

Article

A Training-Free LLM Framework with Interaction Between Contextually Related Subtasks in Solving Complex Tasks

Hongjia Liu and Jinlong Li *

School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China; inalian@mail.ustc.edu.cn

* Correspondence: jlli@ustc.edu.cn

Abstract

Large language models (LLMs) have shown remarkable capabilities in solving complex tasks. Recent work has explored decomposing such tasks into subtasks with independent contexts. However, some contextually related subtasks may encounter information loss during execution, leading to redundant operations or execution failures. To address this issue, we propose a training-free framework with an interaction mechanism, which enables a subtask to query specific information or trigger certain actions in completed subtasks by sending requests. To implement interaction, we introduce a subtask trajectory memory to enable the resumption of completed subtasks upon receiving interaction requests. Additionally, we propose a new action during execution, which generates a concise and precise description of the execution process and outcomes of a subtask, to assist subsequent subtasks in determining interaction targets and requests. Our framework achieves 46.0% and 52.6% success rates on WebShop and HotpotQA, outperforming the strong baselines by 1.0 and 2.6 absolute percentage points, respectively.

Keywords: agents; in-context learning; large language models; task decomposition

1. Introduction

Large language models (LLMs) have demonstrated strong reasoning capabilities across a wide range of tasks. Recently, researchers have explored their use in interactive and decision-making settings, where LLMs iteratively act based on evolving observations. These applications span simulated environments and game-based settings [1–4], as well as real-world agentic tasks, including automated scientific discovery [5,6], operating system control [7,8], and autonomous software engineering [9,10]. Collectively, these works highlight the potential of LLMs as autonomous agents operating within complex and dynamic environments.

In interactive tasks, ReAct [11] relies on an execution trajectory that interleaves reasoning, actions, and observations, and it has been widely adopted in subsequent work [12,13]. However, as tasks become more complex, the interwoven reasoning and acting increases both the length and complexity of the context, which may lead to a decline in the model's performance [14–16].

To mitigate this problem, researchers divide a complex task into subtasks with a planner—for example, statically planning the subtasks before execution [17,18] or dynamically determining the next subtask during execution [19–21]. Although these subtasks can help to reduce the complexity and length of the context [17,22], there are two potential issues: firstly, subtasks might fail due to a lack of certain previous information, especially in



Academic Editor: Katsuhide Fujita

Received: 15 February 2026

Revised: 2 March 2026

Accepted: 7 March 2026

Published: 10 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

tasks like multi-hop question answering [23,24]; secondly, different subtasks may explore the same environment redundantly or look up information repeatedly, which increases the computational resource overhead. We argue that these issues are caused by certain subtasks that are contextually related but executed within independent contexts. This relevance is unavoidable in many cases, because a complex task is difficult to decompose into largely unrelated subtasks. To address this issue, we propose a novel mechanism for exchanging information between subtasks, which we call interaction.

In our interaction, a subtask can send a request to a completed subtask, namely a target subtask, to query specific information or trigger certain actions, accomplished by an in-context learning approach. Our interaction faces several challenges. First, completed subtasks should support resumption from their execution trajectories to handle received interaction requests. Therefore, we introduce a memory module to store the trajectories of completed subtasks, so the target subtask is able to generate a response based on the retrieved trajectory. Second, to support flexible interaction initiation during subtask execution, we add a new interact action to the action space of subtasks and utilize a ReAct-style prompt to guide execution. Thus, the interaction will be triggered as with other task-specific actions during execution.

Moreover, in our interaction, a subtask determines the interaction target and request based on information from all completed subtasks. Describing this information is a challenge, as it must strike a balance between conciseness, to minimize the context overhead, and completeness, to provide essential information. Typically, the goal generated during planning is used to identify a subtask [19,21,25], which we argue is merely a prior plan without actual outcomes and may result in generating inappropriate interaction targets and requests. For example, in Figure 1, as indicated by the red dashed line, subtask 2 generates a request for information unknown to subtask 1 based on its goal. To resolve this, we add a new action, *finish [overview]*, to the action space of subtasks, which reports a short and precise description of its execution, called the overview. The overview encapsulates the execution process and result in a task-specific manner, guiding subsequent subtasks in determining interaction targets and requests. As demonstrated by the green dashed box in Figure 1, the overview describes the partial completion of subtask 1, enabling subtask 2 to successfully query the effective information.

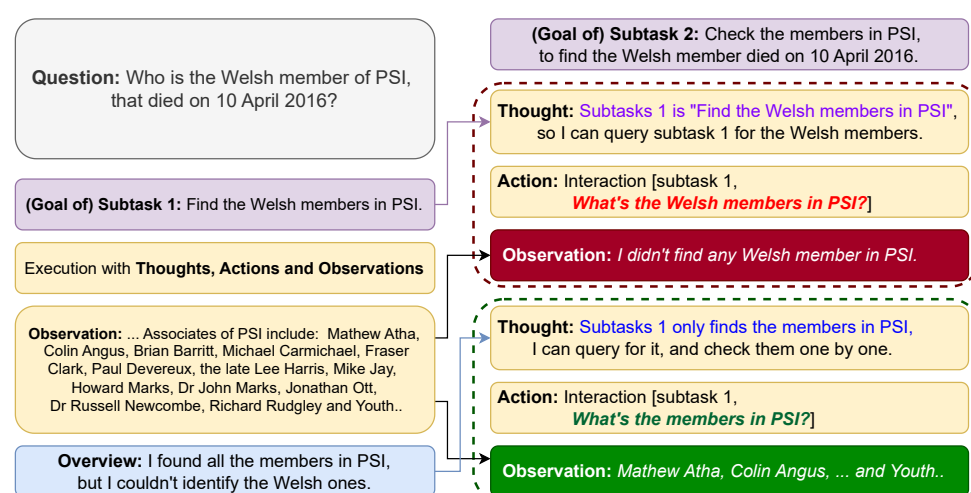


Figure 1. Two interaction requests generated in a question-answering task. **Red dashed box:** Request generated based on goal of subtask 1, which leads to an invalid request because subtask 1 is not accomplished as expected. **Green dashed box:** Request generated based on overview of subtask 1, which is completed with effective information.

To summarize, we introduce an LLM-based framework that enables interaction between subtasks within task decomposition settings. Our contributions are as follows:

- To handle the contextual relevance of subtasks within a task decomposition strategy, we propose a training-free framework with an interaction mechanism, named Interactions For Task Decomposition (IFD), which enables a subtask to perform interactions with completed subtasks in the form of sending requests.
- To handle an interaction request, we propose a memory module to store the execution trajectories of completed subtasks, from which the target subtask retrieves its trajectory to resume and respond to the request.
- To help the executing subtask to determine its interaction target and content, we present a new action to generate a concise and precise description of a subtask, called overview, which describes the execution process and outcomes of a completed subtask.

We evaluate IFD on an interactive decision-making task via WebShop [26] and a data-grounded multi-hop question-answering task via HotpotQA [23]. Compared with previous methods, IFD consistently outperforms all competing approaches on both tasks. Moreover, we assess the contributions of interaction and overview through a statistical analysis and ablation studies. The experiments demonstrate that interaction can lead to performance improvements across various tasks at a small cost.

