TRAD: Enhancing LLM Agents with Step-Wise Thought Retrieval and Aligned Decision

Ruiwen Zhou
skyriver@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Yingxuan Yang
zoeyyx@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Muning Wen
muningwen@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Ying Wen
ying.wen@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Wenhao Wang
wangwenhao-009@cpic.com.cn
China Pacific Insurance
Shanghai, China

Chunling Xi
xichunling@cpic.com.cn
China Pacific Insurance
Shanghai, China

Guoqiang Xu
xuguoqiang-009@cpic.com.cn
China Pacific Insurance
Shanghai, China

Yong Yu
yyu@apex.sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Weinan Zhang
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Numerous large language model (LLM) agents have been built for different tasks like web navigation and online shopping due to LLM's wide knowledge and text-understanding ability. Among these works, many of them utilize in-context examples to achieve generalization without the need for fine-tuning, while few of them have considered the problem of how to select and effectively utilize these examples. Recently, methods based on trajectory-level retrieval with task meta-data and using trajectories as in-context examples have been proposed to improve the agent's overall performance in some sequential decision making tasks. However, these methods can be problematic due to plausible examples retrieved without task-specific state transition dynamics and long input with plenty of irrelevant context. In this paper, we propose a novel framework (*TRAD*) to address these issues. *TRAD* first conducts ***T**hought **R**etrieval*, achieving step-level demonstration selection via thought matching, leading to more helpful demonstrations and less irrelevant input noise. Then, *TRAD* introduces ***A**ligned **D**ecision*, complementing retrieved demonstration steps with their previous or subsequent steps, which enables tolerance for imperfect thought and provides a choice for balance between more context and less noise. Extensive experiments on ALFWorld and Mind2Web benchmarks show that TRAD not only outperforms state-of-the-art models but also effectively helps in reducing noise and promoting generalization. Furthermore, TRAD has been deployed in real-world scenarios of a global business insurance company and improves the success rate of robotic process automation. Our codes are available at: https://github.com/skyriver-2000/TRAD-Official.

KEYWORDS

Large Language Model, LLM Agent, Sequential Decision Making, LLM Reasoning, Information Retrieval

1 INTRODUCTION

Large Language Models (LLMs) [3, 32] have achieved remarkable success on various tasks like question answering [45], chatbot [20], code synthesis [24], text ranking [7], table-based reasoning [43], and retrieval query expansion [17] due to their wide knowledge and excellent ability of text understanding and generation. Recently, a series of works have attempted to build powerful agents based on LLMs for various sequential decision-making tasks, including text-based games [41], online shopping [40], web navigation [4], and information retrieval [48].

Among existing LLM agents, some are trained with large-scale expert data by supervised fine-tuning (SFT) [8, 9, 18], while some are tuning-free and utilize in-context learning (ICL) with few expert demonstration examples [13, 34, 42, 46]. In this paper, we focus the scope on tuning-free ICL methods, as they are highly cost-effective and can seamlessly generalize to different tasks using only a small amount of expert samples. Most existing ICL-based agents are prompted with expert trajectories carefully selected by human [28, 38, 42], which work well when few expert trajectories are available. However, when we have access to a large dataset of expert trajectories or an expert policy, the automatic and personalized selection of expert trajectories for each task instruction becomes necessary, and can have an essential influence on task performance.

Recently, Zheng et al. [46] study the problem of demonstration selection and propose *Synapse*, which retrieves relevant expert trajectories by task meta-data, and then prompts LLMs with these retrieved trajectories. *Synapse* performs well on computer control tasks (MiniWob++ [27]) and web navigation tasks (Mind2Web [4]). Nevertheless, retrieving and prompting with complete trajectories can be problematic in the following three aspects.

Plausible examples. Sometimes generalization to data from various domains can be critical. For example, in cross-website and cross-domain subsets of Mind2Web, agents operate on websites unseen in the training set, i.e., memory. In this case, retrieving trajectories with only task meta-data is very likely to provide plausible examples, which share similar task instructions to the current one but require totally different solutions. As shown by experiments in [46], plausible examples provide no more information than random examples and can usually mislead LLM agents to wrong decisions.

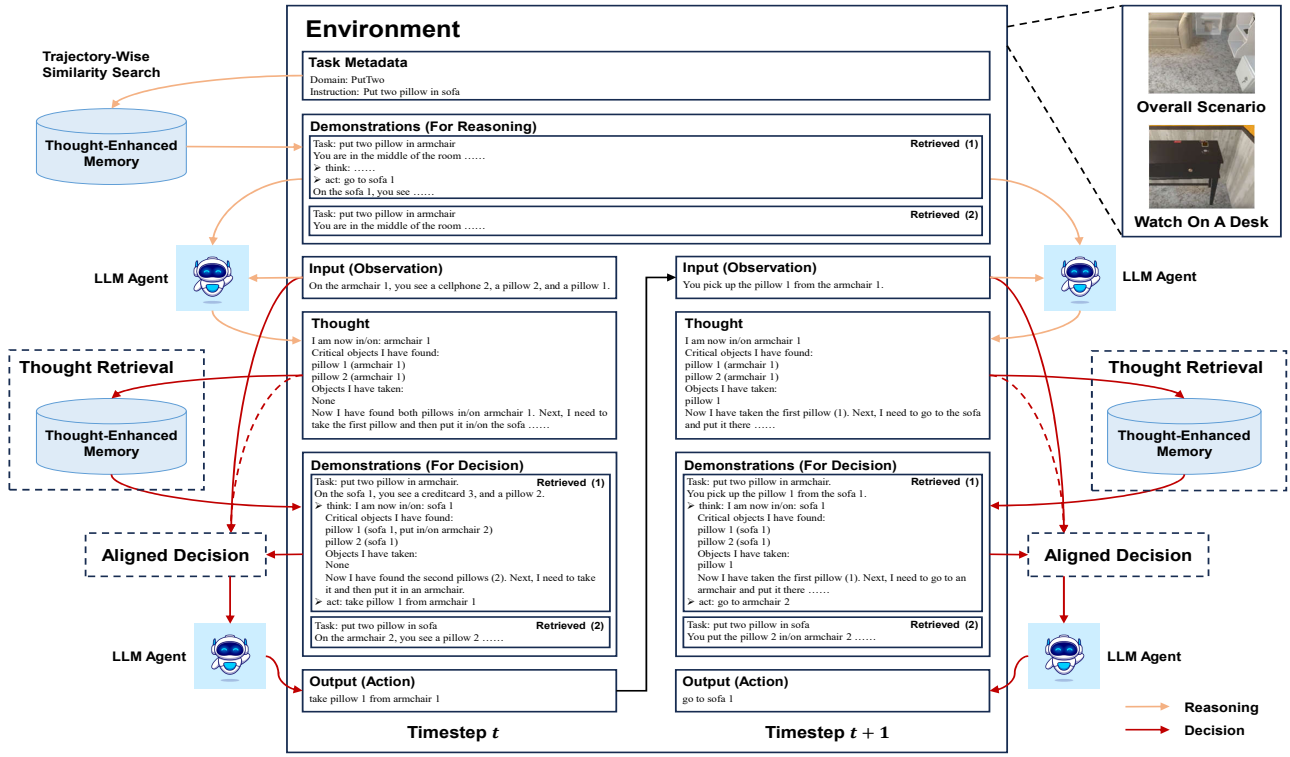


Figure 1: An overall illustration of *TRAD* agent (on ALFWorld [30] enviroment). *TRAD* first pre-processes expert trajectories, labeling each step with high-quality thoughts. At inference time, *TRAD* first conducts *thought retrieval*, which generates thought with trajectory-wise retrieved demonstrations as the query and keys for a more precise step-wise demonstration retrieval. Given the retrieved steps, TRAD employs *aligned decision* module to complement their temporally neighboring steps and corresponding position information (Fig. 2). Finally, the next action is generated according to the enhanced demonstration.

Context limit of LLMs. When facing tasks with long horizons and complex observations, prompting with complete trajectories will result in input sequences longer than the allowed length of LLMs. *Synapse* thus has to reduce the number of trajectory examples or even fail to complete the task directly. Though some long-context LLMs can receive very long prompts, the performance can be harmed due to the issue of long-term forgetting [31].

Irrelevant information in prompts. LLMs are found sensitive to their prompts, and can easily copy their recent input [11, 22]. The decision at the current timestep can be related to very few steps in a retrieved trajectory, while other steps do not provide any helpful information. Therefore, irrelevant steps will have unpredictable effects on the decision of LLM agents. As shown by our experiments, they negatively impact the performance most of the time.

To address the problems of trajectory-wise retrieval and prompting, we delve into step-wise demonstration retrieval and prompting. We discover that, via demonstrating with relevant steps, the input context of the LLM agent can be significantly reduced. Thus, the issue of context limit and irrelevant information can be alleviated. Therefore, the critical part is to retrieve step demonstrations that are truly relevant and helpful. To achieve this, we utilize step-by-step reasoning, i.e. *Chain-of-Thought* technique [38], to abstract the state at each timestep as retrieval queries and keys. The generated *thoughts* can involve historical information or future plans, which is more specific with state transitions and helpful in reducing plausible examples.

In this paper, we propose ***T**hought **R**etrieval* and ***A**ligned **D**ecision* (*TRAD*), a novel framework that achieves step-wise demonstration retrieval via thought matching and enhances the context for action prediction with temporally neighboring steps and their order information. Our contribution can be summarized in four-folds:

- We propose a *thought retrieval* method, where we label thoughts for expert demonstration steps in advance with an LLM, prompt LLM agents to reason at inference time, and achieve step-wise retrieval by a similarity search on thought. To the best of our knowledge, this is the first work that enables the LLM agent with thought retrieval techniques for sequential decision-making.
- Based on the thought retrieval operation, we further propose an *aligned decision* method, where we supply the retrieved steps with their temporal neighbors to overcome imperfect thoughts and enhance task-relevant information.
- We conduct extensive experiments and analysis on Mind2Web [4] tasks and ALFWorld [30], showing that TRAD achieves state-of-the-art (SoTA) performance compared to existing works. *TRAD* brings a 2.99% improvement over the strongest baseline (93.78% $\rightarrow$ 96.77%) to the success rate (SR) on ALFWorld. On Mind2Web, *TRAD* improves element accuracy, step SR, and SR remarkably over the powerful *Synapse* agent [46] by 2.1%, 1.4%, and 0.5%.

