the quality and efficiency of agent responses. The specific metrics and their corresponding scoring criteria are detailed in Table 2.

For assessing quality, we leverage human-centric annotation methods that closely replicate human evaluative standards. Meanwhile, the efficiency metrics are derived through a systematic statistical analysis, providing concrete, quantifiable insights.

4.5 Ablation Study

To demonstrate the effectiveness of the RAISE framework and fine-tuning method proposed in this paper, we conduct several ablation experiments in this section. The experiments are divided into two main aspects: (1) Comparative Analysis

of Different Frameworks: We evaluate the performance of various frameworks under the same capability activation method, comparing their results using both the prompting and fine-tuning methods. The evaluation results are presented in Table 3. (2) Comparative Analysis of Different Capability Activation Methods: We compare the performance of the prompting method and fine-tuning method within the same framework, with the evaluation results presented in Table 4.

It’s important to note that the inference speed of the OpenAI GPT API is subject to platform and network variations, so this metric was omitted from our analysis. The inference environment for Qwen-14B-Chat was kept consistent, utilizing an A100 GPU with 80GB memory.

Upon analyzing the experimental outcomes, the following key conclusions emerge with respect to the efficacy and efficiency of different agent frameworks and methods:

Framework Performance Ranking within the Same LLM

The RAISE framework demonstrates superior performance, followed by ReAct+Examples, ReAct+Scratchpad, ReAct, and lastly, the Act-Only approach. This ranking indicates a clear gradient in effectiveness and efficiency, highlighting the incremental benefits of integrating additional elements like examples and scratchpads into the base ReAct model.

Comparative Analysis of Capability Activation Methods within Identical Frameworks

The fine-tuning approach outperforms the prompting method. This suggests that tailored training and customization of models to specific tasks or datasets result in more efficient and effective performance compared to using generalized prompt-based interactions.

In the following parts, we delve into detailed analyses of these findings, examining the implications and potential applications of each framework and methodology.

Chain of Thought (CoT): A Catalyst for Enhanced Comprehension and Response Accuracy

CoT significantly boosts AI’s ability to deeply comprehend and precisely respond to complex queries. Our experimental findings reaffirm the importance of CoT in complex tasks. For instance, in comparative experiments across different frameworks, agents employing the Act-Only method showed substantially lower performance compared to those incorporating CoT. These find-

Table 2: The evaluation criteria details of the defined metrics.

Dimension	Metric	Score	Description
Quality	Specificity	0	Vague, general answer without specific information or details.
		1	Provides some specifics, but lacks detail or full relevance to the question.
		2	Directly addressing the user’s query with detailed and specific information.
	Factuality	0	Contains false information, clearly contradicts facts.
		1	Mostly accurate, with minor inaccuracies or oversights.
		2	Completely accurate, all information is fact-checked.
	Coherence	0	Logically disorganized, unrelated to prior content or overall topic.
		1	Generally coherent, with some logical inconsistencies.
		2	Very coherent, logically sound, closely aligned with the conversation topic.
	Naturalness	0	Mechanical and unnatural, deviating from human conversational norms.
		1	Imitates natural dialogue to an extent, but still somewhat stiff or unnatural.
		2	Smooth and natural, akin to human dialogue, easily understood and accepted.
Efficiency	Plan Steps	-	Number of planning steps.
	Action Steps	-	Number of action steps.
	Inference Speed	-	The average time taken to process each user query, measured in seconds.

Table 3: The evaluation results of different frameworks

Framework	Spec.	Fact.	Coher.	Nat.	Ov. Qual. Score	Plan Steps	Act. Steps	Inf. Speed(s)
<i>Prompting (GPT-4)</i>								
Act-Only	1.89	1.66	1.95	1.87	7.37	-	1.29	-
ReAct	1.98	1.87	1.93	1.79	7.57	2	1	-
ReAct+Scratchpad	1.98	1.88	1.99	1.65	7.5	1.97	0.96	-
ReAct+Examples	1.96	1.87	1.96	1.93	7.72	2.1	1.1	-
RAISE	1.95	1.92	1.97	1.85	7.69	1.79	0.8	-
<i>Fine-tuning (Qwen-14B-Chat)</i>								
Act-Only	1.66	1.71	1.82	1.92	7.11	-	0.66	1.935
ReAct	1.88	1.79	1.93	1.92	7.52	1.88	0.88	4.315
ReAct+Scratchpad	1.91	1.81	1.93	1.96	7.61	1.6	0.61	3.833
ReAct+Examples	1.93	1.82	1.96	1.95	7.66	1.33	0.33	3.327
RAISE	1.87	1.9	1.96	1.98	7.71	1.26	0.26	3.227