The remainder of this paper is organized as follows. Section 2 discusses related work on large language model agents and identifies existing research gaps. Section 3 details the design and implementation of the proposed IFD framework. In Section 4, we evaluate the performance of the IFD framework through extensive experiments, including comparative studies, ablation studies, and an overhead analysis. Section 5 discusses the role of the proposed interaction mechanism and addresses the limitations of the framework. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Related Work

2.1. Plan and Execute Strategy

The success of Chain-of-Thought (CoT) [27] demonstrates that LLMs can achieve more powerful reasoning capabilities by decomposing tasks. As tasks become increasingly complex, researchers introduce an independent planner to decompose tasks, thereby alleviating interference from complex execution trajectories [17,28–30]. Building upon static decomposition, subsequent work, such as LLM Compiler [30], introduces a compilation strategy to identify parallelizable subtasks, thereby optimizing the execution efficiency. However, these approaches statically plan the execution process and struggle to handle situations where subtasks cannot be completed. To address this issue, researchers have proposed various dynamic planning methods. For instance, ADAPT [19] introduces a planner that further decomposes subtasks upon subtask failure; AdaPlanner [20] adaptively adjusts its plan based on environmental feedback with an LLM-based planner and refiner.

Task decomposition strategies are also widely applied in multi-agent systems, where complex objectives are partitioned into specialized subtasks assigned to distinct agents to leverage collaborative intelligence. For instance, MetaGPT [31] incorporates standard operating procedures into the multi-agent framework, decomposing software development tasks into specific functional roles. Similarly, ChatDev [32] utilizes a “Chat Chain” architecture, breaking down complex projects into atomic sequential stages, where subtasks are completed through multi-agent dialogs. Beyond these static workflows, more dynamic approaches have emerged to handle unpredictable execution paths. For example, TDAG [21] builds a multi-agent framework where the planner dynamically decomposes or updates subtasks based on the results of completed subtasks, allowing the system to adapt its plan in real time to the evolving execution context.

Despite these advancements, existing planning-and-solving frameworks rarely address the challenge of effective information exchange between subtasks operating in independent contexts. In static planning methods [17,30], information flow is often restricted to passing execution results back to the planner for simple placeholder replacement in subsequent steps. In dynamic planning settings, THREAD [22] introduces a parent–child reporting mechanism, but it is limited to conveying only a restricted amount of information. Similarly, multi-agent studies indicate that decentralized communication often leads to “context drift” and accumulated “information noise”, where critical constraints are lost during inter-agent transmission [31,33–35]. To address these issues, methods such as ADAPT [19] rely on task-specific, hard-coded logic to facilitate communication, which hampers system flexibility and generalizability. Consequently, achieving seamless and general-purpose communication between subtasks within dynamic decomposition frameworks remains a critical research gap that limits the efficacy of agents in handling complex tasks.

2.2. LLMs for Interactive Tasks

Interactive tasks involve agents executing actions and processing observations. Researchers are exploring LLM-based frameworks in these tasks. Several methods [36,37] employ models to simulate or predict execution results in response to potential environmental changes. However, these methods rely solely on the world knowledge of LLMs to simulate the environment, without incorporating actual environmental feedback. To address this issue, ReAct [11] integrates reasoning and execution in a single trajectory, enabling the LLM to update its plans based on environmental feedback. Subsequent work largely builds on ReAct’s approach and introduces optimizations [12,38–43]. For example, Re-flAct [41] enhances ReAct by introducing a goal-state reflection step before action selection to improve the consistency between the current state and the target objective; Pre-Act [42] augments ReAct with explicit multi-step planning, enabling the agent to generate and refine a structured plan prior to executing actions.

Despite the success of ReAct in enhancing agent performance, trajectories that incorporate both reasoning steps and environmental feedback significantly increase the context length and complexity for the LLM, potentially causing degraded reasoning performance and hallucinations [14–16]. Indeed, many existing agent studies have reported issues such as error accumulation and hallucinated intermediate states when operating over extended interaction trajectories [3,44,45]. Consequently, mitigating the negative impacts of such redundant and lengthy trajectories on LLM performance remains a persistent challenge in developing efficient agents.

3. Methodology

We propose IFD, a training-free framework with interaction between subtasks in the form of requests, as shown in Figure 2. IFD consists of a dynamic planner and an executor, as well as a subtask trajectory memory and an interaction handler, both of which are used for interactions between subtasks.

The execution flow of a representative subtask T_{i+1} is depicted in Figure 2. First, the planner generates subtask T_{i+1} based on completed subtasks and their overviews. The executor processes the subtasks in a ReAct-style trajectory until generating an interaction with the subtask T_d via a request. The interaction handler resumes subtask T_d by loading its trajectory from the subtask trajectory memory and iteratively executes to produce a response. After receiving the response, the executor continues executing subtask T_{i+1} and generates an overview upon completion. Lastly, the trajectory and overview of subtask T_{i+1} are stored in the subtask trajectory memory.

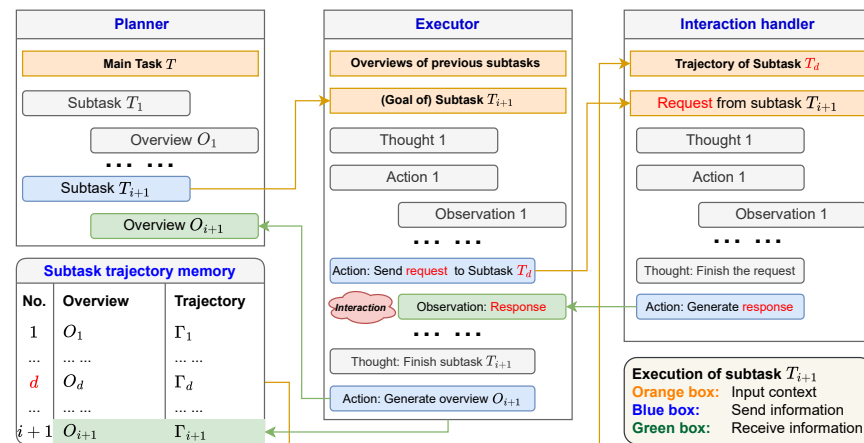


Figure 2. The framework of IFD, illustrating the execution process of subtask T_{i+1} .

3.1. Planner

IFD employs an LLM as a planner to dynamically decompose a complex main task T into easier subtasks $T_1, T_2, \dots, T_n$. To dynamically generate a new subtask T_{i+1} , the planner considers the goals of previous subtasks $[T_i]$, as well as their overviews $[O_i]$, which are short descriptions of the execution processes and results of the subtasks.

Dynamically Planning with Overviews

Given a main task T , the planner dynamically generates T_{i+1} , as shown in Equations (1) and (2).

$$T_1 = LLM_p(\mathcal{T}(T)), \quad (1)$$

$$T_{i+1} = LLM_p(\mathcal{T}([T_i], [O_i])), \quad T > 1, \quad (2)$$

where $LLM_p(\cdot)$ is the LLM-based planner, the sequence $[T_i] = T_1, T_2, \dots, T_i$ denotes the previous subtasks, and $[O_i] = O_1, O_2, \dots, O_i$ corresponds to their overviews. We use a textual pattern $\mathcal{T}$ to structure $[T_i]$ and $[O_i]$ into a coherent description of the solution process of the main task T .

3.2. Executor

For a new subtask T_{i+1} , the executor iteratively generates thoughts and actions to accomplish it, reporting an overview O_{i+1} upon completion. The action space of the executor in IFD includes two additional actions, namely to initiate an interaction or generate an overview.