- We have deployed TRAD to the real-world robotic process automation scenarios of a global business insurance company, where *TRAD* enables the LLM agent to significantly improve the success rate in a bunch of practical tasks. In average, *TRAD* raises step SR from 90.2% to 98.1% and SR from 65.0% to 92.5%.

2 RELATED WORK

2.1 LLM Agents

In recent years, there has been a rapidly growing trend to utilize pre-trained LLMs as the central controller to obtain human-level decision-making capabilities [35]. Among these works: Nakano et al. [18] fine-tune the GPT-3 [3] model for question answering in a text-based web browsing environment. Yao et al. [40] develop WebShop, a simulated e-commerce website environment, and fine-tune a BERT [5] model with imitation learning and reinforcement learning. Yao et al. [42] insert a reasoning section between observation input and action output, significantly improving the performance on ALFWorld [30] and WebShop [40] tasks. Shinn et al. [28] further improve over [42] via verbally reflecting on linguistic task feedback signals. Schick et al. [26] teach LLMs to use external tools via simple APIs in a self-supervised learning way. Park et al. [21] introduce *Generative Agents*, extending LLMs with natural language memories and retrieving them dynamically to plan behavior. Wang et al. [37] propose *DEPS*, an interactive planning approach, which facilitates better error correction by integrating a description of the plan execution process and an explanation of failure feedback. Wang et al. [34] employ an exploration curriculum, a growing skill library, and a novel iterative prompting mechanism, leading to better proficiency in playing Minecraft. Deng et al. [4] construct the Mind2Web dataset from real-world webpages, which consists of three subsets requiring different degrees of generalization, and compare the performance of imitation learning and few-shot inference.

As can be seen above, most existing LLM agents focus on: 1) improving task performance by direct fine-tuning [4, 18, 40]; 2) enhancing planning or reasoning by explicitly prompting [28, 37, 42]; 3) extending the application with an external memory or tool library [21, 26, 34]. However, providing more relevant information in prompts, as a fundamental way to elicit better task understanding, does not receive sufficient attention. When near-optimal demonstrations are accessible, selecting few-shot demonstrations properly can be a simple yet very effective way to improve task performance, which is investigated in our work.

2.2 In-Context Example Selection

LLMs have been shown excellence of few-shot learning [3], and the selection of in-context examples can yield a significant improvement on the overall performance. Liu et al. [16] first propose to retrieve the k-nearest neighbors (k-NN) of the input as in-context examples, and achieve improvement over random retrieval baselines. Rubin et al. [25] select relevant samples with an encoder trained with label similarity, and obtain better performance over BM25 and pre-trained encoder baselines. Zhang et al. [44] consider selecting and labeling unlabeled examples as demonstrations to achieve the best performance, and view this problem as a sequential decision making task to solve by reinforcement learning. Wu et al. [39] further select examples in a subset recalled from k-NN search via minimizing the entropy of output.

IRCoT [33] should be the most relevant work to ours, which retrieves relevant documents with reasoning steps on question-answering tasks. However, their method consists of retrieving with a complete historical trajectory and accumulating retrieved trajectories over time, which are not transferable to complex sequential decision-making tasks, and we propose a method different from theirs in that: (i) Our method focuses on both providing more relevant demonstrations and reducing irrelevant context for sequential decision-making tasks, while theirs is limited to question-answering tasks and only addresses the first issue. (ii) Our method retrieves completely different steps across timesteps and complements the retrieval results with temporal information, while theirs only accumulates relevant documents at every reasoning step and heuristically cuts off the earliest ones to fit in the context limit of LLMs. (iii) Our method prepares pseudo-golden thoughts for expert trajectories in the memory to enable retrieval with trajectories without thoughts, and utilizes single-step thoughts as both queries and keys for precise retrieval, while theirs uses thoughts only as queries with raw documents as keys.

The selection of in-context examples has been studied thoroughly for non-sequential tasks like question answering and sentiment analysis. However, for sequential decision-making tasks, how to select the examples to improve the overall performance remains unclear. Zheng et al. [46] propose a trajectory-wise retrieval solution, while a more precise step-wise solution is still desired as discussed in Section 1, which motivates our work.

2.3 LLM Planning and Reasoning

Our work proposes to use thought, which can be viewed as a general abstraction of the current state, as queries and keys for retrieval. Nevertheless, plans, code comments, and any other text that extracts comprehensive information about the current state can serve as an alternative. Therefore, we particularly review some remarkable reasoning and planning works based on LLMs, and most of them are complementary to our work.

Wei et al. [38] first introduce the concept of *Chain-of-Thought* (CoT) by providing with explicit step-by-step reasoning process in example outputs improving performance on arithmetic, commonsense, and symbolic reasoning tasks. Wang et al. [36] further find that a single reasoning path can be sub-optimal, and propose *self-consistency* to address this problem by sampling multiple reasoning paths. For efficient yet flexible search of reasoning paths, Yao et al. [41] apply tree search with self-evaluation to find globally excellent thoughts. Besta et al. [2] later extend the tree-search structure to a graph search for even better flexibility and overall performance.

The works mentioned above consider problems that are non-sequential or solvable by a single complete reasoning path after receiving the input. For harder sequential decision-making problems: Zhou et al. [47] introduce *least-to-most* prompting to solve hard problems by decomposing the problem and solving sub-problems sequentially. *ReAct* proposed by Yao et al. [42] interacts with the environment in a reason-then-act style, which enriches the context for action prediction. *Code-as-Policies* [14] writes executable codes

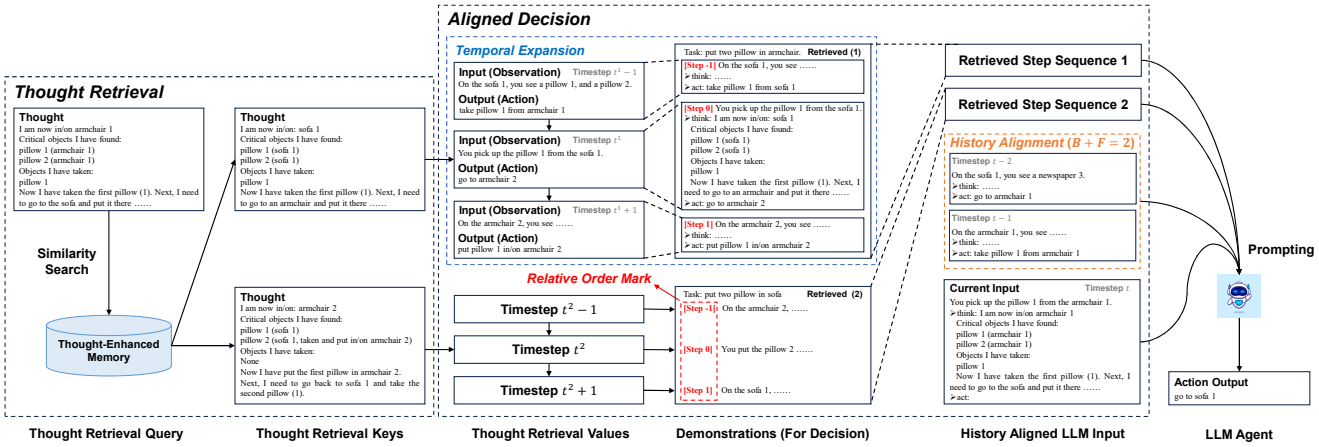


Figure 2: An illustration of our *aligned decision* method, where $B = F = 1$ and the i-th retrieved step is at time t^i in its trajectory. The aligned decision method consists of three sub-processes to the retrieved step demonstrations and prompting: 1) Temporal Expansion: Collect at most B previous steps and F subsequent steps for each retrieved step, and transform each step into a sequence of length $B + F + 1$ from $t^i - B$ to $t^i + F$; 2) Relative Order Mark: For each step in one demonstration step sequence, we label its relative position to the retrieved step in this sequence, i.e., the previous one ($t^i - 1$) with `[Step -1]` and the next one ($t^i + 1$) with `[Step 1]`; 3) History Alignment: For the current episode, we complement current observation (and thought, optional) with $B + F$ previous steps to enrich information and align with demonstrations.

for embodied control by hierarchically expanding undefined programs, which can be viewed as implicit reasoning or CoT process. Liu et al. [15] propose to incorporate the strength of classical planners by translating the original problem into a PDDL [1] problem to solve by classical planners. Hao et al. [10] and Ding et al. [6] share a similar insight that reasoning can be implemented indeed by planning, where [10] use LLMs as world models and [6] conduct MCTS for thought generation with a light-weight extra network.

To summarize, LLM planning and reasoning have continuously received huge attention from researchers in recent years. This makes our work flexible and improvable with more powerful planning and reasoning methods in the future.

3 THE TRAD FRAMEWORK

As discussed in Section 1, trajectory-wise retrieving and prompting lead to issues of plausible examples, LLM context limits, and irrelevant information. To resolve these issues, we propose a novel method called ***T**hought **R**etrieval* and ***A**ligned **D**ecision* (*TRAD*), as illustrated in Fig. 1. Our *TRAD* agent utilizes thought, which is obtained by reasoning about its current state, to retrieve similar steps from expert trajectories, and is then complemented with steps temporally correlated to the retrieved ones and their temporal position information to predict the action. Formally, our *TRAD* agent can be summarized in one equation:

$$\pi_{TRAD}(a_t|\xi, o_{0:t}, a_{0:t-1}) = \text{LLM}(\text{AD}(\text{TR}(\tau_t, \mathcal{M}), \xi, o_{0:t}, a_{0:t-1})) \,,$$

where ξ is the current task, $o_{0:t}$ and $a_{0:t-1}$ are historical observations and actions, τ_t is the thought generated by LLM about the current state, TR and AD denote our *thought retrieval* and *aligned decision* modules, and $\mathcal{M}$ refers to the thought-enhanced memory. We will present each module of *TRAD* in the following subsections.

3.1 Thought Preparation

Most expert trajectories, collected by either human or other expert agents, do not contain their reasoning process. Therefore, before we utilize thoughts for retrieval, we should prepare thoughts for each demonstration step in the memory. Specifically, we start from a small subset of expert demonstrations and provide thoughts written by human experts for each step in it. Given this small subset as few-shot examples in prompts, we can query LLMs to label thoughts for a large memory. Although ground-truth actions are not accessible at inference time, we can prompt LLMs with them to generate thoughts of higher quality. In this way, LLMs produce pseudo-golden thoughts consistent with expert actions, and we obtain a *thought-enhanced memory* $\mathcal{M}$ supporting both trajectory-wise retrieval with task meta-data and step-wise retrieval with thoughts.

