RaDA: Retrieval-augmented Web Agent Planning with LLMs

Minsoo Kim[1] **Victor S. Bursztyn**[2] **Eunyee Koh**[2]
Shunan Guo[2] **Seung-won Hwang**[†1]
[1]Seoul National University [2]Adobe Research
{minsoo9574, seungwonh}@snu.ac.kr
{soaresbu, eunyee, sguo}@adobe.com

Abstract

Agents powered by large language models (LLMs) inherit important limitations, such as the restricted context length, dependency on human-engineered exemplars (e.g., for task decomposition), and insufficient generalization. To address these challenges, we propose RaDA, a novel planning method for Web agents that does not require manual exemplars, efficiently leverages the LLMs' context, and enhances generalization. RaDA disentangles planning into two stages: for a new given task, during Retrieval-augmented Task Decomposition (RaD), it decomposes tasks into high-level subtasks; next, during Retrieval-augmented Action Generation (RaA), it traverses the trajectory obtained with RaD to iteratively synthesize actions based on dynamically retrieved exemplars. We compare RaDA with strong baselines covering a broad space of design choices, using both GPT-3.5 and GPT-4 as backbones; and we find consistent improvements over previous SOTA in two challenging benchmarks, CompWoB and Mind2Web, covering settings with different complexities. We show the contributions of RaDA via ablation studies and qualitative analysis; and we discuss the structural benefits of our more compositional design.

1 Introduction

The adaptability of general-purpose, large language models (LLMs) is a key driver of their growing application across diverse scenarios. Central to this adaptability is in-context-learning (ICL), which enables LLMs to perform new tasks based on *demonstrations*, bypassing the need for a dedicated fine-tuning phase (Brown et al., 2020). This capability is crucial in scenarios involving LLMs deployed as agents in interactive environments, where finetuning LLM-based agents with extensive data across all possible interactions is impractical (Yao et al., 2023; Carta et al., 2023).

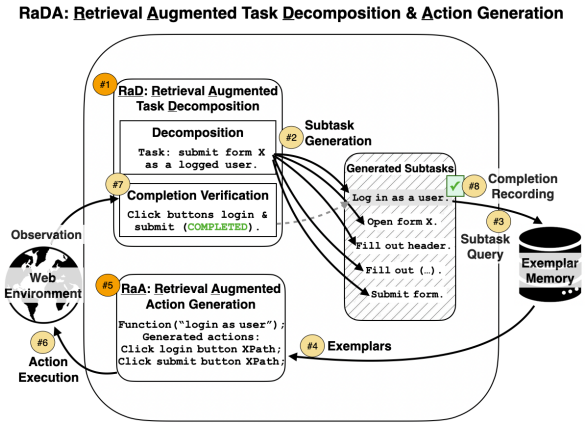

Figure 1: Overview of RaDA: Planning is disentangled as Retrieval-augmented Task Decomposition (RaD, step #1) followed by Retrieval-augmented Action Generation (RaA, step #5). Steps #1-4 show how RaD makes efficient use of exemplar memory to decompose a complex task into subtasks. Steps #5-8 show how RaA grounds each subtask on retrieved exemplars, executes it, verifies its completion, and updates the state of the plan.

However, despite their promise, existing methods of LLM task adaptation face significant limitations. First, ICL-based methods incur a significant cost of prompt engineering. Existing approaches rely on exemplars which must be carefully authored by human annotators, aiming to fully detail the target task. Second, extensive research has shown that LLMs exhibit a high degree of sensitivity to prompt format and the choice of exemplars, making prompt optimization very challenging (Madaan and Yazdanbakhsh, 2022; Yang et al., 2024).

An increasingly prominent line of work proposes the dynamical retrieval of exemplars for ICL conferring two crucial advantages: 1) It automates the prompt engineering process, reducing the cost of exemplar annotation. 2) It aligns exemplars to the task context, enhancing the robustness of exemplar selection. These advantages lead to performance improvements over human-engineered prompts, as

[†]Corresponding author.

Findings of the Association for Computational Linguistics ACL 2024, pages 13511–13525

empirically demonstrated in a wide range of experimental results (Khattab et al., 2022; Luo et al., 2023; Li et al., 2023).

In the context of LLMs-as-agents, the closest approach to ours is Synapse (Zheng et al., 2024b), which retrieves trajectory demonstrations based on semantic retrieval, and leverages them as replacements to human-written prompts. In this work, we consider Synapse as a baseline for pure Retrieval-augmented Action Generation (RaA).

However, RaA such as Synapse seeks to infer the entire action plan for the current task, based on similar entire trajectories of previously seen tasks. This process would fail to generalize when the target task is: 1) **Unseen**, *i.e.*, outside of the corpus of previously seen exemplars, and 2) **Complex**, *i.e.*, compositions of more atomic tasks. To demonstrate these limitations, we will experimentally show how the performance of existing state-of-art approaches strongly degrades in complex or unseen scenarios.

We hypothesize that this failure to generalize can be attributed to the lack of compositional consistency observed in LLMs, regardless of size (Dziri et al., 2023; Zheng et al., 2024a; Chen et al., 2024). That is, LLMs cannot abstract and recombine learned behaviors, such that having LLMs directly transform a task description into an action sequence cannot generalize for unseen and complex tasks. As a solution, we disentangle planning into decomposition into subtasks, then composing retrieved actions in a two-stage process: first, we generate a high-level plan based on a principle of subtask decomposition via Retrieval-augmented Task Decomposition (RaD); second, we perform grounded action generation for each of the subtasks via RaA.

