

Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park
Stanford University
Stanford, USA
joonspk@stanford.edu

Joseph C. O'Brien
Stanford University
Stanford, USA
jobrien3@stanford.edu

Carrie J. Cai
Google Research
Mountain View, CA, USA
cjcai@google.com

Meredith Ringel Morris
Google DeepMind
Seattle, WA, USA
merrie@google.com

Percy Liang
Stanford University
Stanford, USA
pliang@cs.stanford.edu

Michael S. Bernstein
Stanford University
Stanford, USA
msb@cs.stanford.edu

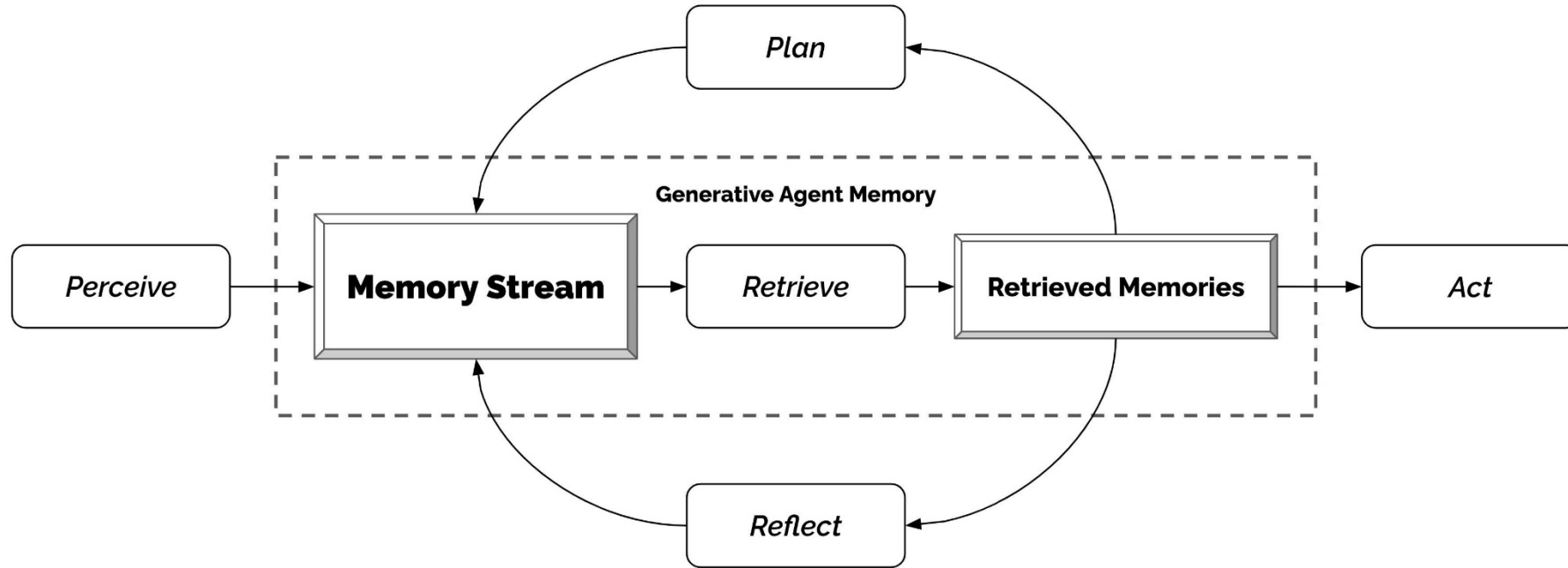


Motivation

Despite striking progress in large language models [18] that can simulate human behavior at a single time point [39, 80], fully general agents that ensure long-term coherence would be better suited by architectures that manage constantly-growing memories as new interactions, conflicts, and events arise and fade over time while handling cascading social dynamics that unfold between multiple agents.

In this paper, we introduce generative agents—agents that draw on generative models to simulate believable human behavior—and demonstrate that they produce believable simulacra of both individual and emergent group behavior.

ARCHITECTURES



To enable generative agents, we describe an agent architecture that stores, synthesizes, and applies relevant memories to generate believable behavior using a large language model. Our architecture comprises three main components. The first is the memory stream, a long-term memory module that records, in natural language, a comprehensive list of the agent's experiences. A memory retrieval model combines relevance, recency, and importance to surface the records needed to inform the agent's moment-to-moment behavior. The second is reflection, which synthesizes memories into higherlevel inferences over time, enabling the agent to draw conclusions about itself and others to better guide its behavior. The third is planning, which translates those conclusions and the current environment into high-level action plans and then recursively into detailed behaviors for action and reaction. These reflections and plans are fed back into the memory stream to influence the agent's future behavior.