3.2 Thought Retrieval

Given pseudo-golden thoughts for all steps in the memory, which can serve as keys for step-wise similarity search, we now present our *thought retrieval* method to select relevant demonstrations at inference time. To be specific, we first conduct trajectory-wise demonstration retrieval as in [46] for thought generation. With these trajectory demonstrations, at each timestep t we prompt the LLM to generate a thought τ_t for step-wise retrieval. Note that this process does not directly effects decision-making, hence it can be further simplified if necessary and the issues mentioned in Section 1 will not impact the agent severely.

With the thought τ_t, which can be viewed as an abstraction, about current state, we conduct dense retrieval to find relevant steps in the *thought-enhance memory* $\mathcal{M}$. Here any encoder pre-trained on a large corpus for retrieval, e.g., Sentence-BERT [23] and DPR [12], can be utilized to encode the query thought and key thoughts into dense vectors. Using a cosine similarity between the query and keys, we then collect top-K relevant steps that belong to mutually different trajectories and their corresponding task instructions.

Table 1: Success Rate of Different Methods on 6 Types of ALFWorld Tasks. We compare *TRAD* with *ReAct* [42], *Synapse* [46], and their strong combination. *TRAD* significantly outperforms all baselines in terms of overall performance, achieves the best performance in 5 out of 6 types of task, and shows a decent performance on Heat task. The improvement of *TRAD* over all baselines on overall performance is statistically significant (measured by student's t-test at $p < 0.05$).

Method	Put	Examine	Clean	Heat	Cool	PutTwo	All
ReAct (Random)	0.8472±0.0393	0.8333±0.0454	0.9570±0.0304	0.8841±0.0205	0.9841±0.0224	0.8431±0.0277	0.8980±0.0093
ReAct (Fixed)	0.7778±0.0708	**0.9630±0.0262**	0.9032±0.0263	**0.9275±0.0205**	**1.0000±0.0000**	0.8824±0.0480	0.9055±0.0186
Synapse	0.9444±0.0196	0.7037±0.0262	0.9355±0.0000	0.9130±0.0615	**1.0000±0.0000**	0.8039±0.0555	0.8955±0.0106
Synapse + ReAct	0.9167±0.0340	0.9444±0.0454	**1.0000±0.0000**	0.9130±0.0000	0.9524±0.0000	0.8627±0.0555	0.9378±0.0035
TRAD (Ours)	**0.9583±0.0000**	**0.9630±0.0524**	**1.0000±0.0000**	0.8986±0.0205	**1.0000±0.0000**	**0.9804±0.0277**	**0.9677±0.0141**

3.3 Aligned Decision

Now we have relevant demonstration steps from *thought retrieval*. However, the query thought can be imperfect due to the lack of expert action information at inference time. As we will show by ablation experiments in Section 4.4, directly using these steps to form single-step demonstrations does not provide satisfactory performance, which is similar to the plausible example issue of trajectory-wise retrieval. Therefore, we propose an *aligned decision* method to incorporate more information during the decision-making process. *Aligned decision* complements LLM agents with steps temporally correlated to the retrieved ones and their temporal position information. As illustrated in Fig. 2, the *aligned decision* method can be decomposed into following three sub-processes.

Temporal expansion. For each retrieved step, we first expand it into a step sequence involving B previous steps and F subsequent steps. When the number of previous or subsequent steps is smaller than B or F, we simply take all previous or subsequent steps. This transforms each retrieved step into at most $(B + 1 + F)$ temporally successive steps, allowing LLM agents to correct their imperfect thoughts by looking at more related steps at decision-making time.

Relative order mark. Given K expanded step sequences by *temporal expansion*, we insert a mark for each step (including the retrieved ones) indicating the relative position w.r.t. its corresponding retrieved step, and incorporate this rule of mark in the prompt for decision. For example, the last step before the retrieved one will be marked as `[Step -1]`, the retrieved step as `[Step 0]`, and the first step after the retrieved one as `[Step 1]`. This provides temporal information about the $(B + 1 + F) \times K$ demonstration steps, and promotes more accurate demonstration following.

History alignment. Sometimes the optimal policy to a task, like ALFWorld, can be history-dependent, hence using single-step input for action prediction is unreasonable. Since we aim to reduce input content for less forgetting and noise, we should neither use all historical observations and actions. Moreover, even if we include previous actions as auxiliary information, there exists a mismatch where expert demonstrations are given as sequences of length $B + 1 + F$ while current input is a single step. We thus propose to insert at most $B + F$ previous input-output pairs (i.e. $o_{t-(B+F):t-1}$, $a_{t-(B+F):t-1}$) before current input o_t, transforming current input into a similar sequence to demonstrations.

4 EXPERIMENTS

In this section, we aim to study the following research questions:

- **RQ1** How does *TRAD* perform against existing SoTA methods?
- **RQ2** Does *thought retrieval* help to reduce irrelevant context and improve the overall performance?
- **RQ3** Does *aligned decision* help to supply information when generalization is important?
- **RQ4** Diving into *aligned decision*, are all *temporal expansion* (TE), *relative order mark* (ROM), and *history alignment* (HA) necessary for improvement?
- **RQ5** How will the performance and advantage of *TRAD* be effected by critical hyper-parameters?

4.1 Experiment Setup

To answer the above research questions, we conduct extensive experiments on ALFWorld [30] and Mind2Web [4] tasks. For each task, we introduce the details of evaluation as follows.

ALFWorld [30] is a text-based game aligned with ALFRED [29] benchmark. It involves 6 types of tasks where an agent must take a series of actions (e.g. *go to shelf 1*, *take vase 2 from shelf 1*, *put vase 2 in/on cabinet 5*) to achieve a high-level goal given by a natural language instruction (e.g. *put some vase on a cabinet*). This environment is challenging in three aspects: 1) Agent should determine likely places of a householding object and explore them one by one to find such object; 2) Agent should understand the usage of some objects like microwaves, fridges, and desklamps; 3) Some tasks can take an agent more than 30 steps to solve, requiring substantial long-term memorization.

Following Shridhar et al. [30], we evaluate on the subset of 134 out-of-distribution tasks, comparing the task success rates of *TRAD* to *ReAct* [42] and *Synapse* [46] (without state abstraction as observations are short). As *ReAct* and *Synapse* has provided sufficiently strong performances, we do not include more complex reasoning and planning baselines and corresponding variants of *TRAD* due to our API cost limit. Note that the original *ReAct* uses fixed but not retrieved trajectories as demonstrations, hence we test two *ReAct* baselines to eliminate such an effect:

- *ReAct* (Fixed) uses fixed human-written trajectories as demonstrations;
- *ReAct* (Random) randomly samples trajectories from the memory as demonstrations.

For fair comparison, *TRAD* uses thoughts in exactly the same format as *ReAct*, and shares a consistent memory of expert trajectories with *Synapse*. We also add a strong baseline (*Synapse+ReAct*)

Table 2: Results (%) of all methods on Mind2Web benchmark. *TRAD* achieves the best overall performances and the most improvement on the two harder subsets, especially the most out-of-distribution Cross-Domain subset. The improvement of *TRAD* over all baselines on three overall metrics is statistically significant (measured by student's t-test with $p < 0.01$).

Method	Cross-Task			Cross-Website			Cross-Domain			All		
	Ele. Acc	Step SR	SR	Ele. Acc	Step SR	SR	Ele. Acc	Step SR	SR	Ele. Acc	Step SR	SR
MindAct	20.3	17.4	0.8	19.3	16.2	0.6	21.0	18.6	1.0	20.6	18.0	0.9
ReAct (Random)	31.0	24.7	1.6	25.7	19.1	0.6	27.9	22.9	1.8	28.3	22.7	1.6
ReAct (Relevant)	31.3	26.0	1.2	26.7	20.5	0.6	28.0	23.1	1.6	28.5	23.4	1.4
Synapse w/o Retrieval	33.1	28.9	3.2	27.8	22.1	**1.1**	30.0	26.5	1.4	30.4	26.4	1.7
Synapse	34.4	30.6	2.0	28.8	23.4	**1.1**	29.4	25.9	1.6	30.4	26.6	1.6
TRAD (Ours)	**35.2**	**30.8**	**3.6**	**30.4**	**24.0**	0.6	**32.0**	**28.0**	**2.0**	**32.5**	**28.0**	**2.1**

combining the trajectory-level retrieval in *Synapse* and the reasoning in *ReAct*. On ALFWorld, all methods are built with GPT-4 [19] and 2 in-context examples.

Mind2Web [4] is an HTML-based web navigation benchmark collected from real-world webpages, involving various tasks such as searching, trip booking, social network subscription, etc. It contains 3 subsets, i.e., cross-task, cross-website, cross-domain. This environment is challenging in two aspects: 1) Existing LLM agents can hardly understand HTML input well; 2) Unseen tasks and websites can require substantial generalization. Deng et al. [4] find that the cross-website and cross-domain subsets are significantly harder due to the need for generalization to unseen websites.

Since Mind2Web was introduced only about half a year ago, there is a lack of suitable baseline algorithms, and thus we compare our *TRAD* agent to *Synapse* [46] and *ReAct* [42]. Following Zheng et al. [46], we evaluate on all 3 subsets, comparing the element accuracy (Ele. Acc), step success rate (Step SR), and trajectory success rate (SR). For fair comparison, we follow [46] and summarize observations into 5 web elements with the pre-trained element ranker provided by [4] for all methods. Since the observations are still very complex on Mind2Web, including thoughts for every step in trajectories is not available, hence: 1) we do not include a *Synapse + ReAct* baseline; 2) *TRAD* generates thoughts and predicts actions by a single-step prompt with the current observation and previous actions (without previous observations). To eliminate the effect of prompting style and reasoning, we build two *ReAct* baselines using the same format of prompt as *TRAD*:

- *ReAct* (Random), for which we prompt *ReAct* with completely random demonstration steps.
- *ReAct* (Relevant), for which we prompt *ReAct* with demonstrate steps randomly chosen from trajectories retrieved by *Synapse*.