Our approach is closely inspired by two fundamental concepts in human cognitive processing: 1) the ability to disentangle planning and behavior grounding through the use of abstraction (Lachmy et al., 2022), and 2) the efficient reuse of memory through retrieval, to adapt to novel situations through the effective utilization of existing knowledge (Zhao et al., 2022; Sumers et al., 2023).

By leveraging contextually retrieved subtask knowledge in the form of exemplars, RaDA efficiently leverages the LLMs' context in an adaptive manner, while alleviating the need for manually annotated few-shot exemplars. We evaluate our method on two challenging benchmarks for interactive Web environments, covering a range of complex scenarios, while comparing it with strong baselines covering a broad space of design choices. Our contributions can be summarized as follows:

Method	Human Annotation Effort	Input Length	Compositional Generalization
DECOMP	High	Short	✓
RCI	High	Short	✗
SYNAPSE	None	Long	✗
Ours	**None**	**Short**	✓

Table 1: Comparison of related LLM approaches. High human effort indicates requiring manual authoring of optimal few-shot prompts. None indicates no manual authoring effort of exemplars, i.e.) passive demonstrations. Compositional generalization refers to the capability to generalize to novel compositions of existing tasks. Our approach targets compositional generalization, while minimizing human annotation and maximizing context efficiency.

- We propose RaDA, a novel retrieval-augmented planning method for LLMs, which disentangles planning and action grounding to enhance compositional generalization.[1]
- We show that our method improves generalization in challenging scenarios, significantly outperforming competitive baselines on both the CompWoB interactive tasks, and the Mind2Web dataset.

2 Related Work

This section classifies baseline models (Table 1) based on two axes: their target focus and the degree of human supervision involved.

2.1 Decomposition and Planning in LLM

Decomposition Prior literature has explored LLM prompting which aims to induce a modular reasoning structure based on decomposition. Least-to-Most prompting (Zhou et al., 2023) demonstrates that breaking down in subproblems aids generalization in symbolic manipulation, compositional generalization, and math word problems (Cobbe et al., 2021). DecomP (Khot et al., 2023) further enables diverse decomposition structures, including recursion and other non-linear decomposition structures, extending the applicability to long-context multi-hop QA. Wang et al. (2023) show that decompositional prompting is effective for math reasoning even in zero-shot settings, and

[1]Code is available at: https://github.com/ldilab/RaDA

Drozdov et al. (2022) shows that combining decomposed subproblems with dynamic exemplar selection enhances performance on compositional generalization in simple NLU tasks (Keysers et al., 2020).

RaDA tackles **compositional generalization**, by leveraging exemplar retrieval of actions per dynamically defined *subtask*, which allows the agent to flexibly compose grounded plans that can generalize to new scenarios. This is closest to DecomP in Table 1 (column 3), which demonstrates compositional generalization on a multi-hop reading comprehension task. Differently from DecomP, RaDA does not require extensive manual exemplars, and thus can be evaluated on interactive tasks.

Action Generation for LLM Agents LLM have been leveraged as agents in diverse scenarios, for their ability to adapt and ground actions to environment observations, such as robotics (Ahn et al., 2022; Huang et al., 2022). Building on these, ReAct (Yao et al., 2023) proposes to interleave the generation of both reasoning traces and task-specific actions, enabling the LLM to track and update action plans for interactive tasks in virtual environments. An extension of ReAct, Reflexion (Shinn et al., 2023), adds a trial-level memory mechanism, enabling the agent to learn from multiple attempts of the same task, through trial-and-error.

For our target application of LLM-based agents **for Web tasks**, the following baselines have studied the use of LLMs with human-engineered exemplars. RCI (Kim et al., 2023) leverages a process of iterative self-refinement, to enhance grounding by repairing errors through LLM's self-critique. AdaPlanner (Sun et al., 2023) leverages an adaptive planning mechanism, which utilizes few-shot exemplars to revise a plan in response to errors.

RaDA is closest to Synapse (Zheng et al., 2024b) in Table 1 (column 1) as neither requires human-engineered exemplars, instead obtaining exemplars through retrieval, where an exemplar retrieval corpus can be constructed by collecting trajectories with little to no annotation effort. However, Synapse requires the retrieval of whole-trajectory exemplars, which often do not exist in memory; and even when they do, can become too lengthy for LLMs' context. In contrast, RaDA (Table 1, column 2) can leverage partial trajectories via RaD, making more efficient use of the context and allowing for improved compositional generalization. Comparative discussion with respect to human annotation follows in the next section.

2.2 Human Annotation

For supervising the ICL process, early work explored the use of human-engineered, task-specific exemplars (Wei et al., 2022; Liu et al., 2023). However, this approach not only incurs high annotation cost we aim to avoid, but also exhibits various types of sensitivity to the specification of prompts, such as the format of the prompt text (Min et al., 2022; Kojima et al., 2022; Yang et al., 2024; Wei et al., 2023; Madaan and Yazdanbakhsh, 2022), or the order of the exemplars presented in the few-shot prompt (Zhao et al., 2021; Lu et al., 2022),

Similarly, methods such as DecomP can become too costly and ineffective to annotate, as it requires detailed manual demonstrations of problem decomposition and subproblem solution. In contrast, our method disentangles prompts into high-level subtask planning and subtask-level demonstrations, alleviating the need for annotating exemplars that demonstrate subtask composition in detail, and enhancing robustness in compositional generalization. Simultaneously, the same process improves context length efficiency, by removing the need for lengthy exemplars for the overall task.