RETRIEVAL

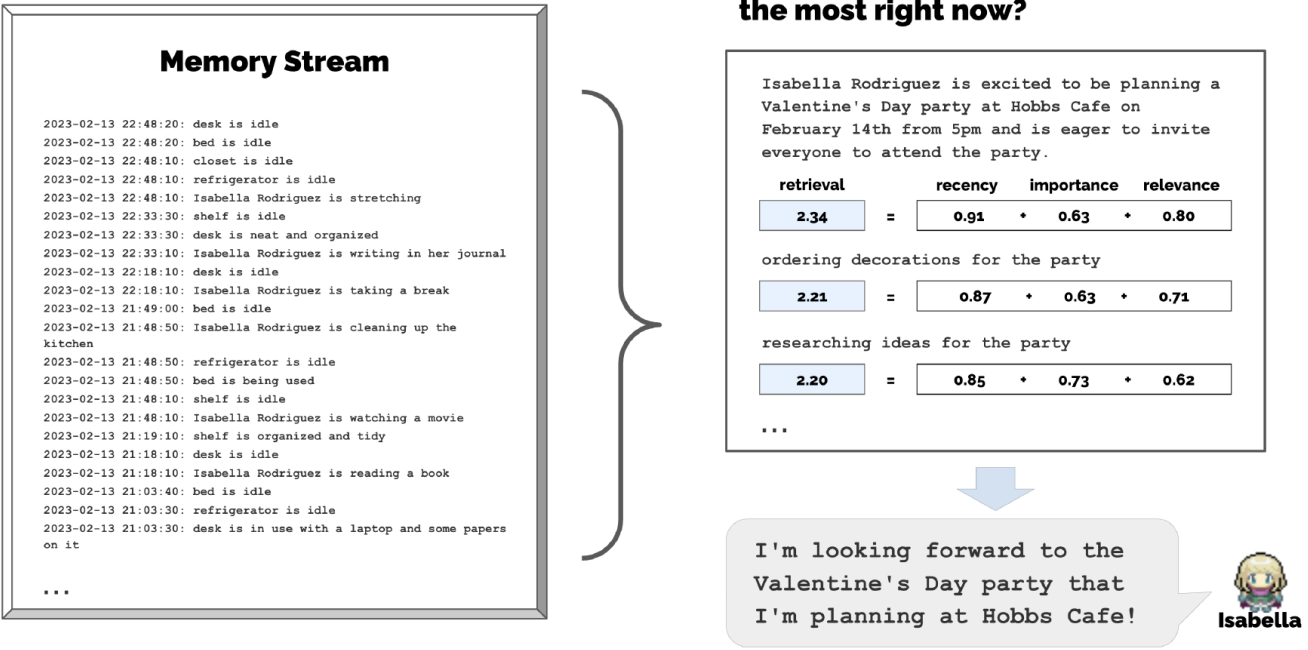


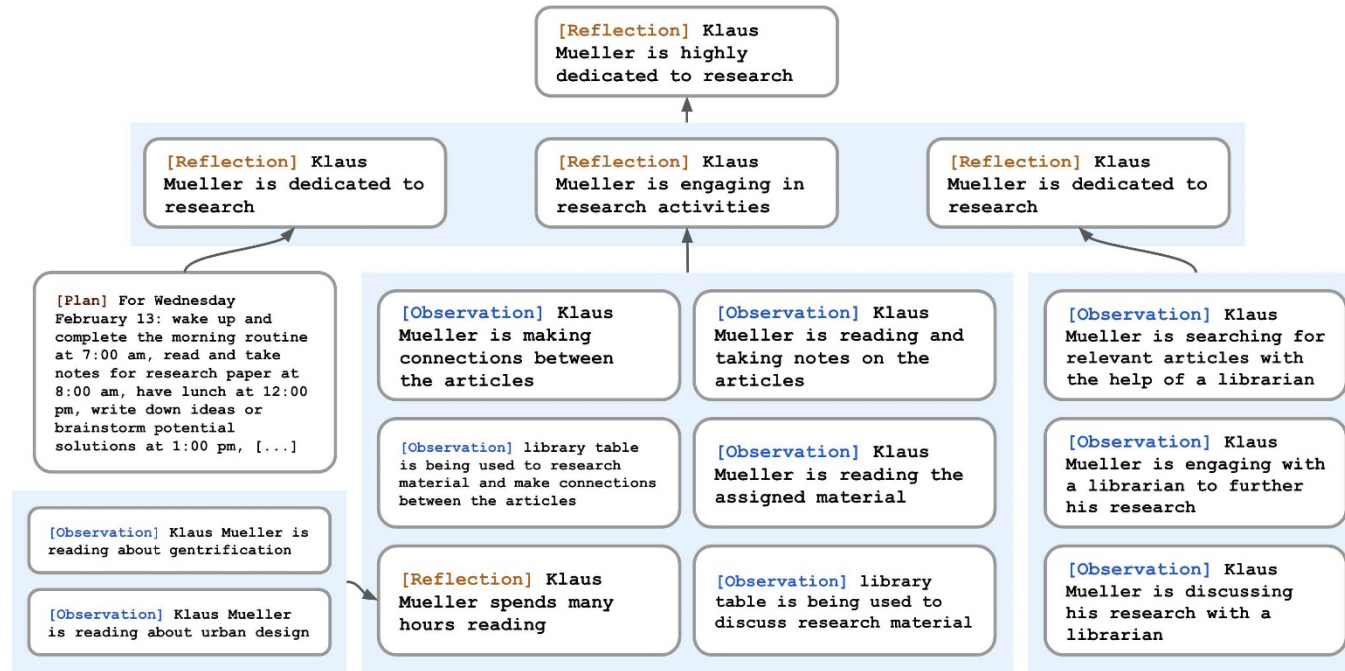
Figure 6: The memory stream comprises a large number of observations that are relevant and irrelevant to the agent’s current situation. Retrieval identifies a subset of these observations that should be passed to the language model to condition its response to the situation.

The retrieval function scores all memories as a weighted combination of the three elements:

$$score = \alpha_{recency} \cdot recency + \alpha_{importance} \cdot importance + \alpha_{relevance} \cdot relevance.$$

In our implementation, all α s are set to 1.

REFLECTION



Reflections are generated periodically; in our implementation, we generate reflections when the sum of the importance scores for the latest events perceived by the agents exceeds a threshold (150 in our implementation).

We query the large language model with the 100 most recent records in the agent's memory stream and prompt the language model. The model's response generates candidate questions.

We use these generated questions as queries for retrieval, and gather relevant memories (including other reflections) for each question. Then we prompt the language model to extract insights and cite the particular records that served as evidence for the insights.

PLAN

Like reflections, plans are stored in the memory stream and are included in the retrieval process. This allows the agent to consider observations, reflections, and plans all together when deciding how to behave. Agents may change their plans midstream if needed.

To create such plans, our approach starts top-down and then recursively generates more detail. The first step is to create a plan that outlines the day's agenda in broad strokes. To create the initial plan, we prompt the language model with the agent's summary description (e.g., name, traits, and a summary of their recent experiences) and a summary of their previous day.

BOLAA: BENCHMARKING AND ORCHESTRATING LLM-AUGMENTED 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 Research, USA

[◇]CTO Office, Salesforce, USA

[◇]Corresponding Authors: {huan.wang, cxiong, ssavarese}@salesforce.com

Motivation

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.

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. Nevertheless, they fail to jointly consider the agent architectures along with their LLM backbones.

The increasing complexity of tasks may require the orchestration of multiple agents.

Re-WOO recently identifies that decoupling reasoning from observation improves the efficiency for LAA.