We do not include the *ReAct* (Fixed) baseline as it is hard to write or pick demonstrations commonly helpful for such diverse test sets. We also provide the results of the simplest MindAct [4] baseline without reasoning and retrieval for completeness. On Mind2Web, all methods are built with GPT-3.5-turbo and 3 in-context examples.

4.2 Evaluation on ALFWorld

The success rate of each method tested on ALFWorld is shown in Tab. 1. Generally, our *TRAD* agent achieves an average success rate of 96.77%, significantly outperforming *ReAct* (~90%), *Synapse* (89.55%), and even their strong combination (93.78%). It is also worth noting that the worst trial of *TRAD* among 3 random seeds achieves a success rate of 94.8%, outperforming the best trial produced by any other method (94.0%).

Down to the success rate on each type of task, we observe that the success rate of each method varies more on the simplest *Put* task and the hardest *PutTwo* task. We discuss the results of these two tasks respectively as follows:

- On the simplest *Put* task, *ReAct* performs even more poorly than other harder tasks. We find that the two vital reasons for *ReAct*'s failure on *Put* task are incorrect location and usage of objects, e.g. trying to put an object in a closed safe. As this issue can be alleviated through a combination with *Synapse*, the necessity of retrieving relevant demonstrations thus justified.
- *TRAD* achieves the largest improvement on the hardest *PutTwo* task. *PutTwo* requires to correct the locations of two objects and a comprehensive understanding of its task process. Since *TRAD*'s outstanding performance on this hardest task is obtained from a reduced input context at decision-making time, we can conclude that step-wise *thought retrieval* is helpful by reducing the noise of irrelevant steps and finding relevant examples more precisely.

4.3 Evaluation on Mind2Web

To verify the capability of *TRAD* under more realistic scenarios, we compare *TRAD* to *ReAct* and the current SoTA method, *Synapse*, on the Mind2Web benchmark, and the results are shown in Tab. 2. We also include the results of *Synapse* without retrieval here to better illustrate the effect of different retrieval methods.

Generally, *TRAD* achieves the highest performance in terms of all 3 metrics averaged on 3 subsets. Considering that the trajectory-level retrieval of *Synapse* only brings marginal boosts on Cross-Task and Cross-Website subsets, and even slightly impacts the performance on the Cross-Domain subset, our *TRAD* method can be thus justified in two aspects:

- By reducing input context and utilizing step-wise relevant demonstrations, our step-wise *thought retrieval* helps more than the trajectory-wise retrieval with task meta-data in *Synapse* to improve on the simplest Cross-Task subset.
- By eliminating plausible examples and complementing temporal correlated steps, *aligned decision* helps to improve on the two harder subsets, especially the most out-of-distribution Cross-Domain subset.

Furthermore, we observe that the two *ReAct* baselines perform poorly on this task, which indicates that:

Table 3: Results (%) of ablation studies on Mind2Web benchmark. TE builds the basic structure of *aligned decision* and is thus critical for performance boost on all three subsets. HA and ROM work well to promote generalization on the two harder Cross-Website and Cross-Domain subsets but provide little help on the Cross-Task subset. The improvement of *TRAD* over all ablation baselines on Ele. Acc and Step SR is statistically significant (measured by student's t-test with $p < 0.05$).

Method	Cross-Task			Cross-Website			Cross-Domain			All		
	Ele. Acc	Step SR	SR	Ele. Acc	Step SR	SR	Ele. Acc	Step SR	SR	Ele. Acc	Step SR	SR
TRAD w/o TE	34.2	28.4	1.2	27.4	20.4	0.6	29.1	24.0	1.4	30.0	24.5	1.3
TRAD w/o HA	**36.2**	**31.1**	**4.0**	28.3	22.2	0.6	29.4	24.9	1.8	30.8	25.9	**2.1**
TRAD w/o ROM	35.7	30.5	3.6	28.9	22.3	0.6	31.5	27.2	1.9	32.1	27.2	2.0
TRAD (Ours)	35.2	30.8	3.6	**30.4**	**24.0**	0.6	**32.0**	**28.0**	**2.0**	**32.5**	**28.0**	**2.1**

- The thoughts generated by GPT-3.5-turbo on Mind2Web tasks are not sufficient for LLM agents to infer the correct action.
- The single-step prompting style which removes previous observations does not benefit overall performance.

On the contrary, *TRAD* utilizes these imperfect thoughts for retrieval rather than direct decision-making, and is complemented with temporally correlated steps via *aligned decision*. Therefore, *TRAD* is not negatively impacted by the imperfect thoughts, but transforms them into helpful information.

Before we start the study on detailed design and hyper-parameter choices of *TRAD*, we can summarize our performance evaluation on ALFWorld and Mind2Web benchmarks and answer the first three research questions as follows.

Answer to RQ1: On both householding (ALFWorld) and web navigation (Mind2Web) tasks, *TRAD* significantly outperforms curernt SoTA methods and becomes the new SoTA method.

Answer to RQ2: On ALFWorld benchmark, *Synapse + ReAct* generates thoughts in exactly the same way with our *TRAD*, and uses entire relevant trajectories (more information than *TRAD*) as demonstrations for action prediction. However, *TRAD* shows obvious advantage over this baseline. Therefore, we can conclude that *TRAD* benefits from more relevant demonstrations and less irrelevant input context brought by *thought retrieval*.

Answer to RQ3: On Mind2Web benchmark, *TRAD* achieves the most improvement over *Synapse* on the Cross-Domain subset which requires the most generalization. Therefore, we can tell that the *aligned decision* method complements critical information for decision-making on unseen input.

4.4 Ablation Studies

We have verified the effectiveness of *TRAD* on two different scenarios, i.e., automatic householding and web navigation. Next, we are to examine the effect of each module in *TRAD*. Due to our limited budget for API usage, all ablation studies are conducted on the Mind2Web benchmark with GPT-3.5-turbo.

4.4.1 The Effect of Aligned Decision. First, we study the effect of macro building blocks of *TRAD*. Since eliminating *thought retrieval* will disable *aligned decision* at the same time and break the framework fundamentally, we do not remove the *thought retrieval* module, but ablate each component of *aligned decision*, i.e., *temporal expansion* (TE), *relative order mark* (ROM), and *history alignment* (HA), and compare the corresponding performances. The results are shown in Tab. 3.

From Tab. 3, we observe that the performance without each component varies differently on the simplest Cross-Task subset and the two harder subsets:

- On the harder Cross-Website and Cross-Domain subsets, the elimination of all three modules in *aligned decision* results in a significant performance drop, and the effect of *temporal expansion* is the most significant. This is intuitive, since only retrieved steps are provided to the agent without TE, and thus the agent becomes more vulnerable to imperfect thoughts.
- On the simplest Cross-Task subset, however, *history alignment* and *relative order mark* are not that helpful and even cause performance drop. As discussed earlier (Section 1 and Section 3.3), when the issue of plausible examples is not severe, reducing context and prompting with the most relevant demonstration becomes the dominant factor of performance boost. Therefore, only *temporal expansion* remains beneficial for recovering from imperfect thoughts, while the other two components lead to sub-optimal performance.

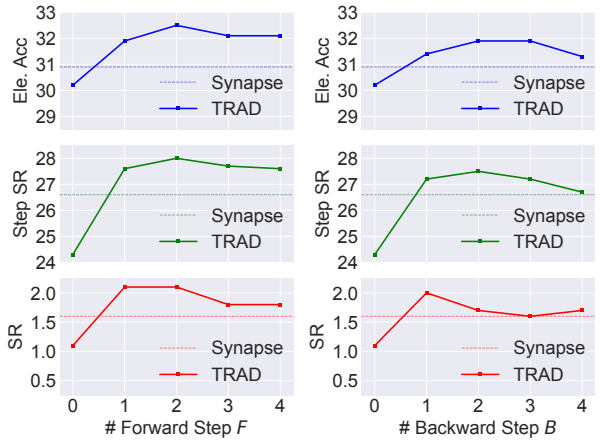
Generally, the *aligned decision* method provides more information about the source trajectories of retrieved steps and the current trajectory, and helps especially for scenarios where generalization is essential. We can now summarize these observations and answer the fourth research question.

Answer to RQ4: Among the sub-processes in *aligned decision*, 1) *temporal expansion* provides tolerance for imperfect thoughts and improves the overall performance of *TRAD* consistently; 2) *relative order mark* and *history alignment* complement *TRAD* with temporal information about the trajectories of retrieved steps and the current trajectory, which serve as useful context for out-of-distribution decision-making but may become less useful for in-distribution decision-making.

4.4.2 The Effect of Expansion Steps B and F. Next we vary a critical hyper-parameter, the number of temporal expansion steps, and investigate how the overall performance will change accordingly. To avoid an expensive grid search on B and F, we consider only one-side expansion by varying B or F from 0 to 4 with the other set to 0. The results over all 3 subsets are shown in Fig. 3.

From Fig. 3, we can have the following observations:

- Both forward expansion ($F > 0$) and backward expansion ($B > 0$) achieve improvement compared to no expansion ($F = B = 0$). This justifies our design of *aligned decision*.



(a) Varying F (b) Varying B

Figure 3: The effect of varying subsequent steps F and previous steps B on Mind2Web benchmark. Solid lines correspond to the performance metrics of *TRAD* given different F and B, and the dashed lines correspond to the *Synapse* baseline. Forward expansion ($F > 0$) generally provides more improvement than backward expansion ($B > 0$) over no expansion ($F = B = 0$) and the *Synapse* baseline. F or B does not help more when they are sufficiently large.

- Either forward expansion or backward expansion does not benefit from increasing a large enough F or B further. This proves our hypothesis that irrelevant context too far from the current state is of little value and even noisy.
- Generally, forward expansion performs better than backward expansion when varying F and B. The reason for this phenomenon might be that historical information has been incorporated in thoughts and thus future information helps more.
- *TRAD* achieves its best performance when $F = 2$ and $B = 0$, and consistently outperforms *Synapse* with forward expansion.

4.4.3 The Effect of Demonstration Amount K. Finally, we look into a common yet important hyper-parameter, the number of retrieved demonstrations K, and see how the advantage of *TRAD* over the baseline (*Synapse*) will change given different $K \in \{1, 2, 3, 4, 5\}$. We show the results over all 3 subsets in Fig. 4. Note that the trajectory-wise prompting in *Synapse* frequently exceeds the context limit when $K = 5$, and thus we omit this result.

