BOLAA: BENCHMARKING AND ORCHESTRATING LLM AUTONOMOUS AGENTS

Zhiwei Liu[*], Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, Ran Xu, Phil Mui[◇], Huan Wang[♦], Caiming Xiong[♦], Silvio Savarese[♦]

Salesforce AI Research, USA
[♦]Corresponding Authors: {huan.wang, cxiong, ssavarese}@salesforce.com

ABSTRACT

The massive successes of large language models (LLMs) encourage the emerging exploration of LLM-based Autonomous Agents (LAAs). An LAA is able to generate actions with its core LLM and interact with environments, which facilitates the ability to resolve complex tasks by conditioning on past interactions such as observations and actions. Since the investigation of LAA is still very recent, limited explorations are available. Therefore, we provide a comprehensive comparison of LAA in terms of both agent architectures and LLM backbones. Additionally, we propose a new strategy to orchestrate multiple LAAs such that each labor LAA focuses on one type of action, *i.e.* BOLAA, where a controller manages the communication among multiple agents. We conduct simulations on both decision-making and multi-step reasoning environments, which comprehensively justify the capacity of LAAs. Our performance results provide quantitative suggestions for designing LAA architectures and the optimal choice of LLMs, as well as the compatibility of both. Original codes[1] are released. We also provide a unified benchmark protocol with AgentLite[2].

1 INTRODUCTION

Recent booming successes of large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023) motivate emerging exploration of employing LLM to tackle various complex tasks (Zhang et al., 2023), amongst which **L**LM-based **A**utonomous **A**gents (LAAs) (Shinn et al., 2023; Madaan et al., 2023b; Huang et al., 2022; Kim et al., 2023; Paul et al., 2023; Yao et al., 2023a) stand with most spotlights. LAA extends the intelligence of LLM to sequential action executions, exhibiting superiority in interacting with environments and resolving complex tasks via collecting observations. To name a few, BabyAGI[3] proposes an AI-powered task management system, which leverages OpenAI LLM[4] to create, prioritize, and execute tasks. AutoGPT[5] is another popular open-source LAA framework that enables the API calling capability of LLMs. AutoGen Wu et al. (2023) and Langchain[6] also release library for building agent and multi-agent systems. Also, new LAA methods Yao et al. (2023a); Zheng et al. (2024); Jang (2023); Yao et al. (2023b) emerge quickly in regards of the optimal approach for agent interactions and reasoning.

Due to the initial investigation, LAA is rather under-explored. Firstly, the optimal agent architecture is undetermined. ReAct (Yao et al., 2023a) prompts the agents with pre-defined examples such that the LLM learns to generate the next action via in-context learning. Moreover, ReAct argues that

[*] zhiweiliu@salesforce.com

[1] https://github.com/salesforce/BOLAA

[2] https://github.com/SalesforceAIResearch/AgentLite/tree/main/benchmark

[3] https://github.com/yoheinakajima/babyagi

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

[5] https://github.com/Significant-Gravitas/Auto-GPT

[6] https://github.com/langchain-ai/langchain/

an agent should have intermediate reasoning steps before action executions. ReWOO (Xu et al., 2023) introduces additional planning steps for LAA. Reflection (Jang, 2023; Yao et al., 2023b) are verified as effective self-correction reasoning strategy for agents. Langchain generalizes the agent with zero-shot tool usage ability. Intrinsically, the optimal architecture of agents should be aligned with both tasks and the associated LLM backbone, which is less explored in the existing works.

Secondly, understanding the efficacy of the existing LLMs in LAA is far from comprehensive. The existing preliminary works only compare the performances of a few LLM backbones. ReAct adopts the PaLM (Chowdhery et al., 2022) as the backbone LLM. ReWOO employs OpenAI text-davinci-003 model for instruction-tuning Alpaca model (Taori et al., 2023) for agent planning. MIND2Web (Deng et al., 2023) compares Flan-T5 and OpenAI GPT3.5/4 for generalist web agent. Nevertheless, few current works comprehensively compare the performance of LAA with regard to various pre-trained LLMs. A very recent work (Liu et al., 2023) releases a benchmark for evaluating LLMs as Agents. And AgentTuning (Zeng et al., 2023) enables the generalization ability for LLM on multiple enviroments. Nevertheless, they seldomly consider the agent architectures along with their LLM backbones. Selecting the optimal LLMs from both efficacy and efficiency perspectives advances the current exploration of LAA.

Thirdly, the increasing complexity of tasks may require the orchestration of multiple agents. ReWOO recently identifies that decoupling reasoning from observation improves the efficiency for LAA. In this paper, we argue that as the task complexity increases, especially in open-domain environments, it is better to coordinate multiple agents to complete one task. For example, regarding the web navigation task, we could employ one *click agent* to interact with clickable buttons and request another *search agent* to retrieve additional resources. Nonetheless, there are few works discussing how to orchestrate multiple agents and investigating the impacts of orchestration.