In particular, the execution trajectory Γ_{i+1}^j of subtask T_{i+1} with j steps of LLM-based executor $LLM_e(\cdot)$ is described in Equation (3).

$$\Gamma_{i+1}^j = \{t_1, a_1, o_1, t_2, a_2, o_2, \dots, t_j, a_j, o_j\}, \quad (3)$$

where t_j and a_j represent a thought and action generated by the executor, and o_j is the subsequent observation from the environment.

3.2.1. Interaction Initiation by Request

To initiate an interaction with a previous subtask during execution, interact is added to the action space $\mathcal{A}$ of the executor, as described in Equation (4).

$$\mathcal{A} \leftarrow \mathcal{A} \cup \text{interact}(d, r), \quad (4)$$

where d indicates the target subtask of this interaction, and r represents a request in natural language.

When the execution of subtask T_{i+1} begins, the executor is provided with overviews $[O_i]$ of previous subtasks to formulate an appropriate target d and request r , as shown in Equation (5).

$$\text{interact}(d, r) = a_{j+1} = \text{LLM}_e([O_i], \Gamma^j \cup t_{j+1}), \quad (5)$$

3.2.2. Overview Generation After Execution

When a subtask is completed, the executor generates an overview as a short description of the execution process and result, which is accomplished by *finish*, another added action, as shown in Equations (6) and (7).

$$\mathcal{A} \leftarrow \mathcal{A} \cup \text{finish}(O_{i+1}), \quad (6)$$

$$\text{finish}(O_{i+1}) = a_{j+1} = \text{LLM}_e([O_i], \Gamma^j \cup t_{j+1}), \quad (7)$$

Besides task completion, the executor is required to generate an overview when the fail or step limit is reached in order to report the encountered difficulties and acquired information.

3.3. Interaction

When an interaction is initiated by the executor, the target subtask resumes and generates a response. To achieve this, we introduce a subtask trajectory memory, storing the trajectories of completed subtasks, and an LLM-based interaction handler for generating responses.

3.3.1. Subtask Trajectory Memory

Subtask trajectory memory $\mathcal{M}$ is a dictionary that stores the execution trajectory Γ_i and overview O_i of each completed subtask T_i , labeled with their numbers i , as shown in Equation (8).

$$\mathcal{M} = \{(1, \Gamma_1, O_1), (2, \Gamma_2, O_2), \dots, (i, \Gamma_i, O_i)\}, \quad (8)$$

After a subtask T_{i+1} finishes, its entire trajectory Γ_{i+1} and generated overview O_{i+1} will be added to memory $\mathcal{M}$.

3.3.2. Interaction Handler

Consider an interaction initiated by the executing subtask T_{i+1} towards a target subtask T_d via a request r . The LLM-based interaction handler $\text{LLM}_i(\cdot)$ resumes T_d by retrieving its historical trajectory Γ_d from the subtask trajectory memory $\mathcal{M}$. To fulfill the request, the interaction handler generates an incremental trajectory $\Delta\Gamma_d$ by conditioning on the historical context Γ_d , as shown in Equation (9).

$$\Delta\Gamma_d = \text{LLM}_i(r \mid \Gamma_d), \quad (9)$$

where $\Delta\Gamma_d = (t_1, a_1, o_1, \dots, t_n, a_n, o_n)$ denotes the sequence of reasoning traces and environmental observations generated during resumption. The execution of $\Delta\Gamma_d$ terminates when it produces a final response *res* via a *finish* action, as shown in Equation (10).

$$a_{last} = \text{last}(\Delta\Gamma_d) = \text{finish}(\text{res}) \quad (10)$$

For simple information queries, $\Delta\Gamma_d$ often comprises single-step reasoning ($n = 1$); for requests necessitating external environmental interaction, it becomes a multi-step

sequence. Finally, the response *res* is returned to the subtask T_{i+1} as a new observation o_k corresponding to its interaction action $a_k = \text{interact}(d, r)$.

4. Experiments

To evaluate our IFD framework, we compare it with baseline methods on an interactive web shopping environment, WebShop [26], and a Wikipedia-based multi-hop question answering benchmark, HotpotQA [23]. We conduct additional analyses and experiments on these two benchmarks to evaluate the impacts of interaction and overview. To provide deeper insight into the interaction mechanism, we further analyze the failure modes encountered in these experiments; detailed discussions are provided in Appendix C.

4.1. WebShop

WebShop [26] is a simulated shopping website environment with 1.18 million real-world products, requiring an agent to purchase a product corresponding to a given instruction. The agent is challenged to interact with various webpages, ranging from search result pages to product description pages; make decisions based on natural language information; select from the available options; and purchase the product that best matches the given instruction, e.g., “I would like a 3-ounce bottle of bright citrus deodorant for sensitive skin and a price lower than USD 50.00”.

4.1.1. Baseline Methods

We compare IFD with the following baseline methods: ReAct [11], a classic prompt-based method that serves as our base executor; TDAG [21], a multi-agent framework featuring a structured communication mechanism; and LATS [46], an LLM framework based on Monte Carlo Tree Search that achieves leading performance in terms of average rewards.

4.1.2. Setup

To ensure consistency with the baseline methods, we employ GPT-3.5-Turbo-16k as the backbone model and conduct our evaluation on the “all” version of the WebShop benchmark. In this setting, the search engine returns up to 50 products, with 3 products displayed per page, and the agent is permitted to examine up to 3 products. The detailed model configurations settings are provided in Appendix A.

We provide the agent with two types of actions to interact with the environment: (1) **search** [product], which triggers a product search and returns the first page of results; (2) **click** [button], which clicks a button on the webpage to navigate to a new page, select product options, or purchase a product. To enable the interaction between subtasks, we introduce two additional actions in IFD: (3) **interact** [target, request], which sends a request to a target subtask and receives the result as an observation from the target subtask, and (4) **finish** [overview], which finishes the current subtask and returns an overview to the planner.

We evaluate the performance of agents using two primary metrics: the success rate and average reward. In the WebShop environment, each task specifies a set of required attributes A_{req} (e.g., brand, color, size). The average reward provides a fine-grained measure of an agent’s performance, calculated as the ratio of matched attributes in the purchased product A_{match} to the total number of requirements, i.e., $R = |A_{match}| / |A_{req}| \in [0, 1]$. The success rate represents the proportion of trials that achieve a perfect match, where $R = 1.0$.

4.1.3. Results

To ensure a robust and statistically significant evaluation, we conducted 500 trials across five independent runs (seeds 0–4). We report the mean performance alongside the standard deviation to evaluate stability, and we further employ the bias-corrected and

accelerated (BCa) bootstrapping method to derive 95% confidence intervals (CI), as shown in Table 1. For the results of the baseline methods, with the aim of maintaining consistent backbone models and experimental settings, we report the highest values found in the baseline literature. We take the results of ReAct from the paper on TDAG [21], and the results of other baseline methods are directly taken from the original papers. We explicitly label the data sources for each metric in the tables.

The statistical robustness of IFD is validated through aggregated results over 500 trials across five random seeds. As summarized in Table 1, IFD achieves a stable mean reward of $76.2\% \pm 1.2\%$ and a success rate of $46.0\% \pm 2.8\%$. The narrow 95% BCa confidence intervals further demonstrate its consistent performance across diverse data samples.

Table 1. Statistical results of IFD on WebShop across 5 independent runs (100 trials per seed).