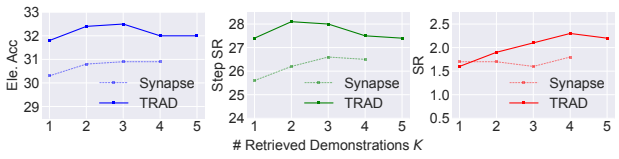


Figure 4: The effect of varying the number of retrieved demonstrations K on Mind2Web benchmark. Solid lines correspond to the performance metrics of *TRAD* given different K, and the dashed lines correspond to the *Synapse* baseline. K has a mild effect on the performance of *TRAD* and *Synapse*, and the advantage of *TRAD* over *Synapse* remains stable when K varies.

From Fig. 4, we see that K has a mild effect on the performance of *TRAD* and *Synapse*, and that the advantage of *TRAD* over *Synapse* consistently remains for all $K \in \{1, 2, 3, 4\}$.

With results in Section 4.4.2 and Section 4.4.3, we now respond to our last research question.

Answer to RQ5: The performance and advantage of *TRAD* generally remains stable with different hyper-parameter choices, i.e., temporal expansion steps, number of retrieved demonstrations. Its performance and advantage only degrade when using long backward extension, which is possibly due to the fact that historical information has already been incorporated in thoughts and does not provide further help for decision-making.

4.5 Case Studies

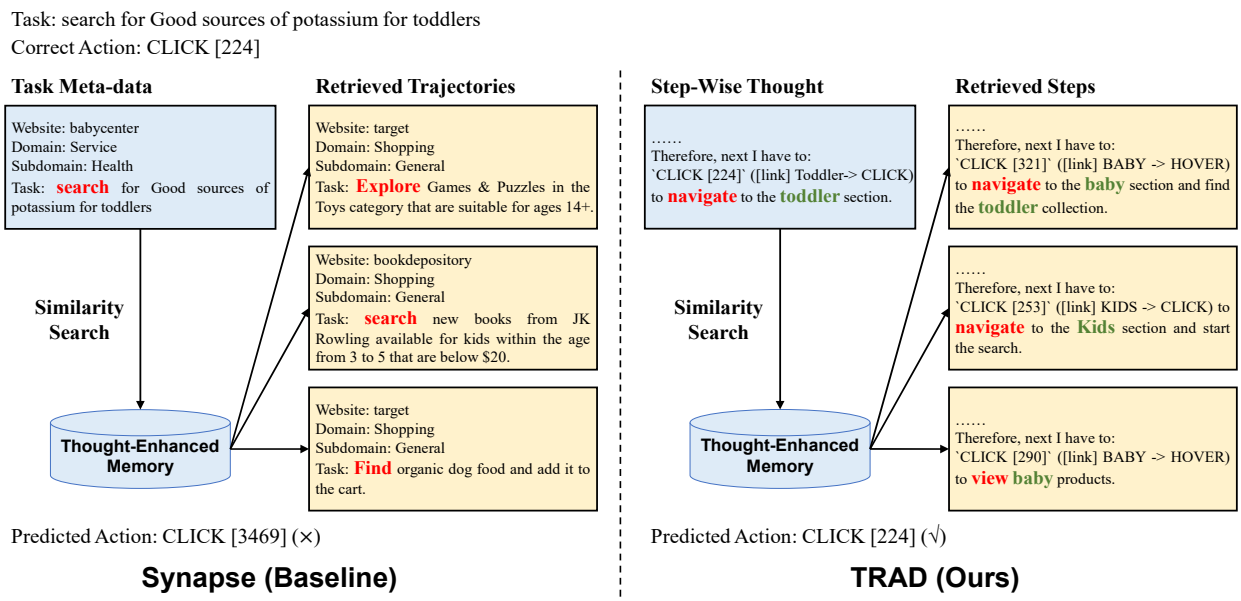
At the end of this section, we present some representative trajectories or steps, where we can intuitively learn the advantages of *TRAD*. We show two cases produced by *Synapse* and our *TRAD* agent on the cross-domain subset of Mind2Web in Fig. 5, to demonstrate: 1) the difference between task meta-data retrieval and *thought retrieval*; 2) the reason for retrieval rather than direct prediction with thought and the tolerance for imperfect thoughts.

In Fig. 5a, the trajectory-wise retrieval of *Synapse* is obviously problematic, which only considers **"search"** in task instructions and the retrieved trajectories are completely irrelevant to the current one. However, when we use these irrelevant demonstrations for thought production and conduct *thought retrieval* afterwards, the retrieved demonstrations become much more relevant as they all relate to **baby (toddler)** and reflect the process of interacting with **navigation** links or buttons to unfold invisible web pages during web browsing. With the demonstrations from *thought retrieval*, *TRAD* is capable of making the correct decision.

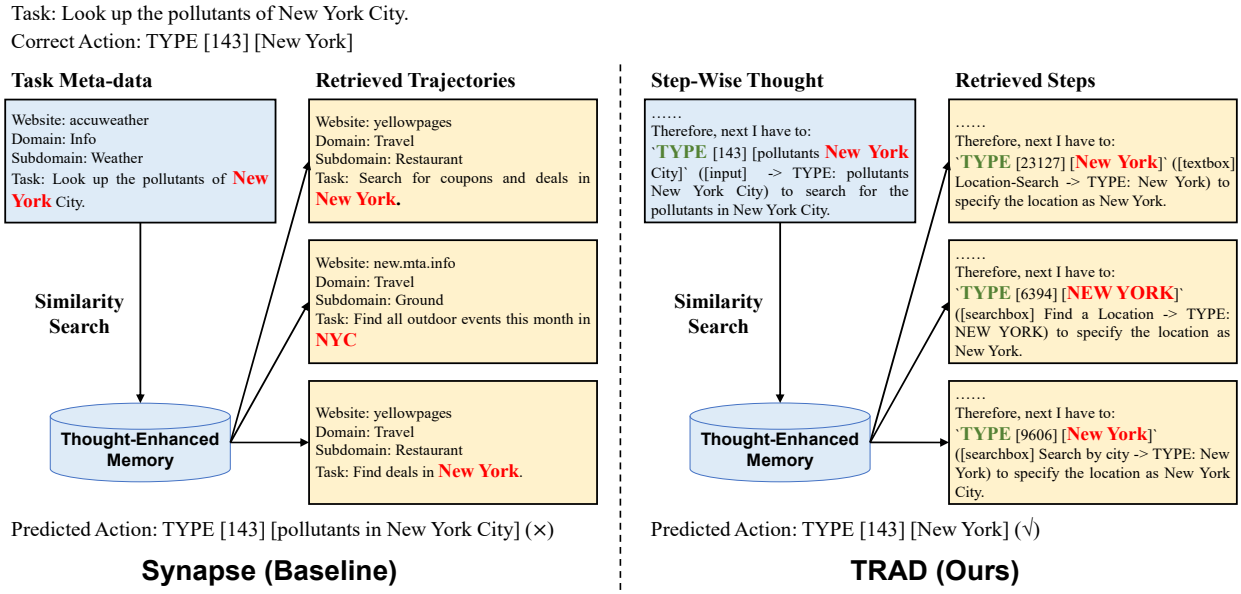
In Fig. 5b, both *Synapse* and *TRAD* seem to retrieve relevant examples trying to find something in **New York**, but if we examine the trajectories retrieved by task meta-data, 2/3 of them fulfill the condition "New York" by clicking some link or button rather than **typing** in a text box. Unfortunately, the correct action under the current state is typing, not clicking, and thus *Synapse* fails to type the correct content. On the contrary, *TRAD* learns to type the correct content "New York" into the text box, even if its thought is incorrect. This also validates our hypothesis that using thought for retrieval instead of prediction helps to correct imperfect thoughts.

5 REAL-WORLD DEPLOYMENT OF TRAD

Since Dec. 2023, we have deployed our *TRAD* agent to automate some real-world office tasks in a mainstream insurance company, which owns a global business with approximately 170 million customers worldwide. We select 4 different websites and collect 100 expert trajectories for some representative tasks on each website as our memory. For evaluation, we collect 20 unseen tasks on each website, using step success rate (Step SR) and trajectory success rate (SR) as evaluation metrics. Tasks involve filling in insurance inquiry forms, implementing advanced information retrieval, etc. Since the websites are complex and contain thousands of web elements, prompting with complete trajectories is not available, hence we only consider single-step prompting with historical actions as auxiliary information.



(a) Representative Case 1



(b) Representative Case 2

Figure 5: Comparison between Synapse trajectory-wise retrieval with task meta-data and TRAD step-wise retrieval with thought. (a) The trajectory-wise retrieval of *Synapse* only considers "search" in task instructions and the retrieved trajectories are completely irrelevant. However, by generating thoughts with these irrelevant trajectories, *thought retrieval* finds more relevant step-wise demonstrations related to baby (toddler) and navigation. (b) The trajectory-wise retrieval of *Synapse* retrieves plausible examples which do not type in a text box with task meta-data. Although thoughts are imperfect, *thought retrieval* finds more relevant demonstrations and *TRAD* learns to input "New York".

To verify the effectiveness of *TRAD*, we use two different *ReAct* agents that the company has attempted as our baseline:

- *ReAct*-RD: randomly selects expert steps **in random trajectories** as demonstrations.
- *ReAct*-RV: randomly selects expert steps **in relevant trajectories** retrieved by task instruction as demonstrations.

To be specific, the difference between *TRAD* and *ReAct*-RV is using thought for a second-time step retrieval and the aligned decision module. To further investigate the effect of *thought retrieval* and *aligned decision*, we also deploy a TR agent which removes our *aligned decision* method, namely the *TRAD* w/o TE baseline in Tab. 3. We list the results in Tab. 4.

Table 4: Evaluation results on real-world websites from a mainstream global business insurance company.