To address these research gaps, this paper proposes to comprehensively compare the performances of LAAs. We dive deep into the agent architecture of LAAs and the LLM backbones. Specifically, we construct agent benchmarks from the existing environments to evaluate the performances of various agent architectures built upon various LLM backbones. The tasks in our agent benchmarks are associated with different task complexity levels, which enables the agent performance analyses w.r.t. task complexity. Regarding the orchestration of multiple LAAs, we propose a novel LAA architecture BOLAA[7], which has a controller module on top of multiple labor agents, for enabling the selection and communication between multiple labor LAAs. The core in BOLAA is agent selection module and communication module. This paper studies these two modules from both heuristic-based and LLM-based methods. The contributions of this paper are as follows:

- We develop 6 different LAA agent architecture. We combine them with various backbone LLMs to justify the designing intuition of LAA from prompting, self-thinking, and planning. We also develop BOLAA for orchestrating multi-agent strategy, which enhances the action interaction ability of individual agents.
- We conduct extensive experiments on both decision-making web navigation environment and knowledge reasoning task environment. We report the performance in terms of final sparse rewards and intermediate recalls, which provides qualitative indications for the optimal choice of LAAs as well as their compatible LLMs.
- BOLAA on the WebShop environment consistently yields the best performance compared with other LAA architectures. Our results demonstrate that the importance of designing specialist agents to collaborate on resolving complex task, which should be as equally important as training a large LLM with high generalization ability.

2 Related Work

2.1 Augmented Language Agent Architecture

The completion of a complex task typically entails multiple stages. An agent must possess an understanding of these stages and plan accordingly. Chain-of-Thoughts (CoT) (Wei et al., 2022) is a groundbreaking work that prompts the agent to deconstruct challenging reasoning tasks into

[7] For easy memorizing, we intentionally name it the same as paper title.

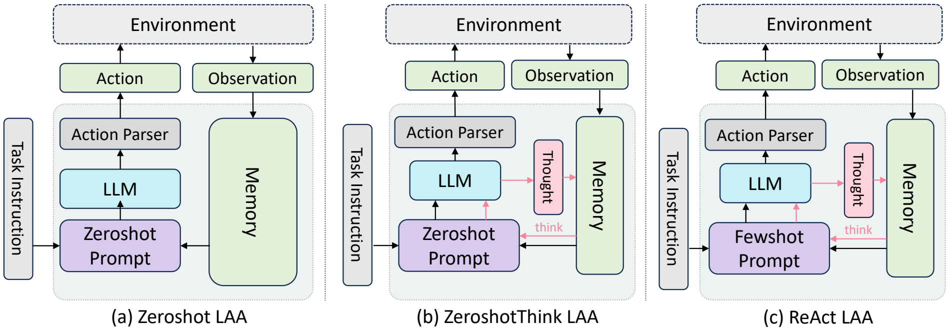


Figure 1: The LAA architectures for Zeroshot-LAA (ZS-LAA), ZeroshotThink LAA (ZST-LAA) and ReAct LAA. ZS-LAA generates actions from LLM with zeroshot prompt. ZST-LAA extends ZS-LAA with self-think. ReAct LAA advances ZST-LAA with fewshot prompt. They all resolve a given task by interacting with environment via actions to collect observations. Better view in colors.

smaller, more manageable steps. On the other hand, ReAct (Yao et al., 2023a) proposes leveraging this aptitude for reasoning and action. This agent architecture has given rise to various applications, including HuggingGPT (Shen et al., 2023), Generative Agents (Park et al., 2023), WebGPT (Nakano et al., 2021), AutoGPT (Gravitas, 2023), BabyAGI (Nakajima, 2023), and Langchain (Chase, 2023). However, these approaches neglect to incorporate valuable feedback, such as environment rewards, to enhance the agent's behaviors. Self-refine (Madaan et al., 2023a; Murthy et al., 2023; Hao et al., 2023; Shinn et al., 2023; Yao et al., 2023b) tackles this limitation by employing a single LLM as a generator, refiner, and provider of feedback, enabling iterative refinement of outputs.

2.2 Web and Tool Agent

Web navigation is the foundation for humans to collect information and communicate. Before the boom of LLM, previous endeavours (Liu et al., 2018; Shi et al., 2017) already explored how to train web agent in a web simulation environment. Very recently, a series of works have been devoted to developing LAA to tackle complex web navigation tasks. MIND2Web (Deng et al., 2023) collects a web browser data to fine-tune LLM to generate executable actions, which functions as a Web LAA. WebAgent (Gur et al., 2023) is able to decompose task instruction into sub-tasks, which directly generates executable python program for web navigation. WebArena (Zhou et al., 2023) supports realistic tasks simulation for designing Web LAA. Langchain and ChatGPT redefines LLM to behave as Web LAA. SeeAct Zheng et al. (2024) enables the vision ability of web agent.

Besides web browsing, LLMs are also able to leverage external tools to enhance their capabilities and solve complex tasks, such as *Gorilla* (Patil et al., 2023), *ToolLLM* (Qin et al., 2023), tool documentation (Hsieh et al., 2023) and etc (Liu et al., 2023; Zeng et al., 2023). These works verify the superior ability of LLMs in harnessing tools to solve more complex and open domain tasks.