3 Method

3.1 Compositionality Challenge in Planning

Before describing our method in detail, we first provide a description of standard LLM-based agents, within which we highlight the challenge of compositional generalization.

Language Model Agents Formally, given a task instruction T, an LLM agent π is configured with the prompt P, with the goal of generating a plan Γ for the task, as follows:

$$\Gamma_T = \{a_1, ..., a_t\} \leftarrow \pi(\cdot|P, T) \quad (1)$$

where the plan consists of a sequence of actions to be executed in the environment Env, as

$$s_{t+1}, r_{t+1} = \mathsf{Env}(s_t, a_t) \quad (2)$$

and task success is measured by the episodic reward function $r : S \times A \rightarrow \{0, 1\}$.

Compositional Consistency Existing planning methods which are encapsulated by Eq.1, including RCI and Synapse, amalgamate both overall task planning and the generation of environment-specific actions, as a simultaneous process. This

Algorithm 1 RaDA

Input: task T, environment Env, retriever ϕ, and π_D, π_{Q_T}, π_V, representing LLM-based RaD, RaA, and subtask completion verifier, respectively.

$s \leftarrow$ Env.reset(T)
history $\leftarrow \{\}$
C $\leftarrow \{\}$
$\Gamma \leftarrow \{\}$
$h \leftarrow 5$
while Env is not terminated **do**
  $\Gamma = \{\gamma_1, ..., \gamma_n\} \leftarrow \pi_D(\cdot|\phi(T, E), s, T)$ ▷ Subtask Decomposition (RaD)
  while C $\neq \Gamma$ **do**
    $Q_T \leftarrow \pi_V(s, T, \Gamma, C,$ history$[-h:])$ ▷ Subtask-compositional Query
    $\{a_1, ..., a_{t_{Q_T}}\} \leftarrow \pi_{Q_T}(\cdot|\phi(Q_T, E), s, Q_T,$ history$[-h:])$ ▷ Action generation (RaA)
    for a **in** $\{a_1, ..., a_{t_{Q_T}}\}$ **do**
      $s, r,$ info $\leftarrow$ Env.step(a)
      history $\leftarrow$ history $\cup \{(s, a)\}$
    end for
    Verified $\leftarrow \pi_V(s, T, \Gamma, C,$ history$[-h:])$ ▷ Subtask Completion Verification
    for γ in Verified **do**
      C $\leftarrow$ C $\cup \{\gamma\}$
    end for
  end while
end while

is effective when task T as a whole maps to action sequences in the memory, but would not generalize for tasks that are complex and novel compositions of existing behaviors, unless LLMs can abstract and recombine learned behaviors (or, compositional consistency), which is known to lack in LLMs regardless of model size (Chen et al., 2024). We make consistent observations and replace LLM inference with compositional retrieval to overcome.

3.2 Proposed: Disentangling Planning as RaD+RaA

Our proposed approach aims to tackle this challenge, through a principle of disentangling planning into hierarchical components: 1) A decompositional planner which breaks down a task into a set of high-level subtasks and tracks their completion status, and 2) retrieval-augmented, dynamic subtask functions which operationalize the execution of subtasks, converting them into action sequences.

Rather than inferring planning and action generation for an entire task T from a singular prompt P, we decompose unseen or complex tasks into seen subtasks and compose their efficient and generalization execution.

3.2.1 Compositional Exemplar Memory

Formally, the core component of our method is a nuanced, adaptive retrieval mechanism, with the basic unit of this retrieval process defined as the *subtask*, denoted by γ. A query for subtask retrieval is Q_T, or, *subtask-compositional query*, defined as below.

Given a set of subtasks $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, which decomposes the task T, a subtask-compositional query is formulated as:

$$Q_T = \gamma_1 \oplus \gamma_2 \oplus \ldots \oplus \gamma_m \tag{3}$$

where $\oplus$ denotes concatenation, and γ_j represents the j-th subtask of the m selected subtasks for the query composition. RaDA composes such queries to search its task memory, leveraging the retrieved exemplars to ground its action generation. The formulation in Eq.3 enables effective search for a flexible array of composed task descriptions, ranging from focused, single-subtask queries to complex, multi-subtask compositions, which we further detail in Sec.3.2.3.

For the retrieval model, we leverage dense retrieval using pre-trained sentence embeddings (Karpukhin et al., 2020). Given a query Q, the retriever ϕ produces a ranking over a corpus E of encoded exemplars $\{e_1, e_2, \ldots, e_l\}$ as follows:

$$\{\text{sim}(Q, e) \,|\, e \in E\} \leftarrow \phi(Q, E) \tag{4}$$

Note that in Zheng et al. (2024b), the query Q is predetermined to be the full textual description of task T. Our distinction is leveraging ϕ with a dynamically composed query Q_T, which is adaptive to the current state of the plan.

After ranking, the retriever outputs the top-k exemplars, where k is a hyperparameter set to 3. For the retriever setup, we follow Zheng et al. (2024b). We use cosine similarity as the similarity measure, and leverage pre-trained embeddings to encode the

task description, complemented by additional inputs of state and task metadata for CompWoB and Mind2Web, respectively. The retrieval corpus is constructed by successful trajectories from 48 base tasks of MiniWoB, and for Mind2Web, trajectories in the training set are used.