Method		ReAct-RD	ReAct-RV	TR	TRAD (Ours)
Website 1 (form filling)	**Step SR**	0.843	0.826	0.941	**0.950**
	SR	0.500	0.450	**0.800**	**0.800**
Website 2 (advanced IR)	**Step SR**	0.941	0.937	0.958	**0.974**
	SR	**0.900**	0.850	0.850	**0.900**
Website 3 (advanced IR)	**Step SR**	0.962	0.987	**1.000**	**1.000**
	SR	0.850	0.800	0.850	**1.000**
Website 4 (form filling)	**Step SR**	0.820	0.860	0.845	**1.000**
	SR	0.350	0.350	0.400	**1.000**
Average	**Step SR**	0.891	0.902	0.936	**0.981**
	SR	0.650	0.613	0.725	**0.925**

As can be seen in Tab. 4, *TRAD* achieves the best performance on all 4 websites, showing its advantage can remain when deployed to real-world scenarios. Moreover, we observe that *TRAD* w/o TE baseline also outperforms both *ReAct* agents, but exhibits noticeable disadvantages compared to the complete *TRAD* agents. This justifies our design of both *thought retrieval* and *aligned decision*.

Inference efficiency of *TRAD*. At inference time, our *TRAD* agent only introduces little extra time consumption in *thought retrieval* compared to *ReAct*. We profile the inference process of *TRAD* and *ReAct* on all websites and tasks, and in average *TRAD* takes only 11.7% more time than *ReAct*-RD, which indicates that our method achieves improvement without much sacrifice on efficiency.

6 DISCUSSIONS

6.1 Limitations of TRAD

Although *TRAD* exhibits excellent performances over a diverse set of tasks, it still has limitations like dependence on high-quality thought and trade-off between information and noise in *temporal expansion*, and we briefly discuss about them here.

6.1.1 Dependence on high-quality thought. *TRAD* alleviates the issue of imperfect thoughts by its *aligned decision* module, but its capability still depends heavily on the quality of thoughts and the capability of backbone LLM. To make such a step-wise retrieval-augmented method work well, the abstraction of current state is critical since it serves as the query and key for retrieval, hence the LLM used to build a *TRAD* agent should at least have a decent understanding of the task.

6.1.2 Trade-off in temporal expansion. *TRAD* expects to keep relevant information but reduce irrelevant input context by step-wise *thought retrieval*, while preserving some chance for correcting imperfect thoughts by *temporal expansion*. Here exists a trade-off: a longer *temporal expansion* brings not only more tolerance to imperfect thoughts, but also more irrelevant noise in demonstrations. This trade-off requires careful consideration for different tasks.

6.2 Future Directions

While ablation studies have been conducted to justify our design of *TRAD*, there are some promising ideas worth study which can probably improve *TRAD* further. We leave them as future works, and discuss them as follows.

6.2.1 Better Demonstrations For Reasoning. *TRAD* currently employs relevant trajectories or randomly-chosen steps from them as demonstrations to generate thoughts, which still suffers from the issues discussed in Section 1 to some extent. Therefore, modifications can be made to generate thoughts of higher quality, and thus improve the overall performance of *TRAD*.

6.2.2 Better Representations For Retrieval. As we have discussed in Section 2.3, *TRAD* can utilize any other methods to obtain a comprehensive abstraction of the current state in a sequential decision-making task, which can possibly serve as better queries and keys for the step-wise demonstration retrieval. Therefore, *TRAD* can be combined with more powerful LLM planning and reasoning methods and even dense abstractions produced by LLMs pre-trained on domain-specific data like [8].

7 CONCLUSIONS

In this work, we propose a novel LLM agent augmented by step-wise demonstration retrieval (*TRAD*) for sequential decision-making tasks. *TRAD* first retrieves relevant step demonstrations by its thought about current state, and then complements temporally correlated steps for more informative action prediction. Extensive experiments are conducted on two different sequential decision-making tasks to validate the effectiveness of our solution, and thorough ablation studies justify the design choice and stability of our method. We further present the results from real-world deployment of our method, showing its value in real-world applications.

ACKNOWLEDGMENTS

The Shanghai Jiao Tong University team is partially supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62322603, 62076161).

REFERENCES

[1] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. 1998. Pddl| the planning domain definition language. *Technical Report* (1998).

[2] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *arXiv preprint arXiv:2308.09687* (2023).

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Proceedings of the 34th Advances in Neural Information Processing Systems (NeurIPS)*.

[4] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a Generalist Agent for the Web. In *Proceedings of the 37th Advances in Neural Information Processing Systems (NeurIPS)*.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254* (2023).

[7] Fernando Ferraretto, Thiago Laitz, Roberto Lotufo, and Rodrigo Nogueira. 2023. ExaRanker: Synthetic Explanations Improve Neural Rankers. In *Proceedings of*

the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). 2409—-2414.

[8] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A Real-World WebAgent with Planning, Long Context Understanding, and Program Synthesis. In *Proceedings of The 12th International Conference on Learning Representations (ICLR)*.

[9] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. 2023. Understanding HTML with Large Language Models. In *Findings of the Association for Computational Linguistics (EMNLP)*. 2803–2821.

[10] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8154–8173.

[11] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

[12] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.

[13] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language Models can Solve Computer Tasks. In *Proceedings of the 37th Advances in Neural Information Processing Systems (NeurIPS)*.

[14] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as Policies: Language Model Programs for Embodied Control. In *Proceedings of 2023 IEEE International Conference on Robotics and Automation (ICRA)*. 9493–9500.

[15] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477* (2023).

[16] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[17] Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. 2023. Generative Relevance Feedback with Large Language Models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2026–2031.

[18] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[19] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[20] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th Advances in Neural Information Processing Systems (NeurIPS)*. 27730–27744.

[21] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*. 1–22.

[22] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* (2019).

[23] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3980–3990.

[24] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[25] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning To Retrieve Prompts for In-Context Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 2655–2671.

[26] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Proceedings of the 37th Advances in Neural Information Processing Systems (NeurIPS)*.

[27] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of Bits: An Open-Domain Platform for Web-Based Agents. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Vol. 70. 3135–3144.

[28] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of the 37th Advances in Neural Information Processing Systems (NeurIPS)*.

[29] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10737–10746.

[30] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of 9th International Conference on Learning Representations (ICLR)*.

[31] The LongChat Team. 2023. *How Long Can Open-Source LLMs Truly Promise on Context Length?* https://lmsys.org/blog/2023-06-29-longchat/

[32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023).

[33] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 10014–10037.

[34] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).

[35] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432* (2023).

[36] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The 11th International Conference on Learning Representations, (ICLR)*.

[37] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. In *Proceedings of the 37th Advances in Neural Information Processing Systems (NeurIPS)*.

[38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of the 36th Advances in Neural Information Processing Systems (NeurIPS)*.

[39] Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-Adaptive In-Context Learning: An Information Compression Perspective for In-Context Example Selection and Ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 1423–1436.

[40] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Proceedings of 36th Conference on Neural Information Processing Systems (NeurIPS)*.

[41] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Proceedings of 37th Conference on Neural Information Processing Systems (NeurIPS)*.

[42] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of The 11th International Conference on Learning Representations (ICLR)*.

[43] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large Language Models are Versatile Decomposers: Decomposing Evidence and Questions for Table-based Reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 174–184.

[44] Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active Example Selection for In-Context Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 9134–9148.

[45] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Step-Back Prompting Enables Reasoning Via Abstraction in Large Language Models. In *Proceedings of The 12th International Conference on Learning Representations (ICLR)*.

[46] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2024. Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control. In *Proceedings of 12th International Conference on Learning Representations (ICLR)*.

[47] 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 *The 11th International Conference on Learning Representations (ICLR)*.

[48] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107* (2023).

A PROMPT LIBRARY

A.1 Prompts on ALFWorld

ALFWorld includes 6 different types of task, and we only present the prompt for the Put task here.

A.1.1 Thought preparation. We write thoughts for the same demonstration ($Demo 1 and $Demo 2) as the first two in *ReAct* [42] and use them for thought preparation.

```
You are an agent to interact with a household to solve a task. You will be given a task where you need to put an (two) object(s) to a target either directly or after an
 operation. Each time you first think about your current situation, then output an action, and wait for next observation.
Here is your action space:
* go to target: Move to the target, and you will observe what is in/on the target or know it is closed or opened.
* open target: Open the target when it is closed, and you will observe what is in/on the target. Only cabinets, drawers, fridges, safes, and microwaves can be opened.
* take object from target: Take the object from the target when the object is in/on the target. You can take only one object at the same time.
* put object in/on target: Put an object you have taken/picked up in/on the target. You should go to the target in your last action. You can put no matter there are
other objects in/on the target or not.
* clean object with target: Clean an object you have taken/picked up with the target. The target should be a sinkbasin. You should go to the target in your last action.
 You can clean no matter there are other objects in/on the target or not.
* heat object with target: Heat an object you have taken/picked up with the target. The target should be a microwave. You should go to the target in your last action.
You can heat no matter there are other objects in/on the target or not.
* cool object with target: Cool an object you have taken/picked up with the target. The target should be a fridge. You should go to the target in your last action. You
can cool no matter there are other objects in/on the target or not.
* use object: Use the object. The object should be a desklamp. You should be in/on a place where the object appears.
* look: Look around and see what you are facing. Only look when nothing happens.
Here are two examples.
$Demo 1
$Demo 2

Here is the task.
$Input
```

For the Put task we show part of $Demo 1 here.

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a
handtowelholder 2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: put some spraybottle on toilet.
act: go to cabinet 1
think: I am now in/on: the middle of a room
Critical objects I have found:
None
Objects I have taken:
None
To solve the task, I need to find and take a sparybottle, then put it on toilet. First I need to find a spraybottle. A spraybottle is more likely to appear in cabinet
(1-4), countertop (1), toilet (1), sinkbasin (1-2), garbagecan (1). I can check one by one, starting with cabinet 1.
obs: On the cabinet 1, you see a cloth 1, a soapbar 1, a soapbottle 1.
act: go to cabinet 2
think: I am now in/on: cabinet 1
Critical objects I have found:
None
Objects I have taken:
None
In cabinet 1, there is no spraybottle. Next, I need to check cabinet 2.
......
obs: On the toilet 1, you see a soapbottle 2.
act: put spraybottle 2 in/on toilet 1
think: I am now in/on: toilet 1
Critical objects I have found:
spraybottle 2 (cabinet 2)
Objects I have taken:
spraybottle 2
Now I am on the toilet 1. Next, I need to put spraybottle 2 in/on it.
obs: You put the spraybottle 2 in/on the toilet 1.
```

A.1.2 Thought generation.