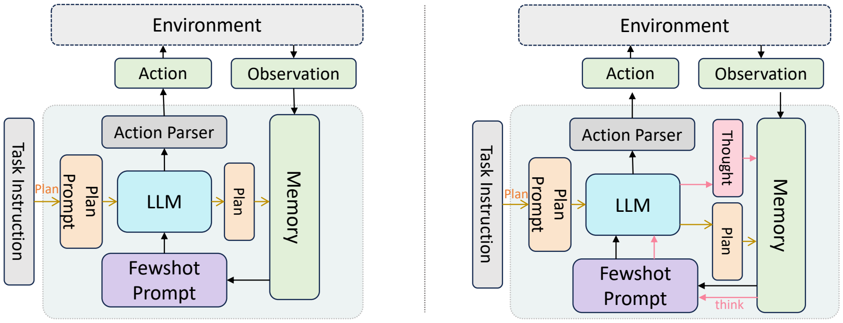
3 Agent Architectures

In this section, we compare various LAA architectures. We first present how to design different individual agents. We then present the our orchestration designing of multiple agents, *i.e.* BOLAA.

3.1 Individual Agents

Hereafter, we present 5 different LAAs. Each type of LAA is able to interact with the environment with its own interaction strategy.

Zeroshot LAA (ZS-LAA) directly extends the LLM to be action executor. Specifically, the prompt for LLMs to function as the action executor consists of detailed descriptions for those actions. For example, if we prompt LAA to understand the *click* action with "*click: using this action to click observed [button], the clickable buttons are in [].*", it may behave as a web navigation agent. We present the architecture of ZS-LAA in Figure 1(a). The working flow is as follows:



(a) PlanAct LAA

(a) PlanReAct LAA

Figure 2: The LAA architectures for PlanAct LAA and PlanReAct LAA.

- *Initial step*: firstly, the ZS-LAA receives the task instruction and constructs the zeroshot prompt. Then, the LLM layer generates a possible response, which is parsed to output a feasible action. After that, the observation from environment is appended into the agent memory.
- *Working steps*: the agent checks whether the task is finished. If not, ZS-LAA retrieves the previous actions and observations from memory, and constructs the prompts for LLM to generate the next executable actions. ZS-LAA continues the working stages until reaching the maximum steps or completing the task.

ZS-LAA is a minimum LAA architecture. It enables the action generation ability of LLM via zeroshot prompt layer, which is easy to generalize to new environments and requires no examples.

ZeroshotThink LAA (ZST-LAA) is an extended version of ZS-LAA. Different from ZS-LAA, ZST-LAA has an additional self-think flow. The architecture of ZST-LAA is presented in Figure 1(b), where we denote the self-think flow as in pink arrow lines. Self-think is running in intermediate steps of action generations flow, which enables the Chain-of-Thought (CoT) reasoning ability.

- *Self-think Step*: before generating the next action, ZST-LAA collect observations and previous actions to construct the *think* prompt. Then, the *thought* is stored into memory.

Self-think step is generally useful when given reasoning tasks. Note that the think prompt is also in a zero-shot format, such as *"think: using this action to plan your actions and reasoning"*.

ReAct LAA additionally advances ZST-LAA in the prompt layer, where fewshot examples are provided. The architecture of ReAct LAA is illustrated in Figure 1(c). ReAct LAA is able to leverage successful running examples to improve the action generation ability of LLM and enhance the environment interaction of LAA, because those fewshot examples endows the in-context learning ability of LLM. However, the drawback for ReAct LAA is that, due to the limited context length, fewer token spaces are available after the occupancy of fewshot examples in the prompt.

PlanAct LAA is designed to facilitate the planning ability of LAA. PlanAct LAA differs from ZS-LAA in two parts: 1) the planning flow and 2) the fewshot prompt. The architecture is depicted in Figure 2. The planning flow is executed before the initial action generation step, which has additional plan prompt to construct the input for the core LLM.

- *Planning Step*: PlanAct LAA generates a plan for a given task before interacting with environments. The plan is memorized and will be retrieved to construct prompts.

It is worth noting that the plan prompt in this paper is in fewshot way, which allows LAA to generate plans based on previous successful plans.

PlanReAct LAA extends PlanAct LAA with additional self-think flow, which also enables the CoT ability. The architecture of PlanReAct LAA is presented in Figure 2. Intuitively, since the Planning flow is executed before the LAA observes the environment, self-think flow alleviates the hallucination incurred from incorrect plans. Next, we introduce our BOLAA orchestrating architecture.