Metric	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean $\pm$ Std	95% CI (BCa)
Reward (%)	76.4	74.7	75.6	76.0	78.3	76.2 ± 1.2	[73.7, 78.5]
Success Rate (%)	48.0	42.0	50.0	45.0	45.0	46.0 ± 2.8	[41.6, 50.4]

When compared with the baseline methods shown in Table 2, IFD achieves leading performance. Specifically, IFD reaches an average success rate of 46.0% and an average reward of 76.2%, which represent substantial improvements of 17% and 34.1%, respectively, over the ReAct baseline. Notably, IFD surpasses the previous state-of-the-art success rate of 45% held by TDAG and the highest average reward of 75.9% reported for LATS. While the existing baseline methods often excel in only one of these two metrics, IFD demonstrates more balanced effectiveness, maintaining a margin of over 10% in the secondary metric compared to each respective leading baseline.

Table 2. Success rate and average reward on WebShop with baseline methods, with GPT-3.5-turbo as backbone model. The best results are shown in bold.

Method	Success Rate (%)	Average Reward (%)	Data Origin
ReAct	29.0	42.1	Reported in TDAG [21]
TDAG	45.0	64.5	Original Paper [21]
LATS	38.0	75.9	Original Paper [46]
IFD (Ours)	46.0	76.2	This Work

Compared to TDAG, a multi-agent framework where subagents communicate with a central controller only upon task completion, our IFD framework employs a flexible interaction mechanism that allows subtasks to interact with previous subtasks during execution. While achieving comparable success rates to TDAG, IFD yields an 11.7% improvement in the mean reward. This gain is attributed to the dynamic exploration enabled by inter-task interactions, facilitating finer adjustments in complex environments—a mechanism that is further analyzed in the subsequent sections.

Additionally, supplementary statistics on task decomposition and subtask execution within the IFD framework are provided in Appendix B.

4.1.4. Interaction Enhances Exploration and Reward

In the WebShop environment, search results are presented with three items per page. The baseline methods [11,46] typically employ a static exploration strategy, confined to selecting the product with the highest predicted reward from the initial three results on the first page. While this approach simplifies the execution path, it significantly restricts the agent's ability to explore a broader search space.

In contrast, our IFD framework introduces a dynamic exploration strategy: a check subtask requests the next products with specific requirements by interacting with the search subtask. The search subtask retrieves more results by flipping pages if no suitable product is found on the current page, as shown in Figure 3.

Experimental results demonstrate that this dynamic exploration strategy based on interaction allows the agent to identify potential optimal solutions beyond the first page, even under the constraint of “inspecting no more than three products per task”. In 54 representative cases where the purchased product originated from subsequent pages, the reward increased by an average of 0.21 compared to simply purchasing the top-scoring item on the first page, which indicates that our interaction indeed enhances exploration and improves the results.

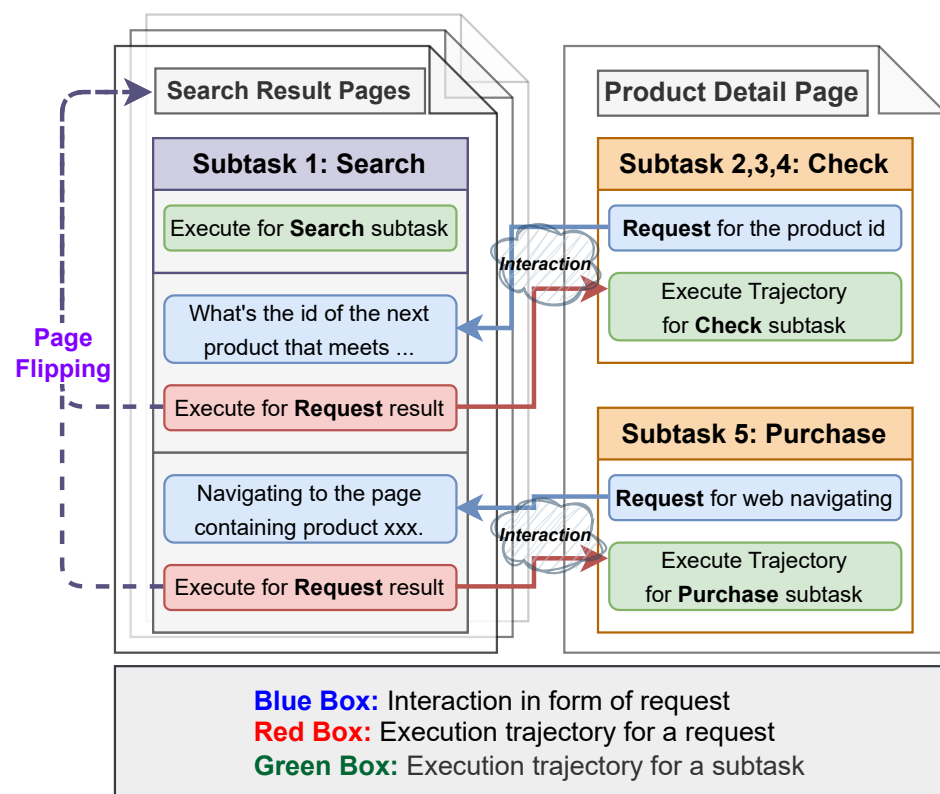


Figure 3. Execution process of IFD in WebShop, including two kinds of interactions. The search subtask searches for results. The check subtask requests the search subtask for the ID of the next product that meets specific requirements. The buy subtask requests the search subtask to navigate to the page containing a certain product.

4.1.5. Interaction Reduces Context Complexity

While dynamic exploration is logically superior, conventional methods struggle to integrate it due to the resultant context inflation. In static strategies, candidate products are fixed immediately after the initial search. Conversely, dynamic exploration involves frequent page transitions and complex decision-making across multiple states, which significantly increases the length and noise of the agent’s context.

Our IFD framework mitigates this issue by leveraging an inter-subtask interaction mechanism to achieve contextual decoupling. As illustrated in Figure 3, interaction ensures that each subtask operates within a streamlined, single-page environment. The search subtask focuses exclusively on global information from search result pages and processes incoming interaction requests. The check and purchase subtasks concentrate on specific product details. When cross-page information is required, these subtasks send interaction

requests to the corresponding subtask, rather than performing physical page flips themselves. Under this mechanism, redundant page information generated during transitions is filtered out by the interaction handler. This ensures that subtasks always execute within a clear, localized context, shielding the model from the performance degradation typically caused by environmental complexity.

To quantify the role of inter-subtask interaction in reducing context complexity, we conducted an ablation study. While maintaining the dynamic exploration strategy, we removed the interaction mechanism from the IFD framework. We compared this framework with the full IFD framework across the aforementioned 54 cases where IFD purchased items not located on the initial search result page.

We report the success rate and average reward, as well as the occurrence of hallucinations during execution. Specifically, we categorized hallucinations into two types: (1) page hallucinations, where the agent misidentifies its current page (e.g., attempting a purchase on a page lacking a “buy” button) and (2) information hallucinations, where the agent misjudges the attributes of a product (e.g., failing to recognize a “waterproof” feature despite it being explicitly stated on the product detail page).

The experimental results are presented in Table 3. Upon removing the interaction mechanism, subtasks are forced to handle contextual information across multiple webpages, causing the “No Hallucination” rate to drop from 98.1% to 38.9%. Specifically, the IFD without interaction encounters information hallucinations in 33.3% of cases and page hallucinations in 38.9% of cases, with 11.1% of cases exhibiting both simultaneously. This leads to a significant performance decline, where the success rate falls from 42.6% to 20.4% and the average reward falls by 23.0%.