In RaDA, ϕ is flexibly utilized in two ways: First, with T as the query during the initial decomposition stage (RaD; Eq.6), and second, using the subtask-compositional query Q_T, during the action generation stage (RaA; Eq.7), which we describe next.

3.2.2 RaD: Retrieval-augmented Subtask Decomposition

The first step of RaDA is Retrieval-augmented Subtask Decomposition (RaD), formalized as follows. We define a plan Γ_T as a collection of subtasks, generated by a *decompositional planner* π_D, as:

$$\Gamma_T = \{\gamma_1, ..., \gamma_n\} \leftarrow \pi_D(\cdot|P_{\pi_D}, T) \quad (5)$$

Importantly, the planner's focus is on decomposition, and Γ_T can be ordered or unordered. To construct the decomposition prompt P_{π_D}, we utilize a basic instructional prompt $P^I_{\pi_D}$ [2] in conjunction with contextualization using retrieved exemplars, as:

$$P_{\pi_D} = [P^I_{\pi_D}, \phi(T, E)]. \quad (6)$$

This process leverages the exemplars retrieved using the overall task T as the search query, which facilitates the refinement the planner's subtask generation on grounded knowledge of the task.

Subtask Completion Verification Once the decomposed subtask set Γ is generated, we leverage a subtask completion verifier to track the progression of the subtasks towards completion, maintaining a set C of completed subtasks. The verifier is implemented using an instruction-prompted LLM, which takes the observation and actions of the last k steps, plus the remaining subtasks $\Gamma - C$ as input, and outputs the completed subtasks, if any have been completed by the latest action. If a subtask is completed, γ is added to C.

3.2.3 RaA: Retrieval-augmented Action Generation

Following the subtask decomposition by RaD, the grounded action generation step, RaA, proceeds to adaptively generate the grounded action sequence, based on the current stage of the overall plan.

Our subtask-level formulation makes this process intuitive, through the definition of a *subtask function* π_{Q_T}, as an LLM whose intended functionality is 1) represented by the subtask-compositional query Q_T, and is 2) implemented by the exemplars retrieved via $\phi(Q_T, E)$.

Formally, RaA defines π_{Q_T}, which generates the grounded action sequence as follows[3]:

$$\{a_1, ..., a_{t_{Q_T}}\} \leftarrow \pi_{Q_T}(\cdot|\phi(Q_T, E)) \quad (7)$$

3.2.4 RaDA

We demonstrate the full step-by-step operation of RaDA in Alg.1. Given a task T, RaDA first proceeds with RaD, decomposing the task into the set of subtasks Γ_T (line 7). Next, RaDA iteratively performs subtask-level action generation through RaA, operating over the subtask set. At each iteration, the subtask-compositional query Q_T is reformulated, via the verifier π_V which verifies the completion of subtasks γ (line 15), before generating Q_T (line 9). Q_T can range from a single task, to encompassing all remaining subtasks. RaDA iterates over the subtasks, until all subtasks are completed (line 8).

4 Experiments

4.1 Benchmarks

To validate the effectiveness of RaDA, we target two web task automation environments with varying compositional characteristics. First, we evaluate on CompWoB, where tasks are compositions of other web tasks, and thus task decomposition is likely to lead to a set of clear subtasks. Second, to further test the generality of our approach, we also evaluate on Mind2Web, a challenging dataset of human-written tasks on real websites, where tasks are not explicitly compositional, making decomposition into subtasks more complex.

CompWoB CompWoB (Furuta et al., 2023) is an interactive environment which modifies the widely-studied MiniWoB (Shi et al., 2017) benchmark for web and computer task automation, to test the compositional generalization capability of agents. The tasks involve elemental computer interactions such as clicking, typing, form-filling, handling popup

[2] See Appendix A.2 for the full prompts.

[3] An instructional prompt is also used as in Eq.6, which we omit for notational clarity. See Appendix A.2 for the full prompts.

messages, and login. Furuta et al. (2023) categorizes 65 base tasks from the original MiniWoB (Shi et al., 2017) by task complexity, based on the performance of existing LLM agents, into easy, medium, and hard categories.

The 50 compositional tasks in CompWoB are created by systematically combining 2 to 8 base tasks from the easy category, with different levels of overall task complexity, divided into five categories: two-way tasks (20), three-way tasks (10), n-way tasks (5), transition tasks (5), and easy-medium two-way tasks (10). In n-way tasks, 4 to 8 tasks are combined, and in transition tasks, an explicit page transition is implemented, e.g. transitioning from login form to the email browser. The easy-medium two-way tasks incorporate base tasks from the medium category.

In CompWoB, the model input is the state $s_t \in S$ is the raw HTML, and the output actions a_t are generated for a programmatic action space A consisting of the form, `action_type(ID, text)`, where `action_type` is one of several possible function types, and `ID` indicates a unique identifier of an HTML element, where function types are *click*, *move*, and *type*. The additional `text` input is utilized when the action is *type*. We report the standard metric of the success rate of the agent in accomplishing tasks.

Mind2Web Mind2Web (Deng et al., 2023) is a dataset of crowdsourced human interaction demonstrations on a diverse set of real websites which are dynamic and complex. Mind2Web targets realistic tasks that require sophisticated interactions with websites, often requiring a large number of steps to complete. The tasks cover diver domains in Travel, Shopping, Service, Entertainment, and Information.