```
You are an agent to interact with a household to solve a task. You will be given a task where you need to put an (two) object(s) to a target either directly or after an
 operation. Each time you first think about your current situation, then output an action, and wait for next observation.
Here is your action space:
* go to target: Move to the target, and you will observe what is in/on the target or know it is closed or opened.
* open target: Open the target when it is closed, and you will observe what is in/on the target. Only cabinets, drawers, fridges, safes, and microwaves can be opened.
* take object from target: Take the object from the target when the object is in/on the target. You can take only one object at the same time.
* put object in/on target: Put an object you have taken/picked up in/on the target. You should go to the target in your last action. You can put no matter there are
other objects in/on the target or not.
* clean object with target: Clean an object you have taken/picked up with the target. The target should be a sinkbasin. You should go to the target in your last action.
 You can clean no matter there are other objects in/on the target or not.
* heat object with target: Heat an object you have taken/picked up with the target. The target should be a microwave. You should go to the target in your last action.
You can heat no matter there are other objects in/on the target or not.
* cool object with target: Cool an object you have taken/picked up with the target. The target should be a fridge. You should go to the target in your last action. You
can cool no matter there are other objects in/on the target or not.
* use object: Use the object. The object should be a desklamp. You should be in/on a place where the object appears.
* look: Look around and see what you are facing. Only look when nothing happens.
When you cannot find some object, remember to check less possible places that you have not visited, or even impossible places like garbagecan.
Here are two examples.
$Demo 1
$Demo 2

Here is the task.
$Input
```

For the Put task we show part of a typical $Demo here.

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 1, a cabinet 2, a cabinet 3, a cabinet 4, a countertop 1, a garbagecan 1, a
handtowelholder 1, a handtowelholder 2, a sinkbasin 1, a sinkbasin 2, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: put some soapbottle on toilet.
> think: I am now in/on: the middle of a room
Critical objects I have found:
```

```
None
Objects I have taken:
None
To solve the task, I need to find and take a soapbottle, then put it on the toilet. First, I need to find a soapbottle. A soapbottle is more likely to appear in
cabinets (1-4), countertop (1), or sinkbasin (1-2). I will start by checking the garbagecan 1, although it's less likely to find a soapbottle there, it's still possible
.
> act: go to garbagecan 1
On the garbagecan 1, you see a soapbottle 1.
> think: I am now in/on: garbagecan 1
Critical objects I have found:
soapbottle 1 (garbagecan 1)
Objects I have taken:
None
I have found a soapbottle (1) in the garbagecan 1. Next, I need to take it.
> act: take soapbottle 1 from garbagecan 1
You pick up the soapbottle 1 from the garbagecan 1.
> think: I am now in/on: garbagecan 1
Critical objects I have found:
soapbottle 1 (garbagecan 1)
Objects I have taken:
soapbottle 1
Now I have taken the soapbottle (1). Next, I need to go to and put it on the toilet 1.
> act: go to toilet 1
On the toilet 1, you see a cloth 1.
> think: I am now in/on: toilet 1
Critical objects I have found:
soapbottle 1 (garbagecan 1)
Objects I have taken:
soapbottle 1
Now I am at the toilet 1. Next, I need to put the soapbottle 1 on it.
> act: put soapbottle 1 in/on toilet 1
```

A.1.3 Action prediction. For action prediction we use almost the same prompt as thought generation. For *TRAD* we only use $B + 1 + F$ steps in each demonstration, and we add the following sentence to tell LLM the meaning of *relative order mark*:

```
The mark [Step $i] in expert examples indicates a coarse relative position of expert demonstration steps to your situation. For example, [Step -1] means the last step,
[Step 0] means the current step, and [Step 1] means the next step.
```

A.2 Prompts on Mind2Web

On Mind2Web we generally follow prompts in *Synapse* [46].

A.2.1 Thought preparation.

```
You are a large language model trained to navigate the web. You will be given a task, an observation, and your previous actions, and each time you should output the
next action and wait for the next observation. Here is the action space:
1. `CLICK [id]`: Click on an HTML element with its id.
2. `TYPE [id] [value]`: Type a string into the element with the id.
3. `SELECT [id] [value]`: Select a value for an HTML element by its id.
Now you are given some expert demonstrations and reasons for their actions, follow these examples and give your reason for the given action. Note that you should take
all previous actions into reasoning, and not take the current action as what you have done.
$Demo 1
$Demo 2
$Demo 3
$Input
```

We show one demonstration here:

```
Task: Find JetBlue press releases for the year 2020
Trajectory:
obs: `<html> <jb-app> <jb-tab-panel tabpanel> <div> <div combobox> <jb-type-ahead-input> <label> From </label> <input id=908 text columbus port columbus intl apt, /> </
jb-type-ahead-input> </div> <div combobox> <jb-type-ahead-input> <label> To </label> <input id=927 text /> </jb-type-ahead-input> </div> </div> </jb-tab-panel> <jb-
footer contentinfo> <div> <a id=1554> Investor Relations <jb-icon img external link should open in /> </a> <a id=107> Press Room <jb-icon img external link should open
in /> </a> </div> </jb-footer> <div> <textarea text /> <div> 250 characters remaining </div> <button id=1911> Submit </button> </div> </jb-app> </html>`
act: `CLICK [107]` ([link]  Press RoomExternal Link should open in a new windo... -> CLICK)
obs: `<html> <main main> <span> <a id=4386> View all releases </a> <div> <div> <a download> </a> <a id=5509 print> <span> Print Core Overhead Bins </span> </a> </div> <
div> <a download> </a> <a id=6087 print> <span> Print Welcome Kiosk (JFK) </span> </a> </div> <div> <a download> </a> <a id=6614 print> <span> Print Core Overhead Bins
</span> </a> </div> <div> <a download> </a> <a id=7192 print> <span> Print Welcome Kiosk (JFK) </span> </a> </div> </div> </span> </main> </html>`
act: `CLICK [4386]` ([link]  View All Releases \uedbe -> CLICK)
obs: `<html> <main main> <div> <div> <label id=8053> Select year: </label> <select id=7685> <option 2023 true> 2023 </option> <option 2022> 2022 </option> <option 2021>
 2021 </option> <option 2020> 2020 </option> <option 2019> 2019 </option> <option 2018> 2018 </option> <option 2017> 2017 </option> <option 2016> 2016 </option> <option
 2015> 2015 </option> <option 2014> 2014 </option> <option 2013> 2013 </option> <option 2012> 2012 </option> <option 2011> 2011 </option> <option 2010> 2010 </option> <
option 2009> 2009 </option> <option 2008> 2008 </option> <option 2007> 2007 </option> <option 2006> 2006 </option> <option 2005> 2005 </option> <option 2004> 2004 </
option> <option 2003> 2003 </option> <option 2002> 2002 </option> </select> </div> <div> <label> Category: </label> <select id=7691> <option true> All Latest News </
option> <option press-release> Press Releases </option> <option articles> Articles </option> </select> </div> <div> <label> Search: </label> <input id=8124 text
newssearch /> <button id=8126 submit> <span> Search </span> </button> </div> </div> </main> </html>`
act: `SELECT [7685] [2020]` ([combobox]  Select year: -> SELECT: 2020)
reason: I have to find:
JetBlue press releases
for the year 2020
Now I have done:
`CLICK [107]` ([link]  Press RoomExternal Link should open in a new windo... -> CLICK)
`CLICK [4386]` ([link]  View All Releases \uedbe -> CLICK) to show all releases
Therefore, next I have to:
`SELECT [7685] [2020]` ([combobox]  Select year: -> SELECT: 2020) due to the condition `for the year 2020`
```

The input are presented in the same format as demonstrations without human-written reasons.

A.2.2 Thought generation.

```
You are a large language model trained to navigate the web. You will be given a task, an observation, and your previous actions. Each time you should output the next
action and wait for the next observation. Here is the action space:
1. `CLICK [id]`: Click on an HTML element with its id.
2. `TYPE [id] [value]`: Type a string into the element with the id.
```

```
3. `SELECT [id] [value]`: Select a value for an HTML element by its id.
Now you are given some expert demonstrations, follow these examples and conduct reasoning about your situation.
$Demo 1
$Demo 2
$Demo 3
$Input
```

We show part of a typical `$Demo` here.

```
Task: Find cheapest cars available at San Francisco Airport for a day.
Trajectory:
obs: `<html> <main> <div> <label id=132> Pick-up location </label> <input id=372 pick-up location /> </div> <div> <div> <a id=859> <span> San Francisco </span> </a> <a
id=896> <span> San Francisco Airport </span> </a> </div> <button id=1137 button> <div> Airports </div> </button> </div> </main> </html>`
previous actions:

reason: I have to find:
the cheapest cars
available at San Francisco Airport
for a day
Now I have done:
None
Therefore, next I have to:
`CLICK [896]` ([link]  San Francisco Airport -> CLICK) to select the pick-up location as San Francisco Airport.
obs: `<html> <body> <a id=1885> <span> Car rental </span> </a> <div> <div> <label> Pick-up location </label> <input id=1994 pick-up location san francisco airport, us (
sfo) /> </div> <div> <button id=2006 button tue, mar 28> <div> <div> Pick-up date </div> <div> Tue, Mar 28 </div> </div> </button> <button id=2131 button fri, mar 31> <
div> <div> Drop-off date </div> <div> Fri, Mar 31 </div> </div> </button> <button id=2251 button> <div> Search </div> </button> </div> </div> </body> </html>`
previous actions:
`CLICK [896]` ([link]  San Francisco Airport -> CLICK)
reason: I have to find:
the cheapest cars
available at San Francisco Airport
for a day
Now I have done:
`CLICK [896]` ([link]  San Francisco Airport -> CLICK) due to the condition `at San Francisco Airport`
Therefore, next I have to:
`CLICK [2131]` ([div]  Fri, Mar 31 -> CLICK) to select the drop-off date as "Fri, Mar 31"
obs: `<html> <body> <a id=5677> <span> Car rental </span> </a> <div> <div> <label> Pick-up location </label> <input id=5786 pick-up location san francisco airport, us (
sfo) /> </div> <div> <table grid> <td gridcell> <span id=5951 checkbox 29 march 2023> <span> 29 </span> </span> </td> </table> <div> <button id=6248 button fri, mar 31>
 <div> <div> Drop-off date </div> <div> Fri, Mar 31 </div> </div> </button> <select id=6264 dropoff-time> <option 00:00> Midnight </option> <option 00:30> 12:30 AM </
option> <option 01:00> 1:00 AM </option> <option 01:30> 1:30 AM </option> <option 02:00> 2:00 AM </option> <option 02:30> 2:30 AM </option> <option 03:00> 3:00 AM </
option> <option 03:30> 3:30 AM </option> <option 04:00> 4:00 AM </option> <option 04:30> 4:30 AM </option> <option 05:00> 5:00 AM </option> <option 05:30> 5:30 AM </
option> <option 06:00> 6:00 AM </option> <option 06:30> 6:30 AM </option> <option 07:00> 7:00 AM </option> <option 07:30> 7:30 AM </option> <option 08:00> 8:00 AM </
option> <option 08:30> 8:30 AM </option> <option 09:00> 9:00 AM </option> <option 09:30> 9:30 AM </option> <option 10:00 true> 10:00 AM </option> <option 10:30> 10:30
AM </option> <option 11:00> 11:00 AM </option> <option 11:30> 11:30 AM </option> <option 12:00> Noon </option> </select> </div> <button id=6369 button> <div> Search </
div> </button> </div> </div> </body> </html>`
previous actions:
`CLICK [896]` ([link]  San Francisco Airport -> CLICK)
`CLICK [2131]` ([div]  Fri, Mar 31 -> CLICK)
reason: I have to Find cheapest cars available at San Francisco Airport for a day.
```