Table 3. Comparison of hallucination rates and performance metrics between IFD and IFD without interaction. All values are expressed as percentages (%).

Method	Page Hallu.	Info Hallu.	No Hallu.	Success Rate	Average Reward
Full IFD	0.0	1.9	98.1	42.6	79.7
Without Interaction	33.3	38.9	38.9	20.4	56.7

These results indicate that, under a dynamic exploration strategy, frequent webpage transitions within a unified context introduce significant environmental noise, which interferes with the agent’s decision-making. IFD utilizes interactions between subtasks to achieve contextual decoupling: it offloads the complex context from the executor to the interaction handler. This ensures that each subtask executes within an independent page environment, thereby shielding the agent’s performance from the interference of contextual complexity.

4.1.6. Cost Analysis

The subtask interaction mechanism introduced by the IFD framework entails additional model inference overhead. To quantitatively evaluate its impact on computational resources, we measured the average input and output tokens of the LLM API, alongside the average number of API calls. These metrics are categorized into two types: (1) for interactions between subtasks, which include both the triggering of interactions and the generation of corresponding responses, and (2) for task execution, which accounts for the overhead dedicated to task processing independently of subtask communication.

Statistical data are presented in Table 4. In the WebShop environment, interaction between subtasks accounts for approximately 22% of the total token consumption. We attribute this overhead to the cross-page operations facilitated by the subtask interaction mechanism. In WebShop, each page transition returns a new page as an observation, which

contains extensive product information, thereby increasing the context length. Additionally, interaction calls represent roughly 25.3% of the total API calls.

Table 4. Quantitative analysis of average computational resource consumption per case in WebShop.

Metric	Task Execution	Subtask Interaction	Total
Input Tokens	36,965 (78.0%)	10,421 (22.0%)	47,386 (100%)
Output Tokens	877 (77.8%)	250 (22.2%)	1127 (100%)
Total Tokens	37,842 (78.0%)	10,671 (22.0%)	48,513 (100%)
API Calls	16.5 (71.2%)	5.6 (25.3%)	22.1 (100%)

Beyond the overall results, we also quantified the computational costs for both IFD and its “without interaction” variant within the ablation study from the preceding subsection, in order to further analyze the trade-off between the additional interaction overhead and the resultant performance gains. As shown in Table 5, while the full IFD framework introduces an 18.1% increase in total tokens and a 20.5% increase in API calls, the benefits are substantial: the hallucination rate is drastically reduced from 61.1% to 1.9%, accompanied by a 23.0% improvement in the average reward. Consequently, the reward per 1K tokens for the full IFD framework is 1.55, demonstrating superior resource efficiency relative to the performance gains.

Table 5. Quantitative analysis of average computational resource consumption per case in ablation study.

Method	Input Tokens	Output Tokens	Total Tokens	API Calls
Full IFD	50,141	1195	51,336	22.3
Without Interaction	42,521	940	43,461	18.5
Overhead Increase	7620 (17.9%)	255 (27.1%)	7875 (18.1%)	3.8 (20.5%)

4.2. HotpotQA

HotpotQA [23] is an interactive Wikipedia-based dataset consisting of 113,000 question-and-answer pairs, where an agent is challenged to search for external documents and reason across multiple pieces of supporting information.

For example, a typical multi-hop question is “When was the former Indian cricketer who made his ODI debuts during the 2000 ICC KnockOut Trophy born?”.

4.2.1. Baseline Methods

Since methods with dynamic task decomposition are rarely evaluated on multi-hop question answering, we adopt ReAct [11] and two recent LLM-based methods, LEAP [47] and the method presented in [12] (referred to as the Q model), as the baseline approaches. To further demonstrate the capabilities of our model, we compare IFD with two approaches refined through multiple attempts, Reflexion [38] and Retroformer [48], which have achieved state-of-the-art performance among LLM-based methods.

4.2.2. Setup

To ensure a fair and consistent comparison, we align our backbone models and benchmark setup with the baseline methods [11,12,38,47,48], using GPT-4 and GPT-4o-mini, and conduct evaluations on the “hard” subset of HotpotQA with a question-only setup. The detailed model configurations settings are provided in Appendix A.

In this configuration, agents are provided only with the initial question and must interact with a Wikipedia API or retriever to gather external facts, rather than having direct access to supporting paragraphs. We define the action space as follows:

(1) **search** [entity], which searches for the entity on Wikipedia and returns the first paragraph if it exists; (2) **lookup** [keyword], which returns the next sentence containing the keyword from the page successfully found by the search; and (3) **answer** [answer], which submits the final answer. To enable interaction between subtasks, we introduce two additional actions in IFD: (4) **interact** [target, request], which sends a query to a target subtask and receives the result as an observation, and (5) **finish** [overview], which concludes the current subtask and returns its overview.

Consistent with baseline work [11,12,38,47,48], we adopt the success rate based on exact match (EM) as our metric. This metric is computed via the HotpotQA evaluation pipeline: after standard normalization, which includes lowercasing, stripping leading and trailing whitespace, and removing punctuation and articles, a generated response is considered a “success” only if it achieves an identical string match with the ground truth.

4.2.3. Results

To ensure a robust and statistically significant evaluation, we conducted 500 trials across five independent runs (seeds 0–4). We report the mean performance alongside the standard deviation to evaluate stability, and we further employ the bias-corrected and accelerated (BCa) bootstrapping method to derive 95% confidence intervals (CI), as shown in Table 6. For the results of the baseline methods, with the aim of maintaining consistent backbone models and experimental settings, we report the highest values found in the baseline literature. We take the results of ReAct from the paper on the Q-model [12] and the results of Reflexion from the paper on Retroformer [48]. Results for other baseline methods are directly taken from the original papers. We explicitly label the data sources for each metric in the tables.

Table 6. Statistical results of IFD on HotpotQA across 5 independent runs (100 trials per seed), using GPT-4o-mini and GPT-4 as backbone models.

Backbone Model	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean $\pm$ Std	95% CI (BCa)
GPT-4	54.0	52.0	50.0	57.0	50.0	52.6 $\pm$ 2.7	[48.2%, 57.0%]
GPT-4o-mini	45.0	44.0	41.0	50.0	40.0	44.0 $\pm$ 3.5	[39.8%, 48.4%]

The statistical robustness of IFD is validated by results obtained across 500 trials (five seeds), as shown in Table 6. Using GPT-4, IFD achieves a mean success rate of 52.6% $\pm$ 2.7% (95% CI: [48.20, 57.00]), while GPT-4o-mini yields 44.0% $\pm$ 3.5% (95% CI: [39.8, 48.4]). These stable metrics across varied seeds confirm that IFD provides reliable and reproducible performance.

As shown in Table 7, IFD demonstrates architectural effectiveness, achieving a success rate of 52.6% on GPT-4 and 44.0% on GPT-4o-mini. Under aligned backbone models, IFD maintains a performance margin of 7.4% to 13.0% over the ReAct baseline. Notably, IFD achieves comparable performance to the training-based Q-model while remaining entirely training-free.

Table 7. Success rate (%) on HotpotQA compared with single-pass execution methods. Methods marked with an asterisk (*) require additional training. The best results are shown in bold.