Similar to CompWoB, the model input is the state $s_t \in S$ consisting of the raw HTML from the dataset, but necessarily reduced using a retrieval model, into a smaller set of top-ranked candidate HTML elements, to handle the massive scale of real-world webpages. The output actions use the same programmatic action space of the form, `action_type(ID, text)`, where `action_type` in Mind2Web, are *click*, *select*, and *type*.

To focus on the evaluation of RaDA on compositional consistency, we select a subset of the challenging *Cross-Domain* split of Mind2Web, ranked based on compositional task complexity. Task complexity is measured by the number of gold action steps in the solution. We report the following standard metrics used for this task: element accuracy, measuring the correctness of the selected `ID`, operation F1, predicting the correctness of the chosen `action_type`, and the success rate for each step in a task, where success indicates both the element and operation selection are correct. For each task, metrics are averaged over the steps of a task, and the macro average across all tasks is reported.

4.2 Baselines

RCI Recursive Criticism and Improvement (Kim et al., 2023) first samples an initial plan consisting of low-level actions, which is refined once using LLM self-criticism to enhance grounding, Next, before the agent executes each action in the plan, multiple iterations of self-refinement is utilized to improve the grounding of the action. RCI is the SOTA LLM method reported in (Furuta et al., 2023).

Synapse Synapse combines a trajectory-level formulation of planning with retrieved trajectory exemplars, to enhance grounding of the overall plan, based on similar experiences stored in an exemplar memory. For the retrieval memory, exemplars from the 48 base tasks of MiniWoB are used, and in Mind2Web, the same training set trajectories are used as the exemplar memory for retrieval, both as in RaDA.

AdaPlanner We additionally compare with approaches which focus on post-hoc error repair. AdaPlanner (Sun et al., 2023) is a few-shot prompting method which leverages a post-hoc form of plan grounding, by re-planning in response to encountered failures from the environment.

MindAct As the original RCI and AdaPlanner are not designed to be evaluated on Mind2Web, and cannot be easily applied without annotating new few-shot exemplars for the dataset, we compare with the closest equivalent few-shot prompted approach from Deng et al. (2023), as well as Synapse, which is evaluated on Mind2Web. MindAct, the baseline agent from Mind2Web, utilizes a multiple-choice formulation of action generation, performing fine-grained reasoning for grounded action generation using few-shot prompts.

4.3 Models

RaDA Our full model, RaDA, uses both retrieval-augmented plan generation (RaD) and retrieval-augmented subtask grounding (RaA), to enable

CompWoB Setting	Original	Reverse
GPT-3.5 Turbo		
RCI (Kim et al., 2023)	28.7	19.2
AdaPlanner (Sun et al., 2023)	20.6	18.9
Synapse (Zheng et al., 2024b)	25.4	16.0
RaA (ours-ablate RaD)	38.0	23.8
RaD (ours-ablate RaA)	50.0	37.8
RaDA (ours)	***63.6**	***49.0**
GPT-4		
RCI (Kim et al., 2023)	56.0	43.5
AdaPlanner (Sun et al., 2023)	29.5	-
Synapse (Zheng et al., 2024b)	41.4	-
RaDA (ours)	***68.6**	***57.8**

Table 2: Results on the CompWoB environment. The baseline results are from the original CompWoB paper. [4] Note that the GPT-4 results for the reverse task are reported only for RCI in Furuta et al. (2023). Best results are denoted in bold. Asterisk (*) denotes statistically significant (P<.001) improvement over all baselines using a paired t-test.

compositional generalization without manual few-shot exemplars. All LLM modules are implemented using instructional prompts, shown in Appendix A.2, and all subtask functions are defined using retrieved exemplars. In implementation, RaDA utilizes a decomposition trigger to initiate decomposition. In CompWoB, decomposition behavior is always triggered, and in Mind2Web, decomposition behavior is triggered if the task length exceeds the minimum length exemplar by 50%. Finally, in CompWoB, the subtask-compositional query Q_T is directly outputted by π_V as a single subtask, and in Mind2Web, Q_T encompasses all remaining subtasks after verification.

RaA (RaD ablation) On CompWoB, we further evaluate variants of RaDA, to understand the effectiveness of each proposed component. First, we test RaA, which partially ablates RaD, and does not use a subtask-decompositional planner to disentangle subtask-level planning from grounded action generation. In this variant, subtask decomposition is utilized only to generate the fine-grained retrieval queries for exemplars. For a fair comparison, we design RaA such that the overall set of exemplars consumed by RaA and RaDA are identical. The retrieved exemplars are simultaneously concatenated into the input context, such that action grounding can leverage retrieved exemplars, albeit without a disentangled notion of subtask progress.

[4] For each of the WoB tasks we perform multiple runs with different seeds, for a total of 10 runs per task.

	Element Acc.	Operation F1	Step SR
MindAct	21.6	52.8	18.6
Synapse	32.8	46.59	29.33
RaDA	***33.39**	***48.83**	***29.92**

Table 3: Results on the cross-domain generalization setting of Mind2Web datasets (GPT-3.5 Turbo) are reported from Deng et al. (2023); Zheng et al. (2024b).[5] Best results are denoted in bold. Asterisk (*) denotes statistically significant (P<.001) improvement over MindAct and Synapse using a paired t-test.