3.2 BOLAA: Orchestrating Multiple Agents

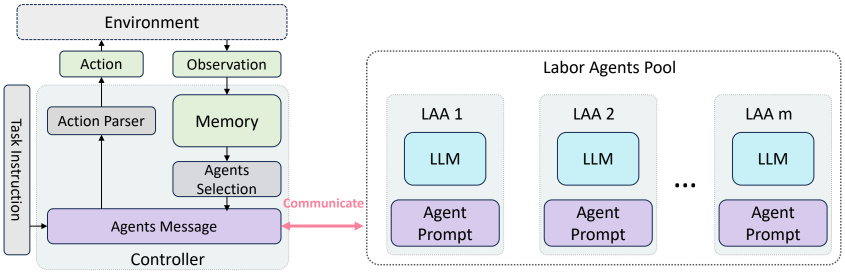


Figure 3: The BOLAA architecture, which employs a controller to orchestrate multiple LAAs.

Though the success of the existing LLMs in completing various language understanding tasks, plenty of issues are still under-explored, such as the context length constraints, in-context learning and generalization ability, and etc. Hence, it is challenging to employ an individual LAA to complete all tasks, especially when tasks are of high complexity. Therefore, we propose a new agent architecture for orchestrating multiple LAAs, which is illustrated in Figure 3. BOLAA has two main modules, the labor agents pool and the controller. The labor agents pool manages multiple LAAs. Each LAA may only focus on completing specialized tasks. For example, in the web navigation environment, we could establish *click* LAA and *search* LAA. In this way, the former only generates the next button to click, while the later only outputs search query, which divides a complex task into feasible tasks. The controller is devised to selectively call LAAs from agents pool. Controller has agents selection layer to choose the most relevant LAA(s) to call.

Agent Selection in BOLAA is one the core parts for orchestration. In this paper, we investigates two types of selection process, *i.e. heuristic-based* and *LLM-based* method. The heuristic-based method is to pre-define rules for selecting the labor LAA. Rules could be defined based on observation, generated actions, etc. The LLM-based method is designing the controller based an LLM, and enabling the labor agent selection as an action generation process of the LLM. As such, the controller is functioning as the orchestrator agent, and its action is to select the optimal labor agent.

Communication in BOLAA is the other core part for orchestration. After selecting the labor LAA, the controller constructs the message for the selected LAA and builds the communication. Again, we also designed two types of methods for communication, *i.e. heuristic-based* and *LLM-based* method. Heuristic-based method generates the communication message by a pre-defined template, which ingests the observation from labor agents as a message for controller to send out. LLM-based method drives an LLM to generate the communication message. The response from labor agents are organized by a prompt template. Then, the LLM is prompted to generate communication messages to those labor agents. The prompt for LLM includes the labor agent information, the task instructions, history executions and demonstration examples.

After obtaining the response from the labor LAA, the controller parses it to an executable action and then interacts with the environment. Note that we can also design those labor LAAs to be think/plan agent. In this way, the self-think and plan work flows are also retained.

4 Experiment

4.1 Environment Benchmark

We construct the evaluation benchmarks from two environments, *i.e.,* the WebShop (Yao et al., preprint) and HotPotQA (Yang et al., 2018) with Wikipedia API usage (Yao et al., 2023a). In WebShop enviroment, we sample 900 tasks ranging from 6 different complexity levels for benchmark evaluation. The BOLAA in WebShop is devised to be the orchestration on one search LAA and one click LAA to generate search query and click elements, respectively. And the selection layer is heuristic-based. Labor LAAs are selected based on observations. In HotPotQA environment, we sample 300 tasks from 3 complexity levels. The BOLAA in HotPotQA is a reasoning LAA and a search LAA, which tackling question reasoning and document retrieval tasks, respectively. The

Table 1: Average reward in the WebShop environment. Len denotes the maximum context length. **Bold** results denote the best results in one row, *i.e.* best LAA architecture w.r.t. one LLM. Underline results denote the best performance in one column, *i.e.* best LLM regarding one LAA architecture.

LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.3971	0.2832	0.3098	0.3837	0.1507	**0.5169**
vicuna-7b	2k	0.0012	0.0002	**0.1033**	0.0555	0.0674	0.0604
vicuna-13b	2k	0.0340	0.0451	0.1509	0.3120	0.4127	**0.5350**
vicuna-33b	2k	0.1356	0.2049	0.1887	0.3692	0.3125	**0.5612**
llama-2-7b-chat	4k	0.0042	0.0068	0.1248	0.3156	0.2761	**0.4648**
llama-2-13b-chat	4k	0.0662	0.0420	0.2568	**0.4892**	0.4091	0.3716
llama-2-70b-chat	4k	0.0122	0.0080	0.4426	0.2979	0.3770	**0.5040**
mpt-7b-instruct	8k	0.0001	0.0001	0.0573	0.0656	**0.1574**	0.0632
mpt-30b-instruct	8k	0.1664	0.1255	0.3119	0.3060	0.3198	**0.4381**
xgen-8k-7b-instruct	8k	0.0001	0.0015	0.0685	0.1574	0.1004	**0.3697**
longchat-7b-16k	16k	0.0165	0.0171	0.069	0.0917	0.1322	**0.1964**
longchat-13b-16k	16k	0.0007	0.0007	0.2373	0.3978	**0.4019**	0.3205
text-davinci-003	4k	0.5292	0.5395	0.5474	0.4751	0.4912	**0.6341**
gpt-3.5-turbo	4k	0.5061	0.5057	0.5383	0.4667	0.5483	**0.6567**
gpt-3.5-turbo-16k	16k	0.5657	0.5642	0.4898	0.4565	0.5607	**0.6541**

selection layer is LLM-based, where we designed prompts to ask LLM to select which LAA to call. More details about environments are in appendix. The original implementation codes[8] are released. We further provide a unified benchmark evaluation protocol[9] with AgentLite.