作者强调了专业的agents的重要性，据此开发了BOLAA中的控制器，来进行多agents协同工作。LAA的测试需要同时考虑其agents架构和使用的LLMs。

AGENT ARCHITECTURES

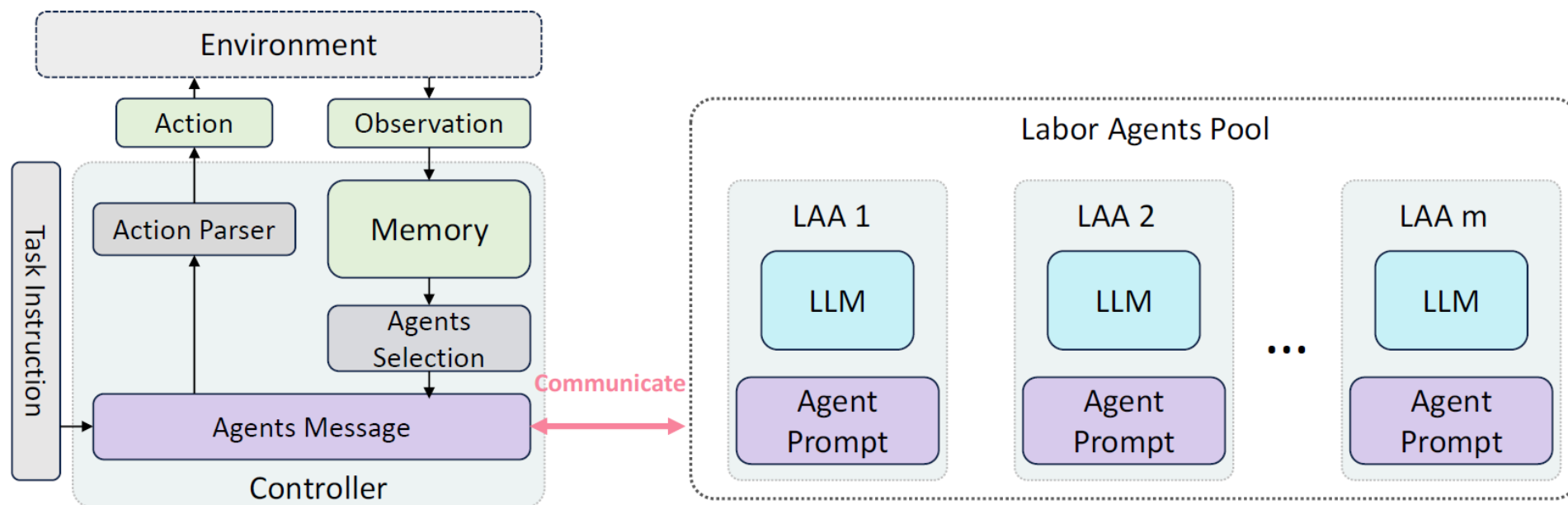


Figure 3: The BOLAA architecture, which employs a controller to orchestrate multiple LAAs.

*BOLAA*中的agents调用通过一个labor池来实现，其中labor池控制器可以以agents的形式来实现。

AGENT ARCHITECTURES

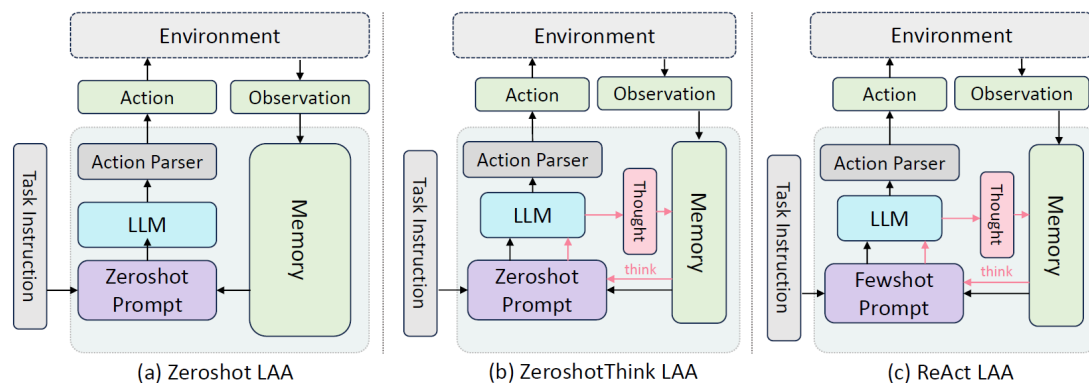


Figure 1: The LAA architectures for Zero-shot-LAA (ZS-LAA), Zero-shotThink LAA (ZST-LAA) and ReAct LAA. ZS-LAA generates actions from LLM with zeroshot prompt. ZST-LAA extends ZS-LAA with self-think. ReAct LAA advances ZST-LAA with fewshot prompt. They all resolve a given task by interacting with environment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