	Element Acc.	Operation F1	Step SR
GPT-3.5 Turbo			
MindAct	9.76	55.43	8.46
Synapse	24.73	37.78	21.34
RaDA	***30.13**	***58.18**	***26.71**
GPT-4			
RaDA	*47.97	*59.00	*41.24

Table 4: We show SOTA methods suffer performance degradation in *complex subset* of the cross-domain split of Mind2Web, selected by the rank of task complexity. Best results are denoted in bold. Asterisk (*) denotes statistically significant (P<.001) improvement over MindAct and Synapse using a paired t-test.

RaD (RaA ablation) We further evaluate RaD, which ablates RaA, thereby removing the ability to retrieve exemplars based on a dynamically generated query. In this variant, we utilize a static set of few-shot exemplars which replace the dynamically retrieved exemplars from RaDA. We choose 5 exemplars consisting of three easy (click-button, click-link, click-dialog), 1 medium (search-engine), and 1 hard (choose-date) tasks according to the task classification of CompWoB.

5 Results

5.1 Results on CompWoB

In Table 2, we report the results on the CompWoB original and reverse tasks. The latter provides upside down instructions (e.g. do A,B,C → do B and

[5] Due to high API costs, we compute the performance on the full set (912 examples), using the official logs released by (Zheng et al., 2024b), using an oracle decomposition trigger.

C, after doing A), reflecting the inherent ambiguity in real-world web instructions. Experimental results from Furuta et al. (2023) show that these simple permutations significantly degrade the performance of LLM agents.

Compared to strong baselines on CompWoB, RaDA outperforms all compared models significantly, in both the original and reverse settings. We also report the results of the two ablations, RaA, which utilizes subtask-level exemplar retrieval for action generation, but grounds all steps of the task plan simultaneously, and RaD, which utilizes a decompositional planner, but with a fixed set of exemplars. Compared to the $63.6\%/49.0\%$ accuracy of RaDA, both RaA and RaD show much lower accuracies, of $38.0\%/23.8\%$ and $50.0\%/37.8\%$ respectively, indicating that subtask decomposition and dynamic exemplar retrieval are complementary. Collectively, these findings substantiate our hypothesis that RaDA's fusion of high-level planning with retrieval-augmented grounding enhances compositional generalization. While baseline models falter when confronted with the compositional task of CompWoB, particularly evident in the reverse setting, and with GPT-3.5, ours consistently demonstrates robust performance across all scenarios, irrespective of model scale.

Furthermore, we evaluate RaDA using the more powerful GPT-4, and observe that performance scales with more performant LLMs, achieving a new state-of-the-art result for LLMs on CompWoB.

5.2 Results on Mind2Web

In Tables 3 and 4, we report the results on the Mind2Web dataset. Our results on the complex subset of Mind2Web reveal that this subset of Mind2Web poses a challenge for the original Synapse method, as shown by the drop in performance ($32.8\% \rightarrow 24.73\%$ for element accuracy, $29.33\% \rightarrow 21.34\%$ for step success rate). This indicates that even for state-of-the-art models, compositional generalization remains challenging.

In contrast, RaDA shows strong performance on both the full and the complex subset, outperforming Synapse significantly on all metrics. We further report the results on GPT-4, and the results are consistent with CompWoB, indicating that our method scales well with more performant LLMs in diverse scenarios.

	RCI	Synapse	RaA	RaD	RaDA	RaDA (GPT-4)
CompWoB Original						
Two-way	46.9	46.9	46.5	69.0	**86.0**	87.0
Three-way	29.7	25.6	20.0	57.0	**61.0**	57.0
N-way	22.2	7.4	**34.0**	16.0	32.0	38.0
Transition	0.0	0.0	**40.0**	39.6	36.0	68.0
Easy-medium	8.8	3.8	39.0	24.4	**51.0**	59.0
Total	28.7	25.4	38.0	50.0	**63.6**	68.6
CompWoB Reverse						
Two-way	33.6	38.5	31.0	60.5	**78.5**	88.5
Three-way	16.3	0.0	15.0	17.0	**21.0**	32.0
N-way	12.6	0.0	2.0	**30.0**	10.0	18.0
Transition	2.0	0.0	24.0	**34.0**	30.0	36.0
Easy-medium	5.4	0.3	29.0	19.0	**47.0**	53.0
Total	19.2	16.0	23.8	37.8	**49.0**	57.8

Table 5: Analysis of success rate on each compositional task type in original and reverse settings of CompWoB. The columns on the left consist of results with GPT-3.5-turbo, and the best results among these are bolded.

6 Analysis

6.1 CompWob Success Rate by Compositional Task Type

To provide a better understanding of the performance gains of RaDA on CompWoB, we present a breakdown of CompWoB results by the type of compositional task in Table 5. We first observe that on compositional tasks such as n-way, transition, and easy-medium, existing approaches perform extremely poorly, showing the challenge of compositional generalization. In the case of vanilla Synapse, the reverse setting of CompWoB significantly effects performance further, indicating that exemplar retrieval alone does not solve compositional generalization.

In contrast, RaDA achieves a consistent and large improvement over baselines on all compositional task types, indicating that its disentangled structure enables better compositional generalization. Next, compared to the RaA variant, which has the advantage of a longer context length, but ablates the disentangled structure, RaDA performs better on 8 of the 10 tasks across the original and reverse orders, while being comparable on the remaining two. Similarly, compared to RaD, which ablates the dynamic retrieval of exemplars, it performs better on 7 of the 10 tasks. Finally, the GPT-4 results show that RaDA's performance scales consistently with more performant LLMs.