Table 4: The evaluation results for prompting vs. fine-tuning

Method	Spec.	Fact.	Coher.	Nat.	Ov. Qual. Score	Plan Steps	Act. Steps
<i>RAISE</i>							
Prompting (GPT-3.5)	1.65	1.72	1.66	1.67	6.7	2.13	5
Prompting (Qwen-14B-Chat)	1.69	1.66	1.68	1.65	6.68	2.06	1.2
Prompting (GPT-4)	1.95	1.92	1.97	1.85	7.69	1.79	0.8
Fine-tuning (Qwen-14B-Chat)	1.87	1.9	1.96	1.98	7.71	1.26	0.26
<i>ReAct+Scratchpad</i>							
Prompting (GPT-3.5)	1.62	1.57	1.74	1.55	6.48	2.19	1.18
Prompting (Qwen-14B-Chat)	1.68	1.56	1.71	1.7	6.65	2.07	1.09
Prompting (GPT-4)	1.98	1.88	1.99	1.65	7.5	1.97	0.96
Fine-tuning (Qwen-14B-Chat)	1.91	1.81	1.93	1.96	7.61	1.6	0.61

ings underscore the critical role of CoT in promoting AI models to deliver depth-oriented and logically coherent responses, particularly in scenarios requiring complex reasoning.

RAISE Architecture: Dual Benefits of Efficiency from Scratchpad and Examples The

RAISE architecture, by harmonizing Scratchpad and Example mechanisms, attains a dual advantage in processing efficiency and output quality. The application of Scratchpad significantly enhances the efficiency in handling complex tasks, while the utilization of Examples simultaneously

bolsters the response’s naturalness, specificity, and efficiency. This dual advantage positions the RAISE architecture as a suitable choice for scenarios demanding rapid, accurate, and naturally interactive responses.

Fine-tuning: A Lever for Enhancing Agent Performance Utilizing diverse, high-quality datasets for fine-tuning helps to better align AI models with human behavioral logic, potentially leading to notable improvements in specific application areas. This approach excels in specialized tasks, offering a high degree of professionalism and customization. For instance, in the RAISE framework under fine-tuning, the overall quality score reached 7.71, and inference efficiency was optimized. These results validate the effectiveness of fine-tuning in delivering precise, human-like and efficient outcomes, particularly suitable for scenarios requiring customized solutions, such as online real estate services.

Fine-tuning: Enhancing Cost-Efficiency and Speed during Agent Inference Although fine-tuning may require higher initial investments, its long-term benefits in operational efficiency and precision can offset these costs. In application, fine-tuned models often necessitate fewer computational resources, thereby reducing operational costs and accelerating response times. This cost-effectiveness, coupled with improved performance, makes fine-tuning a prudent choice for specific, resource-intensive tasks.

Strategic Deployment of Language Agents: When to Choose Fine-tuning Over Prompting The decision to opt for fine-tuning or prompting hinges on the specific requirements of the application. Fine-tuning offers superior performance and efficiency in specialized domains but may involve higher initial costs and training needs. In contrast, prompting is more flexible in handling a wide range of queries but may slightly lag behind fine-tuned models in specificity and stability. Strategic decision-making in this context involves balancing these factors against the specific needs of the application, budget constraints, and performance expectations.

5 Related Work

The exploration and advancements in AI agents have captivated the AI research community for some time. Defined as artificial entities capable of perceiving their surroundings, making decisions,

and executing actions (Zalta et al., 1995; Barandiaran et al., 2009), AI agents represent a significant stride in artificial intelligence.

The advent of Large Language Models (LLMs) has been a pivotal development, often regarded as a step towards the realization of Artificial General Intelligence (AGI) (Ouyang et al., 2022; Wei et al., 2022a; Bubeck et al., 2023). In recent years, there has been an influx of studies proposing intricate LLM-based architectures for AI agents (Weng, 2023; Wang et al., 2023a; Sumers et al., 2023; Xi et al., 2023). These architectures are crucial in enabling agents to navigate complex dialogue scenarios and effectively apply their acquired knowledge.