4.2 Evaluation Metrics

We mainly use the *reward* score in each environment to evaluate the performances of LAAs. In the WebShop environment, the reward is defined as the attribute overlapping ratio between the bought item and ground truth item. In HotPotQA environment, the reward is defined as the F1 score grading between agent answer and ground-truth answer. Additionally, we develop the *Recall* performance for WebShop environment, which is defined as 1 if the ground truth item is retrieved and 0 if not during one task session. The Recall is reported as the average recall scores across all tasks in WebShop environment.

4.3 LLM Utilization

The core component of LAA is the LLM backbone. We compare different LLMs with various choices of model size and context length. We reported the results w.r.t. open LLM models such as fastchat-3b, vicuna-1.3-7b/13b/33b (Zheng et al., 2023), Llama-2-7b/13b/70b[10] (Touvron et al., 2023), MPT-7b/30b (Team, 2023), xgen-8k-7b, longchat-16k-7b/13b and OpenAI API LLMs, including text-davinci-003, gpt-3.5-turbo and gpt-3.5-turbo-16k.

4.4 Decision-making Simulation

In this section, we present and compare the decision-making performances of LAAs in the WebShop environment. The performance regarding the average reward is reported in Table 1. The agent prompts are constructed based on the maximum context length of different LLM models. We have the following observation:

- BOLAA performs the best compared with the other LAA architectures, especially when built on the high performing LLMs. BOLAA is able to actively select the appropriate LAA and yield

[8] `https://github.com/salesforce/BOLAA`

[9] `https://github.com/SalesforceAIResearch/AgentLite/tree/main/benchmark`

[10] All Llama-2 models are -chat-hf version.

Table 2: Average recall in the WebShop environment. Len denotes the maximum context length. **Bold** results denote the best results in one row, *i.e.* best LAA architecture w.r.t. one LLM. Underline results denote the best performance in one column, *i.e.* best LLM regarding one LAA architecture.

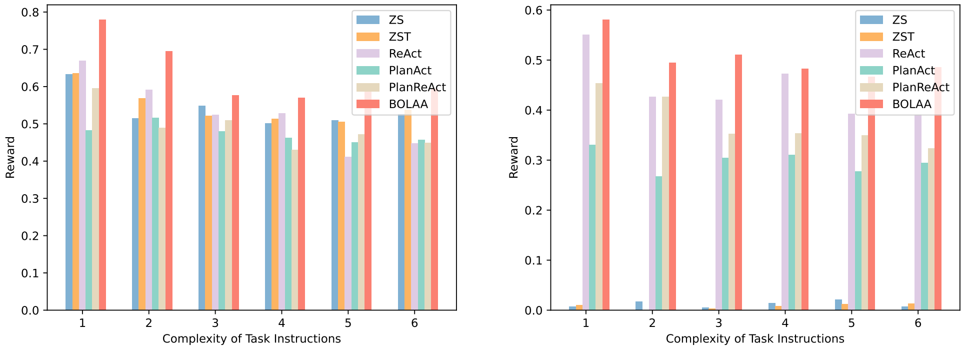
LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.3533	0.3122	0.3800	0.3700	0.3722	**0.3867**
vicuna-7b	2k	0.0833	0.0500	0.3600	0.3233	0.3278	**0.3522**
vicuna-13b	2k	0.0867	0.0644	0.3622	0.3444	0.2367	**0.3700**
vicuna-33b	2k	0.3600	0.3411	0.3822	0.3733	0.3567	**0.3956**
llama-2-7b-chat	4k	0.0678	0.0311	0.3744	0.3400	0.3578	**0.3856**
llama-2-13b-chat	4k	0.2856	0.2211	0.3844	0.3278	0.3500	<u>**0.4078**</u>
llama-2-70b-chat	4k	0.3344	0.3244	0.3789	0.3400	0.3600	**0.4011**
mpt-7b-instruct	8k	0.0144	0.0322	**0.3644**	0.3200	0.3400	0.3600
mpt-30b-instruct	8k	0.2973	0.3372	0.3333	0.3575	0.3412	**0.3900**
xgen-8k-7b-instruct	8k	0.0667	0.1400	0.3711	0.3400	0.3278	**0.3800**
longchat-7b-16k	16k	0.1344	0.1856	0.3644	0.3622	0.3622	**0.3811**
longchat-13b-16k	16k	0.0756	0.0867	0.3678	0.3467	0.3471	**0.3789**
text-davinci-003	4k	0.3800	<u>0.3856</u>	0.3767	0.3711	<u>0.3889</u>	**0.3956**
gpt-3.5-turbo	4k	<u>0.3889</u>	0.3756	**0.3933**	<u>0.3789</u>	0.3867	0.3929
gpt-3.5-turbo-16k	16k	0.3856	0.3833	<u>**0.4011**</u>	0.3756	0.3811	0.3933

qualitative communication, which stabilizes the action generation. We observe that BOLAA, when paired with a 3b fastchat-t5 LLM, performs comparably to other LAA architectures with more powerful LLMs. The superiority of BOLAA indicates that orchestrating multiple smaller-sized LAAs is a better choice if the computing resources are limited. This further exemplifies the potential for fine-tuning multiple smaller-sized specialised LAAs rather than fine-tuning one large generalized LAA.