	RaDA	RaA
Avg. tokens ↓ (Exemplars)	1259.8	3483.5
Avg. tokens ↓ (Total)	1969.4	4110.1
Max tokens ↓ (Exemplars)	1891.1	8465.8
Max tokens ↓ (Total)	2722.4	9761.3
Accuracy ↑ (Original)	63.6	38.0
Accuracy ↑ (Reverse)	49.0	23.8

Table 6: Token efficiency per LLM inference call, and model accuracy. For each model, the average and maximum number of tokens, both for the exemplars only and for the entire LLM input, are reported. Arrows indicate optimal directions: fewer tokens (↓) and higher accuracy (↑).

6.2 Context-token Efficiency

To validate the efficiency of RaDA's exemplar retrieval, we measure the context-token efficiency *per LLM inference call*, as a main driver of LLM inference cost is the quadratic scaling of computation with respect to input context length. We compare token efficiency of RaA and RaDA, measuring the per-call average and maximum numbers of: tokens comprising 1) the exemplars and 2) the entire prompt.

As Table 6 shows, an LLM inference call in RaDA requires on average less than half the amount of tokens, which translates to a 4x computational efficiency. Furthermore, the lengthy input context of RaA due to the concatenation of exemplars results in cases that exceed the 4096 context token limit of the baseline turbo model, thus requiring the usage of `gpt-3.5-turbo-16k-0613` (16385 token limit) to accommodate the longer context.

Additionally, since RaDA performs LLM inference call for each subtask, we also compute the average total number of tokens consumed *per task*. These are 4582.31 / 7285.76, for RaA and RaDA, respectively. While the overall tokens seen is about 2x[6], due to the quadratic scaling of transformer computations, RADA is still more computationally efficient due to shorter exemplars in each inference call, while achieving significantly higher performance. These observations lend further support to our claim, that decomposing tasks leads to more efficient and performant in-context learning for LLMs.

[6]Note that both RaA and RaDA are given access to the same overall set of exemplars. The 2x difference arises from a further processing for RaA which removes redundant exemplars, as RaA concatenates all exemplars together.

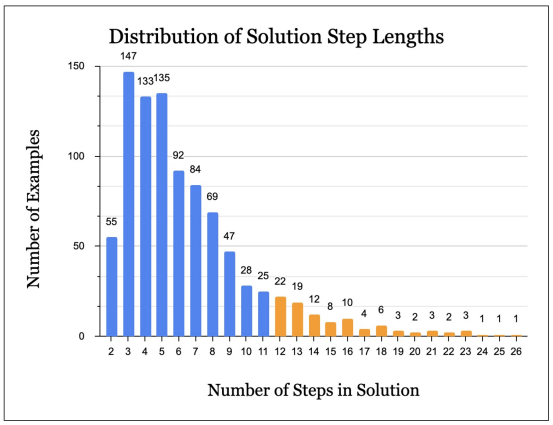

Figure 2: Distribution of action steps required to solve Mind2Web tasks. Higher number of steps indicate higher task complexity, and the gold portion indicates the tasks covered by the complex subset.

6.3 Complexity of Tasks in Mind2Web

In Fig. 2, we show that the distribution of the number of action steps required in the solution, which directly correlate to the complexity of the task (Deng et al., 2023), is skewed towards simpler tasks. For example, the longest task in the cross-domain split requires 26 separate actions to solve correctly, whereas the majority of tasks actually require less than 10 steps, making it likely that a single retrieved exemplar can sufficiently demonstrate a solution. The complex subset in Table.4 corresponds to the gold columns of the figure. Our results demonstrate that state-of-the-art models do not yet generalize well in compositional scenarios, and that RaDA's retrieval-augmented task decomposition and action generation makes meaningful progress towards compositional generalization.

7 Conclusion

We propose RaDA, a planning method which enhances the compositional generalization of LLM agents for Web tasks. RaDA disentangles planning into two components, Retrieval-augmented Task Decomposition (RaD) which decomposes tasks into high-level subtasks, and Retrieval-augmented Action Generation (RaA) which enables the grounded execution of subtasks without the need for manually authored few-shot prompts. We show through experiments on the CompWoB and Mind2Web tasks that RaDA enhances the compositional generalization of LLM agents, significantly outperforming competitive baselines.

Limitations

A limitation of our method is that we rely on a sufficiently good off-the-shelf retrieval model, and our work does not focus on exploring possibilities for using other various types of pre-trained retrievers, of which there are many.

While we did not consider retriever finetuning in this work, future research could consider an overall framework which finetunes the exemplar retriever in tandem with LLM agents.

Another limitation of our method is that we leverage existing trajectories provided by the environments studied, as the corpora for exemplar retrieval. While the cost of collecting trajectories, as opposed to annotating prompt exemplars, is generally much lower, in some tasks human exemplars may be effective in bootstrapping the trajectory collection process. Finally, while leveraging existing exemplars for retrieval is a promising approach, a framework which not only utilizes existing exemplars but collects new ones would improve the applicability of retrieval-based prompting approaches, to more diverse settings.

Acknowledgements

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) [NO. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)] and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) [NO. 2022-0-00077, AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data].