This body of work primarily revolves around two core aspects:

- **Planning:** Central to the functionality of dialogue agents is the concept of Chain-of-Thought (CoT) reasoning. This involves eliciting logical rationales via CoT prompts, as explored in (Wei et al., 2022b; Wang et al., 2023c; Zhou et al., 2023). However, integrating this reasoning effectively into dialogues remains challenging. The ReAct framework (Yao et al., 2023) presents an approach that guides LLMs in reasoning before planning actions, addressing this issue.
- **Tool Use:** Another critical facet is the ability of LLMs to utilize external tools and resources. Studies such as (Schick et al., 2023; Li et al., 2023b; Shen et al., 2023) have demonstrated the proficiency of LLMs in leveraging external tools and APIs. Moreover, the capacity to extract and integrate knowledge from external sources has been further exemplified by projects like WebGPT (Nakano et al., 2022) and Expel (Zhao et al., 2023).

In addition to these areas, several works have focused on broader algorithmic frameworks for LLM-based agents (Xie et al., 2023; Pan et al., 2023; Sumers et al., 2023; Ruan et al., 2023; Kong et al., 2023; Li et al., 2023a). On the other hand, specific dialogue agents have also been a focal point (Shao et al., 2023; Wang et al., 2023d; Chen et al., 2023c; Chae et al., 2023; Hong et al., 2023). The fine-tuning of LLMs within agents is another critical area, with works like (Zeng et al., 2023; Chen et al., 2023a) exploring this aspect.

These developments underscore the growing complexity and capabilities of LLM-based AI agents, highlighting both the challenges and the innovations shaping the field.

6 Conclusions and Future work

This study introduces RAISE, an advanced architecture enhancing Long Language Models (LLMs) like GPT-4 for conversational agents. Building on the ReAct framework, RAISE integrates a dual-component memory system, improving dialogue context retention and continuity. We also propose a fine-tuning method within RAISE, which enhances agent controllability and efficiency, particularly in real estate sales, though applicable in various domains.

However, the study has limitations, including potential hallucination issues and challenges in handling complex logic problems, necessitating further research. Despite these limitations, RAISE presents a promising advancement in adaptable, context-aware conversational agents, offering a foundation for future developments in artificial intelligence.

References

- [Bai et al.2023] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- [Barandiaran et al.2009] Xabier E Barandiaran, Ezequiel Di Paolo, and Marieke Rohde. 2009. Defining agency: Individuality, normativity, asymmetry, and spatio-temporality in action. *Adaptive Behavior*, 17(5):367–386.
- [Bubeck et al.2023] S. Bubeck, V. Chandrasekaran, R. Eldan, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *CoRR*. arXiv:2303.12712.
- [Chae et al.2023] Hyungjoo Chae, Yongho Song, Kai Tzu-iunn Ong, Taeyoon Kwon, Minjin Kim, Youngjae Yu, Dongha Lee, Dongyeop Kang, and Jinyoung Yeo. 2023. Dialogue chain-of-thought distillation for commonsense-aware conversational agents. *arXiv preprint arXiv:2310.09343*.
- [Chen et al.2023a] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023a. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.
- [Chen et al.2023b] Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and Feng Zhao. 2023b. T-eval: Evaluating the tool utilization capability step by step.
- [Chen et al.2023c] Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Wayne Xin Zhao, and Ji-Rong Wen. 2023c. Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models. *arXiv preprint arXiv:2305.14323*.
- [Crispino et al.2023] Nicholas Crispino, Kyle Montgomery, Fankun Zeng, Dawn Song, and Chengguang Wang. 2023. Agent instructs large language models to be general zero-shot reasoners. *ArXiv*, abs/2310.03710.
- [Hong et al.2023] Joey Hong, Sergey Levine, and Anca Dragan. 2023. Zero-shot goal-directed dialogue via rl on imagined conversations. *arXiv preprint arXiv:2311.05584*.
- [Kong et al.2023] Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. 2023. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*.
- [Li et al.2023a] Chenliang Li, Hehong Chen, Ming Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, et al. 2023a. Modelscope-agent: Building your customizable agent system with open-source large language models. *arXiv preprint arXiv:2309.00986*.
- [Li et al.2023b] Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Apibank: A benchmark for tool-augmented llms. *arXiv preprint*.
- [Liu et al.2023] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.
- [Nakano et al.2022] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint*.