Single-Pass Method	GPT-4	GPT-4o-mini	Data Origin
ReAct	44.0	31.0	Reported in Q-Model [12]
LEAP	40.0	–	Original Paper [47]
Q-Model *	50.0	44.0	Original Paper [12]
IFD (Ours)	52.6	44.0	This Work

As shown in Table 8, IFD achieves a success rate of 52.6% in a single-pass manner. This performance is comparable to the optimal results reported for Retroformer with one retry and exceeds the 52% success rate of Reflexion with four retries, trailing the four-retry Retroformer by only 1.4%. Notably, IFD achieves these results without the additional training required by Retroformer or the multiple computational attempts employed by both baselines.

Additionally, supplementary statistics on task decomposition and subtask execution within the IFD framework are provided in Appendix B.

Table 8. Success rate (%) on HotpotQA compared with methods using multiple attempts. Methods marked with an asterisk (*) require additional training. The best results are shown in bold.

Method	Retries	GPT-4	Data Origin
Reflexion	1	46.0	Reported in Retroformer [48]
	4	52.0	
Retroformer *	1	51.0	Original Paper [48]
	4	54.0	
IFD (Ours)	0	52.6	This Work

4.2.4. Interaction Enables Multi-Hop Reasoning with Subtasks

We analyzed 100 execution trajectories of IFD on GPT-4o-mini. On average, each example involved retrieving approximately 3.08 Wikipedia paragraphs. IFD divides each example into an average of 3.1 subtasks, reducing the context to roughly 0.99 paragraphs per subtask. This reduction in contextual complexity is enabled by interaction, allowing subtasks to exchange specific information rather than processing the entire trajectory history.

In HotpotQA, reasoning steps often rely on information from previous hops. We argue that the common failure of task decomposition in multi-hop tasks stems from subtasks being unable to access necessary previous information. IFD addresses this through the interact action, which facilitates the distribution of the reasoning chain across independent, manageable subtasks.

4.2.5. Overview Promotes Interaction

In IFD, the current subtask determines the target and request of an interaction based on overviews of completed subtasks. To investigate the role of overviews, we conducted an ablation study replacing overviews with simple task “goals”. We measured the failure rate as the proportion of interactions associated with failed subtasks among all subtask interactions. Specifically, an interaction is considered a failure if the request asks for information that is unavailable to the target subtask, resulting in responses such as “IDK”.

The experimental results are presented in Table 9. When utilizing the specific goals of completed subtasks for representation, the failure rate of inter-subtask interactions reaches 24.2%. In contrast, the failure rate when using subtask overviews is significantly lower at only 3.6%. This marked improvement suggests that subtask overviews facilitate the generation of more effective interaction targets and requests.

Table 9. Interaction failure rates: comparing overviews vs. goals for completed subtasks.

Metric	Overview	Goal
Total interactions	110	190
Failed interactions	4	46
Failure rate (%)	3.6	24.2

Notably, compared to providing subtask goals, the use of subtask overviews leads to a substantial 42.1% reduction in the total number of interactions between subtasks. We attribute this decrease to two primary factors: first, the executor may adjust the interaction target or query to retry after a failed interaction; second, some subtask overviews inherently contain the information required by subsequent subtasks, thereby eliminating the need for additional interaction requests. These findings demonstrate that overviews effectively curtail unnecessary interactions, thereby optimizing the utilization of computational resources.

4.2.6. Cost Analysis

Similarly to the WebShop experiments, we measured the average LLM API input/output tokens and API calls for IFD within HotpotQA to quantify the computational overhead incurred by subtask interactions, with GPT-4o-mini as the backbone model.

Statistical data are presented in Table 10. In HotpotQA, the overhead introduced by subtask interactions is minimal, accounting for only approximately 5% of the total tokens. We attribute this efficiency to the fact that interactions in HotpotQA primarily serve as internal queries to previous subtask histories, which can be resolved through single-step reasoning. Furthermore, these interaction queries are processed without the need for extensive in-context learning demonstrations; the interaction handler can directly derive answers based on the historical trajectory, significantly reducing input token consumption. Additionally, with an average of only 2.4 extra API calls per task, the communication remains highly efficient without leading to a surge in the request volume.

Table 10. Quantitative analysis of average computational resource consumption per case in HotpotQA.

Metric	Task Execution	Subtask Interaction	Total
Input Tokens	17,600 (95.0%)	925 (5.0%)	18,525 (100%)
Output Tokens	3143 (97.8%)	71 (2.2%)	3214 (100%)
Total Tokens	20,743 (95.4%)	996 (4.6%)	21,739 (100%)
API Calls	6.7 (73.7%)	2.4 (26.3%)	9.1 (100%)

Overall, IFD enables dynamic information transfer between subtasks with a minimal token cost. This addresses the challenge whereby standard task decomposition strategies struggle with multi-hop reasoning due to the lack of information exchange between subtasks, demonstrating the framework's high efficiency and scalability in complex tasks.

5. Discussion

Across different benchmarks, we consistently apply interaction using two standardized actions: *interact* [*target*, *request*] and *finish* [*overview*].

In WebShop, the interaction mechanism ensures that each subtask is executed within a single webpage environment, with all cross-page operations handled through requests. By delegating navigation to the interaction layer, we avoid frequent page transitions within a single subtask in dynamic exploration strategies. This reduction in trajectory complexity minimizes the negative impact of context fragmentation on the large language model's reasoning performance.

In HotpotQA, interaction facilitates information transfer between subtasks with independent contexts. This mechanism effectively manages the complex information-passing logic inherent in multi-hop reasoning tasks, enabling the agent to perform step-by-step reasoning based on specific data retrieved from previous subtasks.

Despite the superior performance and versatility demonstrated by the IFD framework, certain limitations remain. First, the proposed methodology currently relies on manual ICL examples for interaction guidance. While this ensures high-quality logical consistency,

it presents a potential generalization bottleneck, as the framework's flexibility may be constrained when transitioning to entirely unseen or highly dynamic scenarios. Second, in environments where subtasks are contextually unrelated, the interaction mechanism may yield diminishing returns. However, as a dynamic and selective mechanism, the interaction does not incur unnecessary computational overhead in such settings. Consequently, the IFD framework was not evaluated on benchmarks such as ALFWorld [49], where tasks are typically completed through simple sequential execution without the need for data exchange or functional coordination between subtasks.

6. Conclusions

Task decomposition is widely adopted for solving complex tasks using LLMs. However, when the decomposed subtasks are contextually related, task execution may fail due to the independence of their execution contexts. To address this issue, we propose IFD, a training-free framework that incorporates an interaction mechanism to enable information transfer between related subtasks. Experimental results demonstrate that IFD achieves leading performance on the WebShop and HotpotQA benchmarks. Furthermore, additional experiments conducted on 500 random cases validate the stability and robustness of the IFD framework across various task instances.

In future work, we aim to employ lightweight fine-tuning techniques to internalize the *interact [target, request]* and *finish [overview]* actions within the large language model. This approach seeks to enhance the interaction accuracy while developing a more generalized cross-task framework that reduces the reliance on specific interaction demonstrations in the prompt. Additionally, exploring advanced trajectory summarization and generating more representationally powerful overviews remain key priorities to further optimize the information density. Furthermore, extending the application of this interaction mechanism across a broader spectrum of complex, heterogeneous tasks remains a key direction for our subsequent research.