A.2.3 Action prediction. We use the same sentence as in ALFWorld to tell LLM about the *relative order mark*.

```
You are a large language model trained to navigate the web. You will be given a task, an observation, and your previous actions. Each time you should output the next
action and wait for the next observation. Here is the action space:
1. `CLICK [id]`: Click on an HTML element with its id.
2. `TYPE [id] [value]`: Type a string into the element with the id.
3. `SELECT [id] [value]`: Select a value for an HTML element by its id.
Now you are given some expert demonstrations, follow these demonstrations and make your decision.
The mark [Step $i] indicates a coarse relative position of expert demonstration steps to your situation. For example, [Step -1] means the last step, [Step 0] means the
current step, and [Step 1] means the next step.
Note that you should take all previous actions into reasoning. In your output, the action should be quoted by a pair of '``'.
$Demo 1
$Demo 2
$Demo 3
$Input
```

We show the format of demonstrations here:

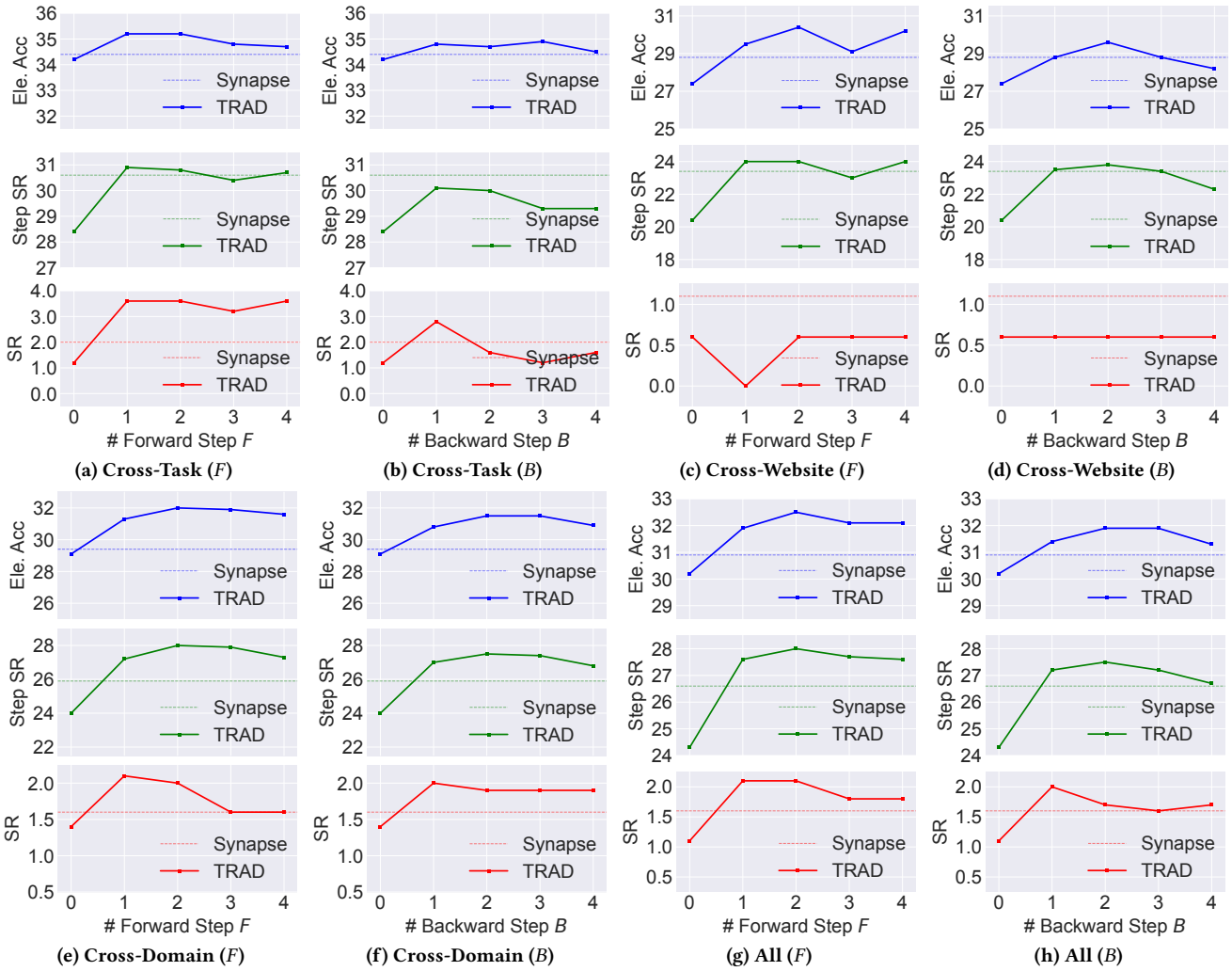
```
Task: Look for a job opening in sales in San Fransisco, and if found, apply for the job.
obs: `<html> <body> <div> <nav navigation> <ul menubar> <li> <button id=8372 menuitem> <span> Research </span> </button> <div menu> </div> </li> </ul> </nav> <nav
cargurus corporate information navigation> <ul menu> <a id=8721 menuitem olink> Our Team </a> <a id=8015 menuitem olink> Careers </a> </ul> </nav> </div> <ul> <a id
=9178 our team> Our Team </a> <a id=9208 careers> Careers </a> </ul> </body> </html>`
previous actions:
`CLICK [117]` ([link]  Our Team -> CLICK)
act: `CLICK [8015]` ([menuitem]  olink -> CLICK)
```

The input are presented in the same format as demonstrations, except that they have no ground-truth actions.

B FULL EXPERIMENT RESULTS

B.1 The Effect of F and B

We list the results of varying subsequent step number F and previous step number B of temporal expansion on each subset and over all 3 subsets of the Mind2Web benchmark in Fig. 6.



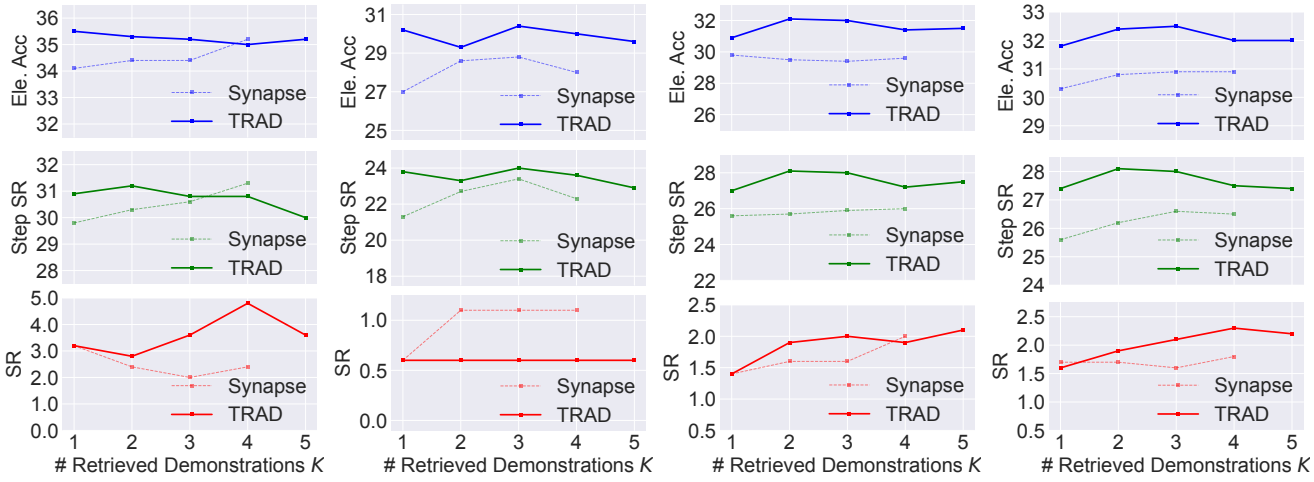
(a) Cross-Task (F) (b) Cross-Task (B) (c) Cross-Website (F) (d) Cross-Website (B)

(e) Cross-Domain (F) (f) Cross-Domain (B) (g) All (F) (h) All (B)

Figure 6: The effect of varying subsequent steps F and previous steps B on Mind2Web benchmark. Solid lines correspond to the performance metrics of *TRAD* given different F and B, and the dashed lines correspond to the *Synapse* baseline. Forward expansion ($F > 0$) generally provides more improvement than backward expansion ($B > 0$) over no expansion ($F = B = 0$) and the *Synapse* baseline. F or B does not help more when they are sufficiently large.

B.2 The Effect of K

We list the results of varying retrieval size K on each subset and over all 3 subsets of the Mind2Web benchmark in Fig. 7.



(a) Cross-Task (b) Cross-Website (c) Cross-Domain (d) All

Figure 7: The effect of varying the number of retrieved demonstrations K on Mind2Web benchmark. Solid lines correspond to the performance metrics of *TRAD* given different K, and the dashed lines correspond to the *Synapse* baseline. K has a mild effect on the performance of *TRAD* and *Synapse*, and the advantage of *TRAD* over *Synapse* remains stable when K varies.