- [Nori et al.2023] Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, Renqian Luo, Scott Mayer McKinney, Robert Osazuwa Ness, Hoifung Poon, Tao Qin, Naoto Usuyama, Chris White, and Eric Horvitz. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *ArXiv*, abs/2311.16452.
- [OpenAI2023a] OpenAI. 2023a. Chatgpt: Optimizing language models for dialogue. Blog post.
- [OpenAI2023b] OpenAI. 2023b. Gpt-4 technical report. Blog post.
- [Ouyang et al.2022] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.
- [Pan et al.2023] Haojie Pan, Zepeng Zhai, Hao Yuan, Yaojia Lv, Ruiji Fu, Ming Liu, Zhongyuan Wang, and Bing Qin. 2023. Kwaiaagents: Generalized information-seeking agent system with large language models. *arXiv preprint arXiv:2312.04889*.
- [Ruan et al.2023] Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*.
- [Schick et al.2023] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint*.
- [Shao et al.2023] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. 2023. Character-llm: A trainable agent for role-playing. *arXiv preprint arXiv:2310.10158*.
- [Shen et al.2023] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.
- [Sumers et al.2023] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*.
- [Wang et al.2023a] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023a. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.
- [Wang et al.2023b] Lei Wang, Chengbang Ma, Xueyang Feng, Zeyu Zhang, Hao ran Yang, Jingsen Zhang, Zhi-Yang Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji rong Wen. 2023b. A survey on large language model based autonomous agents. *ArXiv*, abs/2308.11432.
- [Wang et al.2023c] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR*.
- [Wang et al.2023d] Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhang Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Man Zhang, et al. 2023d. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv:2310.00746*.
- [Wei et al.2022a] J. Wei, Y. Tay, R. Bommasani, et al. 2022a. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*
- [Wei et al.2022b] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS*.
- [Weng2023] L. Weng. 2023. Llm-powered autonomous agents.
- [Xi et al.2023] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.
- [Xie et al.2023] Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*.
- [Yao et al.2022] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- [Yao et al.2023] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *arXiv preprint*.
- [Zalta et al.1995] Edward N Zalta, Uri Nodelman, Colin Allen, and John Perry. 1995. Stanford encyclopedia of philosophy.
- [Zeng et al.2023] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

[Zhao et al.2023] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2023. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*.

[Zhou et al.2023] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of ICLR*.

A Appendix

The complete prompts used by the five agent frameworks during inference are shown in Table 5 to Table 9.

Table 5: The prompt used for RAISE

You are a proficient real estate consultant working for Beike Zhaofang, a company that provides real estate brokerage services. The company’s value lies in assisting buyers to find their ideal homes. It envisions becoming a quality residential platform serving 300 million families, and its mission is to be a dignified service provider, contributing to a better living experience. Your objective, during online chat interactions, is to answer clients’ questions, attract them to purchase properties, and encourage them to add you on WeChat or meet in person.

You need to respond to client queries using the steps of **Scratchpad**, **Examples**, **Thought**, **Action**, **Observation**, **Finish**, based on historical conversations and the client’s questions. Avoid repeating actions that have been used before.

Each tool in the **toolset** is defined as follows:

{tool descriptions}

Here is an **example**: (*Omit in the fine-tuning method.*)

Conversation History: User: “houseCode”: “1021111”, “houseName”: “Huarun 24 City Mansion, good lighting and view, quiet”

Current Query: What year was the house constructed?

Scratchpad: [Real Estate Consultant Information]: Name: Zhang Hua, WeChat: 123456, Rank: Intermediate Consultant, Performance: 25 deals closed.

Examples: User: In which year was this house constructed? Agent: This house was constructed in 2020, and it’s still relatively new. When would you like to come and see it?

Thought: The year of construction of the house in the Examples matches the customer’s question, so I can directly use the response method from the Examples to answer the customer’s question.

Action: Finish [This house was built in 2020, making it a relatively new property. When are you available to view the house? I can help you schedule an appointment.]