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

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 International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. 2024. Two failures of self-consistency in the multi-step reasoning of LLMs. *Transactions on Machine Learning Research*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

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 *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. 2023. Language model agents suffer from compositional generalization in web automation.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson,

Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick 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)*, pages 6769–6781, Online. Association for Computational Linguistics.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP. *arXiv preprint arXiv:2212.14024*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. 2023. Language models can solve computer tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Royi Lachmy, Valentina Pyatkin, Avshalom Manevich, and Reut Tsarfaty. 2022. Draw Me a Flower: Processing and Grounding Abstraction in Natural Language. *Transactions of the Association for Computational Linguistics*, 10:1341–1356.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4644–4668, Toronto, Canada. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y Zhao. 2023. Dr.icl: Demonstration-retrieved in-context learning.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144. PMLR.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2023. Cognitive architectures for language agents.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023. Adaplanner: Adaptive planning from feedback with language models.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le,

and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023. Larger language models do in-context learning differently.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

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 *The Eleventh International Conference on Learning Representations*.

Wenjia Joyce Zhao, Russell Richie, and Sudeep Bhatia. 2022. Process and content in decisions from memory. *Psychological Review*, 129(1):73.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024a. Step-back prompting enables reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2024b. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*.

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 Eleventh International Conference on Learning Representations*.

A Appendix

A.1 Experiment details

For LLM models, `gpt-3.5-turbo-0613` and `gpt-4-0613` are used through the OpenAI API. For the ablated RaA model, `gpt-3.5-turbo-16k-0613` is used, since it requires a larger input context to concatenate all subtask exemplars. For Mind2Web, we use `gpt-3.5-turbo-16k-0613`, following Zheng et al. (2024b). For all ϕ retrieval models, we use semantic similarity using `text-embedding-ada-002` embeddings, over a retrieval corpus of task descriptions and metadata, following Zheng et al. (2024b).

A.2 Prompts

A.2.1 CompWoB

CompWoB System Prompt

You are a large language model trained to navigate the web. To accomplish the task, use methods in the following Agent class to generate actions until you need the new state to proceed.

```
class Agent:
    def __init__(self, args):
        ...

    # Action: type a string via the keyboard
    def type(self, characters: str) -> None:
        ...

    # Action: click an HTML element with a valid xpath
    def click_xpath(self, xpath: str):
        ...

    # Actions: press a key on the keyboard, including:
    # enter, space, arrowleft, arrowright,
    # backspace, arrowup, arrowdown, command+a,
    # command+c, command+v
    def press(self, key_type: str) -> None:
        ...

    # Action: click an option HTML element in a list with a
    # valid xpath
    def click_option(self, xpath: str):
        ...

    # Action: move mouse cursor on an HTML element with a
    # valid xpath
    def movemouse(self, xpath: str):
        ...
```

RaD Subtask Decomposition

System:
{**system prompt**}

Input:
Here is a demonstration of interactions in the web environment, so you can get a sense of the environment:
{**retrieved exemplars for task**}

Here is the task:
{**task**}

Your goal is to decompose the task into a set of high-level subtasks. For now, do not consider their order, simply list them in the order that they are presented in the task. Start each subtask with [Subtask] and end each subtask with newline. End your generation with EOS.

RaD Subtask Planning

System:
{**system prompt**}

Input:
Here is a demonstration of interactions in the web environment, so you can get a sense of the environment:
{**retrieved exemplars for task**}

Here are the subtasks, in no particular order:
{**subtasks**}

Here is the precise task instruction:
{**task**}

Now, generate the sequence of subtasks in the order that they should be executed to complete the task in the exact specification. Follow the format, 1. [Subtask]. 2. [Subtask]. 3. [Subtask]...
End your generation with EOS.

Subtask Verification and Next Subtask

System:
{**system prompt**}

Input:
Here is the list of subtasks:
{**subtasks**}

Here are the subtasks that have been completed so far:
{**completed subtasks**}

Here are the actions that have been executed so far:
{**action history**}

Write the next subtask that should be executed, in the format, [ID: n] where n is the id of the next subtask.
End your generation with EOS.

RaA Action Generation

System:
{**system prompt**}

Input:
Subtask demonstration:
{**retrieved exemplars for subtask**}

Now, here is the current task and state:
{**trajectory**}

Here are the actions that have been executed so far:
{**action history**}

Generate the action(s) for the following subtask:
{**subtask description**}

A.2.2 Mind2Web

Mind2Web System Prompt

You are a large language model trained to navigate the web. 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.

RaD Subtask Decomposition

System:
{**system prompt**}

Input:
Now, you will be presented with a few trajectories of similar tasks. You can refer to them to understand what structure similar websites have, and what action trajectories are likely to be executable and successful.
{**retrieved exemplars for task**}

Here is the task:
{**task**}

Your goal is to decompose the task into a set of high-level subtasks. For now, do not consider their order, simply list them in the order that they are presented in the task. Start each subtask with [Subtask] and end each subtask with newline. Decompose into at most three subtasks. End your generation with EOS.

RaD Subtask Verification

System:
{**system prompt**}

Input:
Here are the subtasks that have not been completed yet:
{**remaining subtasks**}

Trajectory:
{**trajectory**}

Assuming that the previous action was executed successfully, write the subtask that was completed by the action. If no subtask was completed, write NONE. End your generation with EOS.

RaA Action Generation

System:
{**system prompt**}

Input:
Subtask demonstration:
{**retrieved exemplars for remaining subtasks**}

Subtasks that have not been completed yet are:
{**remaining subtasks**}

Trajectory:
{**trajectory**}