Author Contributions: Conceptualization, H.L. and J.L.; methodology, H.L. and J.L.; software, H.L.; validation, H.L. and J.L.; formal analysis, H.L. and J.L.; writing—original draft preparation, H.L.; writing—review and editing, H.L. and J.L.; visualization, H.L.; supervision, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of this study are derived from publicly available resources. The WebShop environment and datasets can be accessed at <https://github.com/princeton-nlp/WebShop> (accessed on 6 March 2026), and the HotPotQA dataset is available at <https://hotpotqa.github.io/> (accessed on 6 March 2026).

Acknowledgments: We thank Jinlong Li for his valuable feedback and assistance in refining the language of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Model Configuration

In all experiments, the executor, planner, and interaction handler modules within the IFD framework utilize consistent large language model configurations, with the exception of specific stop tokens (termination rules) tailored to each module.

Regarding the specific models employed, we use GPT-3.5-turbo-16k for the WebShop task. For HotpotQA, we utilize both GPT-4o-mini-2024-07-18 and GPT-4-0613. To ensure

experimental rigor and reproducibility, identical decoding parameters are applied across all tests. The specific API parameter settings for these backbone models are detailed in Table A1.

Table A1. API parameter settings for backbone models.

Parameter	Setting
Temperature	0
Frequency penalty	0.0
Presence penalty	0.0
Seed	0
Max tokens (HotpotQA)	200
Max tokens (WebShop)	500
Stop tokens (Planner)	['\n', '\n\n']
Stop tokens (executor and handler)	['Observation', 'observation', '\n\n']

Appendix B. Statistics on Subtasks

In this section, we provide supplementary statistics on task decomposition and execution. Table A2 reports the mean values for subtask counts, steps per subtask, and the number and frequency of interaction actions relative to total execution steps. Regarding experimental constraints, the maximum number of subtasks is capped at 5, and the step limit per subtask is set to 15 for both WebShop and HotpotQA.

Table A2. Detailed statistics on subtask decomposition and execution.

Benchmark	Model	Subtasks	Steps per Subtask	Interactions per Task
WebShop	GPT-3.5-Turbo	3.82	5.79	2.80 (12.67%)
HotpotQA	GPT-4	3.15	2.70	1.30 (15.26%)
HotpotQA	GPT-4o-mini	3.09	2.95	1.20 (13.16%)

Appendix C. Failure Mode Analysis for Interaction

In this section, we analyze the failure modes associated with the interaction between subtasks.

In the WebShop environment, since interactions are utilized to execute cross-page navigation (i.e., navigating between different webpages and retrieving partial information), we observed no instances of unresponsive interactions. However, in HotpotQA, across 500 test cases, GPT-4o-mini produced 12 unresponsive interactions, resulting in a failure rate of 2.2%. In contrast, GPT-4 produced only one unresponsive interaction, with a failure rate of 0.2%.

All unresponsive interactions share a common underlying cause: the model fails to locate information regarding a specific entity on Wikipedia during a subtask. Consequently, it attempts to query previous subtasks for the missing data, even when the overviews of these historical tasks clearly do not contain the required information. An example of this failure is illustrated below.

Task: Which airport is further north, Valley International Airport or Aberdeen Regional Airport?

Subtask 1: Find the latitude of Valley International Airport.

Overview 1: Success to find the latitude of Valley International Airport.

Subtask 2: Find the latitude of Aberdeen Regional Airport

...execute process but failed

Thought: I could not find the latitude of Aberdeen Regional Airport.
 Maybe I can query Subtask 1 for some information.
 Action: interact [1, What's the latitude of Aberdeen Regional Airport?]
 Observation: IDK.

We attribute this error to a deficiency in instruction-following capabilities. To address this, in future work, we plan to optimize the prompting strategy or perform lightweight fine-tuning to improve interaction generation.

References

1. Zhou, S.; Xu, F.F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. In Proceedings of the The Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
2. Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. AgentBench: Evaluating LLMs as Agents. In Proceedings of the The Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
3. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv* **2023**, arXiv:2305.16291. [\[CrossRef\]](#)
4. Szot, A.; Schwarzer, M.; Agrawal, H.; Mazouze, B.; Metcalf, R.; Talbott, W.; Mackraz, N.; Hjelm, R.D.; Toshev, A.T. Large Language Models as Generalizable Policies for Embodied Tasks. In Proceedings of the The Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
5. Sun, Q.; Liu, Z.; Ma, C.; Ding, Z.; Xu, F.; Yin, Z.; Zhao, H.; Wu, Z.; Cheng, K.; Liu, Z.; et al. ScienceBoard: Evaluating Multimodal Autonomous Agents in Realistic Scientific Workflows. *arXiv* **2025**, arXiv:2505.19897. [\[CrossRef\]](#)
6. Lu, C.; Lu, C.; Lange, R.T.; Foerster, J.; Clune, J.; Ha, D. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *arXiv* **2024**, arXiv:2408.06292. [\[CrossRef\]](#)
7. Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T.J.; Cheng, Z.; Shin, D.; Lei, F.; et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In Proceedings of the The Thirty-Eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, Vancouver, BC, Canada, 9–15 December 2024.
8. Hu, X.; Xiong, T.; Yi, B.; Wei, Z.; Xiao, R.; Chen, Y.; Ye, J.; Tao, M.; Zhou, X.; Zhao, Z.; et al. OS Agents: A Survey on MLLM-based Agents for General Computing Devices Use. *arXiv* **2025**, arXiv:2508.04482. [\[CrossRef\]](#)
9. Miserendino, S.; Wang, M.; Patwardhan, T.; Heidecke, J. SWE-Lancer: Can Frontier LLMs Earn \$1 Million from Real-World Freelance Software Engineering? *arXiv* **2025**, arXiv:2502.12115.
10. Bouzenia, I.; Pradel, M. Understanding Software Engineering Agents: A Study of Thought-Action-Result Trajectories. *arXiv* **2025**, arXiv:2506.18824.
11. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv* **2022**, arXiv:2210.03629.
12. Zhai, Y.; Yang, T.; Xu, K.; Feng, D.; Yang, C.; Ding, B.; Wang, H. Enhancing decision-making for LLM agents via step-level Q-value models. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, 25 February–4 March 2025; AAAI Press: Palo Alto, CA, USA, 2025; AAAI'25/IAAI'25/EAAI'25. [\[CrossRef\]](#)
13. Rozanov, N.; Rei, M. StateAct: State Tracking and Reasoning for Acting and Planning with Large Language Models. *arXiv* **2024**, arXiv:2410.02810.
14. Dziri, N.; Lu, X.; Sclar, M.; Li, X.L.; Jiang, L.; Lin, B.Y.; West, P.; Bhagavatula, C.; Bras, R.L.; Hwang, J.D.; et al. Faith and Fate: Limits of Transformers on Compositionality. *arXiv* **2023**, arXiv:2305.18654. [\[CrossRef\]](#)
15. Shi, F.; Chen, X.; Misra, K.; Scales, N.; Dohan, D.; Chi, E.H.; Schärli, N.; Zhou, D. Large language models can be easily distracted by irrelevant context. In Proceedings of the International Conference on Machine Learning, PMLR, Honolulu, HI, USA, 23–29 July 2023; pp. 31210–31227.
16. Liu, N.F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; Liang, P. Lost in the Middle: How Language Models Use Long Contexts. *arXiv* **2023**, arXiv:2307.03172. [\[CrossRef\]](#)
17. Khot, T.; Trivedi, H.; Finlayson, M.; Fu, Y.; Richardson, K.; Clark, P.; Sabharwal, A. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. *arXiv* **2022**, arXiv:2210.02406.
18. Sun, S.; Liu, Y.; Wang, S.; Zhu, C.; Iyyer, M. PEARL: Prompting Large Language Models to Plan and Execute Actions Over Long Documents. *arXiv* **2023**, arXiv:2305.14564. [\[CrossRef\]](#)