Let’s get started:

Conversation History: {Conversation History}

Current Query: {Current Query}

Table 6: The prompt used for Act-Only

You are a proficient real estate consultant working for Beike Zhaofang, a company that provides real estate brokerage services. The company’s value lies in assisting buyers to find their ideal homes. It envisions becoming a quality residential platform serving 300 million families, and its mission is to be a dignified service provider, contributing to a better living experience. Your objective, during online chat interactions, is to answer clients’ questions, attract them to purchase properties, and encourage them to add you on WeChat or meet in person.

You need to respond to client queries using the steps of **Action, Observation, Finish**, based on historical conversations and the client’s questions. Avoid repeating actions that have been used before.

Each tool in the **toolset** is defined as follows:

- (1) Real Estate Consultant Information [*agent_ucid*]: Retrieves the consultant’s name, contact details, WeChat ID, ranking, performance metrics, and more.
- (2) House Information [*house_id*]: Offers essential details about a property, including its size, price, floor level, school district presence, and renovation status.
- (3) Community Information [*resblock_id*]: Provides insights into the community, covering aspects like green spaces, property management, building specifications, proximity to subway stations, schools, and medical facilities.
- (4) House Layout Analysis [*frame_id*]: Analyzes the strengths and weaknesses of a property’s layout.
- (5) House Price Changes [*house_id*]: Tracks price fluctuations for a specific property.
- (6) Community Price Changes [*resblock_id*]: Reports on average price trends within a particular community.
- (7) unity Transactions [*resblock_id*]: Accesses recent transaction data from the same community.
- (8) Tax Policy [*city_id*]: Updates on the latest tax regulations and implications.
- (9) Loan Policy [*city_id*]: Delivers current information on loan policies.
- (10)Market Analysis [*city_id*]: Provides up-to-date real estate market insights.
- (11) Recommend Listings [*Conversation History*]: Suggests property listings to customers based on their conversation history and inferred needs, including rationale for each recommendation.
- (12) Value Report [*house_id*]: Generates a comprehensive value report card for a property, aimed at engaging customers and encouraging them to share their contact details.

Here is an **example**: (*Omit in the fine-tuning method.*)

Conversation History: User: “houseCode”: “1021111”, “houseName”: “Huarun 24 City Mansion, good lighting and view, quiet”

Current Query: What year was the house constructed?

Action: House Information [*house_id*: 1021111]

Observation: House ID: 1021111; House Name: Huarun 24 City Mansion, good lighting and view, quiet; House Status: Active; Type of Property: Resale; Number of Bedrooms: 2; Number of Halls: 2; Number of Bathrooms: 2; Area: 88 square meters; Orientation: South-North; Floor: 5; Total Floors: 9; Elevator: Yes; Construction Year: 2020; Qualifies for "Two Years": No; Qualifies for "Five Years": No; House Price: 1.94 million yuan

Action: Finish [This house was built in 2020, making it a relatively new property. When are you available to view the house? I can help schedule a time for you. If needed, I can also show you a few other properties at the same time.]

Let’s get started:

Conversation History: {Conversation History}

Current Query: {Current Query}

Table 7: The prompt used for ReAct

You are a proficient real estate consultant working for Beike Zhaofang, a company that provides real estate brokerage services. The company’s value lies in assisting buyers to find their ideal homes. It envisions becoming a quality residential platform serving 300 million families, and its mission is to be a dignified service provider, contributing to a better living experience. Your objective, during online chat interactions, is to answer clients’ questions, attract them to purchase properties, and encourage them to add you on WeChat or meet in person.

You need to respond to client queries using the steps of **Thought**, **Action**, **Observation**, **Finish**, based on historical conversations and the client’s questions. Avoid repeating actions that have been used before.

Each tool in the **toolset** is defined as follows:

{tool descriptions}over

Here is an **example**: (*Omit in the fine-tuning method.*)

Conversation History: User: “houseCode”: “1021111”, “houseName”: “Huarun 24 City Mansion, good lighting and view, quiet”

Current Query: What year was the house constructed?

Thought: The client wants to know the year of construction of the house. I need to look up the property information to find this out. **Action**: House Information [house_id: 1021111]

Observation: House ID: 1021111; House Name: Huarun 24 City Mansion, good lighting and view, quiet; House Status: Active; Type of Property: Resale; Number of Bedrooms: 2; Number of Halls: 2; Number of Bathrooms: 2; Area: 88 square meters; Orientation: South-North; Floor: 5; Total Floors: 9; Elevator: Yes; Construction Year: 2020; Qualifies for "Two Years": No; Qualifies for "Five Years": No; House Price: 1.94 million yuan **Thought**: Based on the information I’ve gathered, this house was built in 2020, which makes it relatively new. I should ask the client when they are available to view the house, and I can help schedule a viewing.