- Pairing the LLM with the optimal LAA architecture is crucial. For example, Llama-2-13b performs best under PlanAct LAA arch while Llama-2-70b performs best under the BOLAA arch. Also, Longchat-13b-16K performs best when using PlanAct and PlanReAct, which may indicate the extraordinary planning ability of longchat-13b-16k models.
- Increasing the context length alone may not necessarily improve the LAA performances. For example, when comparing longchat-13b-16k with llama-2-13b models, the latter yields better performances though with less context length. By checking the running log of those LAAs, we observe more occurrence of hallucinated generation when the LAA runs for more steps, which in the end degrades the benefits of longer context.
- A powerful LLM is able to generalize under the zeroshot LAA arch. The best performance of OpenAI API-based models are actually under ZS and ZST arch. This indicates the great potential of developing a generic LAA with powerful LLM. Actually, this is currently what open-source projects are working towards, directly calling OpenAI API and tuning the zeroshot agent prompt instead. Our benchmark results quantitatively justify that using only a ZS LAA can already achieve comparable or even better performances than LAA arch with additional Plan or Self-think flow. However, for other less powerful LLMs, fewshot prompts are necessary for LAAs.

We also report the intermediate Recall performances for all LAAs, which are illustrated in Table 2. High recall performances indicate that the LAA is capable of generating a precise search query. High recalls usually lead to better rewards. But they are not tightly related. For example, Llama-2-70b has a recall performance of nearly 0.3344 on ZS LAA, which is comparable to the best LAA. However, the reward performance in Table 1 of ZS LAA Llama-2-70b is only 0.0122. The reason is that generating the search query requires a different LLM ability from generating the correct click action, where the latter is more challenging. Another observation is that our proposed BOLAA generally performs the best on all LLMs, which indicates that separating the search agent from the click agent improves the accuracy of the search action, leading to a higher recall value.

LAA performance w.r.t. Complexity. After the overall performances of those LAAs and LLMs are compared, we conduct more details investigation of the performance w.r.t. the task complexity.



(a) text-davinci-003

(b) Llama-2-70b

Figure 4: The reward w.r.t. task complexity in WebShop. Each bar represents one LAA.

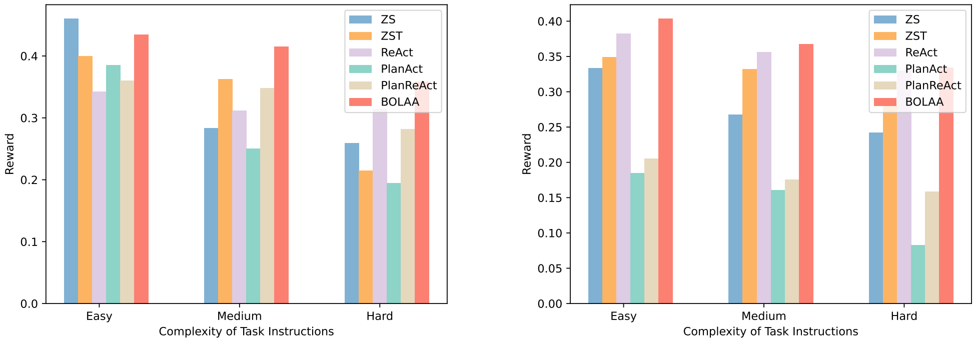
Table 3: Average reward in the HotPotQA environment. Len denotes the maximum context length. **Bold** results denote the best results in one row, *i.e.* best LAA architecture w.r.t. one LLM. Underline results denote the best performance in one column, *i.e.* best LLM regarding one LAA architecture.

LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.0252	0.0067	0.0692	0.1155	0.0834	**0.1221**
vicuna-7b	2k	0.1339	0.0797	0.0318	0.0868	0.0956	**0.1521**
vicuna-13b	2k	0.1541	0.0910	0.2637	0.1754	0.2075	**0.2721**
vicuna-33b	2k	0.2180	0.2223	0.2602	0.1333	0.2016	**0.2754**
llama-2-7b-chat	4k	0.0395	0.0207	**0.2624**	0.1780	0.1417	0.2613
llama-2-13b-chat	4k	0.1731	0.2313	0.2521	0.2192	0.2177	**0.2773**
llama-2-70b-chat	4k	0.2809	0.3207	0.3558	0.1424	0.1797	**0.3681**
mpt-7b-instruct	8k	0.0982	0.0483	0.1707	0.1147	0.1195	**0.1775**
mpt-30b-instruct	8k	0.1562	0.2141	0.3261	0.2224	0.2315	**0.3521**
xgen-8k-7b-instruct	8k	0.1502	0.1244	0.1937	0.1116	0.1096	**0.2231**
vicuna-7b-16k	16k	0.0773	0.1053	**0.2554**	0.1759	0.1642	0.2347
longchat-7b-16k	16k	0.0791	0.0672	**0.2161**	0.1296	0.0971	0.1917
longchat-13b-16k	16k	0.1083	0.0562	0.2387	0.1623	0.1349	**0.2433**
text-davinci-003	4k	0.3430	0.3304	0.4503	0.3577	0.4101	**0.4743**
gpt-3.5-turbo	4k	0.3340	0.3254	0.3226	0.2762	0.3192	**0.3541**
gpt-3.5-turbo-16k	16k	0.3027	0.2264	0.1859	0.2113	0.2251	**0.3225**

Due to the space limitation, we only report the performance of text-davinci-003 and llama-2-70b. The reward performance is illustrated in Figure 4. The BOLAA model consistently performs better on all complexity levels. We also observe the degraded performances when the task complexity is increased, which follows the intuition. Surprisingly, we find out that further increasing the complexity of tasks greater than 4 will not further degrade the performances. The reason is that the recall performance increases when the task is of higher complexity. This is due to the fact that high-complexity task instruction provides more additional context information for the LAA. As such, the *search* action can be more specific and accurate under high complexity levels.

4.5 Knowledge Reasoning Simulation

We benchmark on the HotPotQA environment to evaluate the multi-step reasoning ability of LAAs. Since the available search, lookup and finish operations are all related to knowledge reasoning in this environment and hard to separate, it is challenging to design heuristic-based methods for agent selection and communication. Therefore, the BOLAA architecture for HotPotQA environment constitutes two labor agents, a reasoning agent and a search agent. Reasoning agent has two actions, *ask* and *answer*. Ask action triggers the search agent to search corresponding context. Answer action is to answer the given question. In our implementation, the reasoning agent is also a controller. In this sense, the agent selection and communication are all LLM-based generation process. The results are in Table 3. In general, BOLAA achieves the best performance on most LLMs. This indicates the necessity of separation the reasoning and searching into different agents. Additionally, we also observe the improvements is marginal or even no improvement when with small size LLMs. We observe that those small size LLMs is unable to well generate the communication message between reasoning agent and searching agent. Moreover, comparing ReAct, PlanAct, and PlanReAct, we would conclude that planning flow of LAA hinders performance the in knowledge reasoning environment and tasks. The reason is that knowledge reasoning tasks require contextualized information to conduct reasoning, whereas planning flow is executed ahead of interactions. Thus, those generated plans tend to lead to more hallucination of LAA. Further, regarding this knowledge reasoning task, model size is much more important than the context length. Large-sized model has better abilities in reasoning, thus performing better. We also observe the best performance of Llama-2-70b on all open-source LLMs, which suggests its potentially great fine-tuning ability.



(a) text-davinci-003

(b) Llama-2-70b

Figure 5: The reward w.r.t. complexity level in HotPotQA. Each bar represents one LAA.

LAA performance w.r.t. Complexity. Since we have easy, medium, and high level tasks, we compare the performance of Llama-2-70b and OpenAI gpt-3 model regarding different levels of complexity, as illustrated in Figure 5. We observe degrading performance if increasing the complexity of tasks. In HotPotQA tasks, the hardness is defined as the question answer hops. Therefore, hard question requires more context understanding and reasoning ability of LAA. BOLAA outperforms other agent arches on most cases, especially on hard questions. This suggests that disentangling the searching and reasoning as two agents strengthens question answering ability. Another observation is that though OpenAI text-davinci-003 model consistently outperforms Llama-2-70b on all levels of complexity, their difference is of smaller margin in hard questions. Since hard questions require more reasoning efforts, we can conclude that Llama-2-70b posses comparable reasoning ability with text-davinci-003.

5 Conclusion and Future Work

In this paper, we systematically investigate the performances of various LAA architecture paired with different LLM backbones. We also provide one novel orchestrating method for multiple agents, *i.e.* BOLAA. The benchmarking results provide experimental justification for the LAA investigation and verify the potential benefits of BOLAA architecture. During the investigation, we also identify the challenge of designing BOLAA architecture for environments with compounding actions. In the future, we will continue developing more LLM agent architectures and include more LLMs and environments for evaluations. Also, we will keep exploring how to designing the separation and orchestration of multiple agents, including more advanced techniques in agent selection and communication.

References

Harrison Chase. Langchain. `https://github.com/hwchase17/langchain`, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023.

Significant Gravitas. Autogpt. `https://github.com/Significant-Gravitas/Auto-GPT`, 2023.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. Tool documentation enables zero-shot tool-usage with large language models. *arXiv preprint arXiv:2308.00675*, 2023.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pp. 9118–9147. PMLR, 2022.