19. Prasad, A.; Koller, A.; Hartmann, M.; Clark, P.; Sabharwal, A.; Bansal, M.; Khot, T. ADaPT: As-Needed Decomposition and Planning with Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, 16–21 June 2024; pp. 4226–4252. [\[CrossRef\]](#)
20. Sun, H.; Zhuang, Y.; Kong, L.; Dai, B.; Zhang, C. Adaplaner: Adaptive planning from feedback with language models. In *Thirty-Seventh Conference on Neural Information Processing Systems*; NeurIPS: San Diego, CA, USA, 2024; Volume 36, pp. 58202–58245.
21. Wang, Y.; Wu, Z.; Yao, J.; Su, J. TDAG: A multi-agent framework based on dynamic Task Decomposition and Agent Generation. *Neural Netw.* **2025**, *185*, 107200. [\[CrossRef\]](#) [\[PubMed\]](#)
22. Schroeder, P.; Morgan, N.W.; Luo, H.; Glass, J.R. THREAD: Thinking Deeper with Recursive Spawning. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, Albuquerque, NM, USA, 29 April–4 May 2025; Volume 1, pp. 8418–8442. [\[CrossRef\]](#)
23. Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; Manning, C.D. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 2369–2380. [\[CrossRef\]](#)
24. Qi, P.; Lee, H.; Sido, O.; Manning, C.D. Answering open-domain questions of varying reasoning steps from text. *arXiv* **2020**, arXiv:2010.12527.
25. Sun, C.; Huang, S.; Pompili, D. Retrieval-Augmented Hierarchical in-Context Reinforcement Learning and Hindsight Modular Reflections for Task Planning with LLMs. *arXiv* **2024**, arXiv:2408.06520.
26. Yao, S.; Chen, H.; Yang, J.; Narasimhan, K. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. *arXiv* **2022**, arXiv:2207.01206.
27. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Thirty-Seventh Conference on Neural Information Processing Systems*; NeurIPS: San Diego, CA, USA, 2022; Volume 35, pp. 24824–24837.
28. Lin, B.Y.; Fu, Y.; Yang, K.; Brahman, F.; Huang, S.; Bhagavatula, C.; Ammanabrolu, P.; Choi, Y.; Ren, X. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. In *Thirty-Seventh Conference on Neural Information Processing Systems*; NeurIPS: San Diego, CA, USA, 2024; Volume 36, pp. 23813–23825.
29. Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R.K.W.; Lim, E.P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv* **2023**, arXiv:2305.04091.
30. Kim, S.; Moon, S.; Tabrizi, R.; Lee, N.; Mahoney, M.W.; Keutzer, K.; Gholami, A. An LLM compiler for parallel function calling. *arXiv* **2023**, arXiv:2312.04511.
31. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In Proceedings of the The Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
32. Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. ChatDev: Communicative Agents for Software Development. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, Bangkok, Thailand, 11–16 August 2024; Volume 1, pp. 15174–15186. [\[CrossRef\]](#)
33. Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Chan, C.M.; Yu, H.; Lu, Y.; Hung, Y.H.; Qian, C.; et al. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In Proceedings of the The Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
34. Liu, Z.; Zhang, Y.; Li, P.; Liu, Y.; Yang, D. Dynamic LLM-Agent Network: An LLM-agent Collaboration Framework with Agent Team Optimization. *arXiv* **2024**, arXiv:2310.02170.
35. Li, G.; Hammoud, H.A.A.K.; Itani, H.; Khizbullin, D.; Ghanem, B. CAMEL: Communicative Agents for “Mind” Exploration of Large Language Model Society. In Proceedings of the Thirty-Seventh Conference on Neural Information Processing Systems, New Orleans, LA, USA, 10–16 December 2023.
36. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv* **2022**, arXiv:2204.01691. [\[CrossRef\]](#)
37. Hao, S.; Gu, Y.; Ma, H.; Hong, J.J.; Wang, Z.; Wang, D.Z.; Hu, Z. Reasoning with language model is planning with world model. *arXiv* **2023**, arXiv:2305.14992. [\[CrossRef\]](#)
38. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. In Proceedings of the 37th International Conference on Neural Information Processing Systems—NIPS’23, New Orleans, LA, USA, 10–16 December 2023.
39. Ma, K.; Zhang, H.; Wang, H.; Pan, X.; Yu, W.; Yu, D. LASER: LLM Agent with State-Space Exploration for Web Navigation. *arXiv* **2023**, arXiv:2309.08172.
40. Yang, Z.; Li, P.; Yan, M.; Zhang, J.; Huang, F.; Liu, Y. ReAct Meets ActRe: Autonomous Annotations of Agent Trajectories for Contrastive Self-Training. *arXiv* **2024**, arXiv:2403.14589.

41. Kim, J.; Rhee, S.; Kim, M.; Kim, D.; Lee, S.; Sung, Y.; Jung, K. ReflAct: World-Grounded Decision Making in LLM Agents via Goal-State Reflection. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, Suzhou, China, 4–9 November 2025; pp. 33433–33465. [\[CrossRef\]](#)
42. Rawat, M.; Gupta, A.; Goomer, R.; Bari, A.D.; Gupta, N.; Pieraccini, R. Pre-Act: Multi-Step Planning and Reasoning Improves Acting in LLM Agents. *arXiv* **2025**, arXiv:2505.09970.
43. Dongre, V.; Yang, X.; Acikgoz, E.C.; Dey, S.; Tur, G.; Hakkani-Tür, D. ReSpAct: Harmonizing Reasoning, Speaking, and Acting Towards Building Large Language Model-Based Conversational AI Agents. *arXiv* **2025**, arXiv:2411.00927.
44. Chakraborty, T.; Ghosh, U.; Zhang, X.; Niloy, F.F.; Dong, Y.; Li, J.; Roy-Chowdhury, A.; Song, C. HEAL: An Empirical Study on Hallucinations in Embodied Agents Driven by Large Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, 4–9 November 2025; pp. 21226–21243. [\[CrossRef\]](#)
45. Hadeliya, T.; Jauhar, M.A.; Sakpal, N.; Cruz, D. When Refusals Fail: Unstable Safety Mechanisms in Long-Context LLM Agents. *arXiv* **2025**, arXiv:2512.02445. [\[CrossRef\]](#)
46. Zhou, A.; Yan, K.; Shlapentokh-Rothman, M.; Wang, H.; Wang, Y.X. Language agent tree search unifies reasoning, acting, and planning in language models. In Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024.
47. Zhang, T.; Madaan, A.; Gao, L.; Zheng, S.; Mishra, S.; Yang, Y.; Tandon, N.; Alon, U. In-context principle learning from mistakes. *arXiv* **2024**, arXiv:2402.05403. [\[CrossRef\]](#)
48. Yao, W.; Heinecke, S.; Niebles, J.C.; Liu, Z.; Feng, Y.; Xue, L.; Ramapura Narasimha Murthy, R.; Chen, Z.; Zhang, J.; Arpit, D.; et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
49. Shridhar, M.; Yuan, X.; Côté, M.A.; Bisk, Y.; Trischler, A.; Hausknecht, M. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. *arXiv* **2020**, arXiv:2010.03768.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.