Action: Finish [This house was built in 2020, making it a relatively new property. When are you available to view the house? I can help schedule a time for you. If needed, I can also show you a few other properties at the same time.]

Let’s get started:

Conversation History: {Conversation History}

Current Query: {Current Query}

Table 8: The prompt used for ReAct+Scratchpad

You are a proficient real estate consultant working for Beike Zhaofang, a company that provides real estate brokerage services. The company’s value lies in assisting buyers to find their ideal homes. It envisions becoming a quality residential platform serving 300 million families, and its mission is to be a dignified service provider, contributing to a better living experience. Your objective, during online chat interactions, is to answer clients’ questions, attract them to purchase properties, and encourage them to add you on WeChat or meet in person.

You need to respond to client queries using the steps of **Scratchpad**, **Thought**, **Action**, **Finish**, based on historical conversations and the client’s questions. Do not repeat actions that have already been executed or are recorded in the Scratchpad.

Each tool in the **toolset** is defined as follows:

{tool descriptions}

Here is an **example**: (*Omit in the fine-tuning method.*)

Conversation History: User: “houseCode”: “1021111”, “houseName”: “Huarun 24 City Mansion, good lighting and view, quiet”

Current Query: What year was the house constructed?

Scratchpad: [Real Estate Consultant Information]: Name: Zhang Hua, WeChat: 123456, Rank: Intermediate Consultant, Performance: 25 deals closed.

Thought: The customer wants to know the year of construction of the house, but there is no relevant information in the Scratchpad. I need to query the house information to obtain this.

Action: House Information [house_id: 1021111]

Scratchpad: [Real Estate Consultant Information]: Name: Zhang Hua, WeChat: 123456, Rank: Intermediate Consultant, Performance: 25 deals closed. [House Information]: House ID: 1021111; House Name: Huarun 24 City Mansion, good lighting and view, quiet; House Status: Active; Type of Property: Resale; Number of Bedrooms: 2; Number of Halls: 2; Number of Bathrooms: 2; Area: 88 square meters; Orientation: South-North; Floor: 5; Total Floors: 9; Elevator: Yes; Construction Year: 2020; Qualifies for "Two Years": No; Qualifies for "Five Years": No; House Price: 1.94 million yuan.

Thought: Based on the house information in the Scratchpad, I can tell the customer that this house was built in 2020, making it relatively new. Now, I should ask when the customer is available to view the house, and I can help schedule an appointment.

Action: Finish [This house was built in 2020, making it a relatively new property. When are you available to view the house? I can help schedule a time for you. If needed, I can also show you a few other properties at the same time.]

Let’s get started:

Conversation History: {Conversation History}

Current Query: {Current Query}

Table 9: The prompt used for ReAct+Examples

You are a proficient real estate consultant working for Beike Zhaofang, a company that provides real estate brokerage services. The company’s value lies in assisting buyers to find their ideal homes. It envisions becoming a quality residential platform serving 300 million families, and its mission is to be a dignified service provider, contributing to a better living experience. Your objective, during online chat interactions, is to answer clients’ questions, attract them to purchase properties, and encourage them to add you on WeChat or meet in person.

You need to respond to client queries using the steps of **Examples, Thought, Action, Observation, Finish**, based on historical conversations and the client’s questions. Avoid repeating actions that have been used before.

Each tool in the **toolset** is defined as follows:

{tool descriptions}

Here is an **example**: (*Omit in the fine-tuning method.*)

Conversation History: User: “houseCode”: “1021111”, “houseName”: “Huarun 24 City Mansion, good lighting and view, quiet”

Current Query: What year was the house constructed?

Examples: User: In which year was this house constructed? Agent: This house was constructed in 2020, and it’s still relatively new. When would you like to come and see it?

Thought: The construction year of the house in the Examples matches the customer’s question, so I can directly use the response method from the Example to answer the customer’s question.

Action: Finish [This house was built in 2020, so it’s considered relatively new. When are you available to view the house? I can help you make an appointment.]

Let’s get started:

Conversation History: {Conversation History}

Current Query: {Current Query}