Eric Jang. Can llms critique and iterate on their own outputs? *evjang.com*, Mar 2023. URL `https://evjang.com/2023/03/26/self-reflection.html`.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*, 2023.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*, 2018.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023.

Aman Madaan, Alexander Shypula, Uri Alon, Milad Hashemi, Parthasarathy Ranganathan, Yiming Yang, Graham Neubig, and Amir Yazdanbakhsh. Learning performance-improving code edits. *arXiv preprint arXiv:2302.07867*, 2023a.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023b.

Rithesh Murthy, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Le Xue, Weiran Yao, Yihao Feng, Zeyuan Chen, Akash Gokul, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Rex: Rapid exploration and exploitation for ai agents, 2023.

Yohei Nakajima. Babyagi. `https://github.com/yoheinakajima/babyagi`, 2023.

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

OpenAI. Gpt-4 technical report. *ArXiv*, 2023.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144. PMLR, 2017.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL `www.mosaicml.com/blog/mpt-7b`. Accessed: 2023-05-05.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. 2023.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023a.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *ArXiv*, preprint.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization, 2023b.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms, 2023.

Jianguo Zhang, Kun Qian, Zhiwei Liu, Shelby Heinecke, Rui Meng, Ye Liu, Zhou Yu, Huan Wang, Silvio Savarese, and Caiming Xiong. Dialogstudio: Towards richest and most diverse unified dataset collection for conversational ai, 2023.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL `https://webarena.dev`.

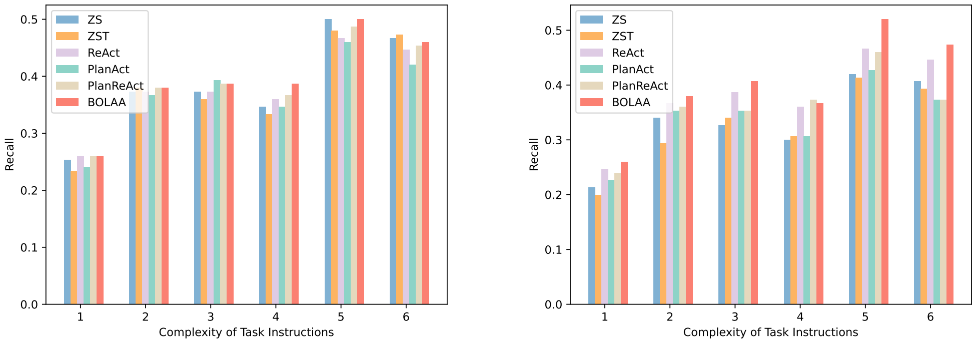
A Environment Setup

WebShop is a recently proposed online shopping website environment with 1.18M real-world products and human instructions. Each instruction is associated with one ground-truth product, and contains attribute requirements, *e.g. I'm looking for a travel monopod camera tripod with quick release and easy to carry, and price lower than 130.00 dollars.* This instruction includes 3 attribute requirements *i.e.* "quick release", "camera tripod" and "easy carry" attributes. We define the complexity of an instruction using the number of attribute requirements. Thus, this instruction example above is of complexity 3. We equally sample 150 instructions regarding each complexity level. Since we have fewer than 150 instructions for complexity larger than 6, we only include instructions from complexity in $\{1, 2, \ldots, 6\}$, which sums up to 900 tasks for benchmark evaluation in the WebShop environment. In the WebShop environment, an agent operates either SEARCH[QUERY] or CLICK[ELEMENT] actions to interact the environment, for evaluating the interactive decision making ability of LAA. The observation from WebShop is simplified web browser, which includes the clickable buttons and associated page content. LAA interacts with the WebShop environment as a web navigation agent.

HotPotQA with Wikipedia API is another environment considered in this paper, which contains multi-hop questions answering tasks that requires reasoning over two or more Wikipedia passages. This simulation environment serves as a powerful tool for evaluating the multi-step planning and comprehension capabilities and information retrieval skills of AI models, ensuring they are proficient in sourcing reliable information from vast online resources. With its unique blend of real-world internet browsing scenarios and text analysis, HotpotQA is an invaluable asset for the advancement of augmented large language agent systems. In HotPotQA environment, an agent has three types of actions, *i.e.*, SEARCH[ENTITY], LOOKUP[STRING] and FINISH[ANSWER] to interact with HotPotQA environment. HotPotQA environment aims at evaluate the knowledge reasoning ability of LAA. We randomly sample 100 questions from easy, medium and hard levels, which constitutes the final 300 benchmark questions for evaluating LAAs.

B Additional Performance Report

We include some additional performance reports in appendix. The recall performance of text-davinci-003 and Llama-2-70b-chat w.r.t. different complexity levels in Webshop enviroment are illustrated in Figure 6. We observe that text-davinci-003 has the better performance compared with Llama-2. And BOLAA generally outperforms other agent architectures on all different levels of complexity.



(a) text-davinci-003

(b) Llama-2-70b

Figure 6: The recall w.r.t. task complexity in WebShop. Each bar represents one LAA.