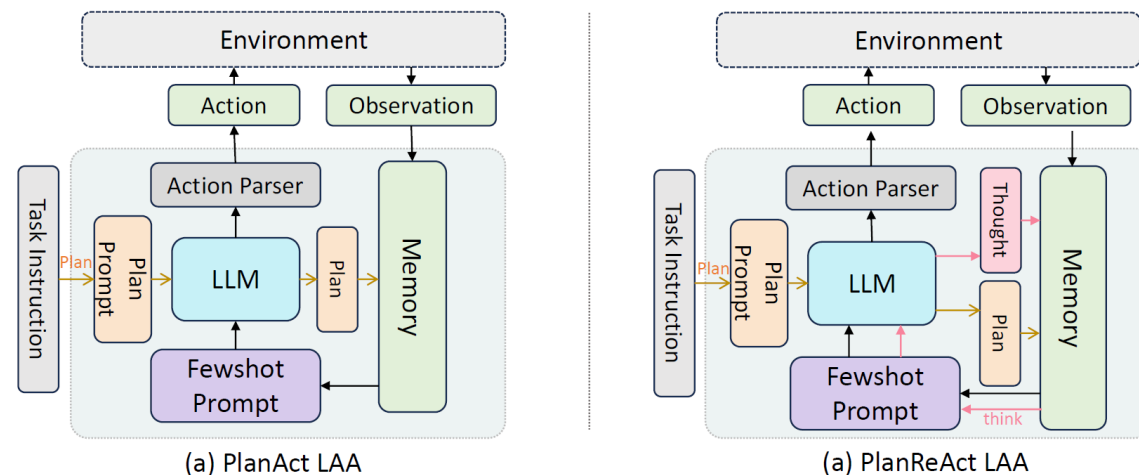


Figure 2: The LAA architectures for PlanAct LAA and PlanReAct LAA.

作者设计了5种agents。其中的区别在于Plan的引入、think的干预、少样本的添加。

在PlanAct LAA中Plan Prompt引入了少样本。而PlanReAct LAA在此基础上引入了Thought，这在一定程度上减少了引入Plan的幻觉问题。

Thought的引入使得Cot在agents中发挥作用。

ENVIRONMENT BENCHMARK

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” ,		WebShop	HotPotQA
HotPotQA with WebShop questions are simulated in a simulation environment	agents' operation	SEARCH[QUERY] or CLICK[ELEMENT] actions	SEARCH[ENTITY], LOOKUP[STRING] and FINISH[ANSWER]
comprehension can be used to source reliable information	采样	按照指令需要的属性个数作为复杂度划分，基于数据集中属性的分布，采用的复杂度范围为{1, 2, ..., 6}，分别采样150个。	按照简单、中等、复杂三个层次分别采样100个样本。
	指标	属性的重合比率	F1 分数

此外，作者还为WebShop做了召回率（Recall）的测试。作者还从任务的复杂度出发对于WEBSHOP和HOTPOTQA数据集进行了测试。

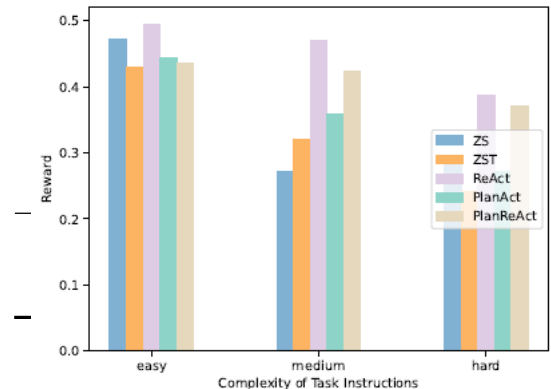
EXPERIMENT

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.

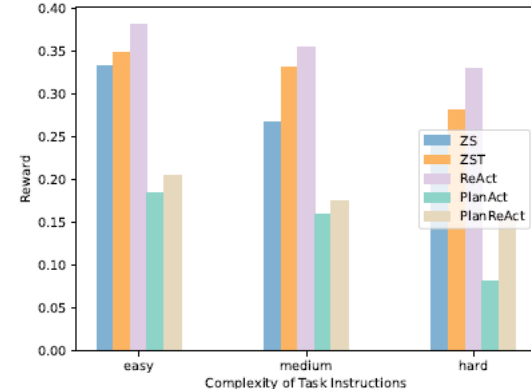
LLM	Len.	LAA Architecture			
-----	------	------------------	--	--	--

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			
-----	------	------------------	--	--	--



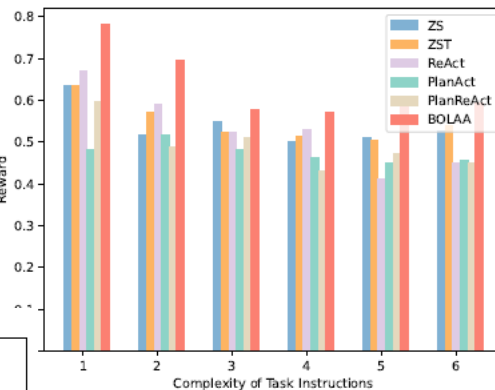
(a) text-davinci-003



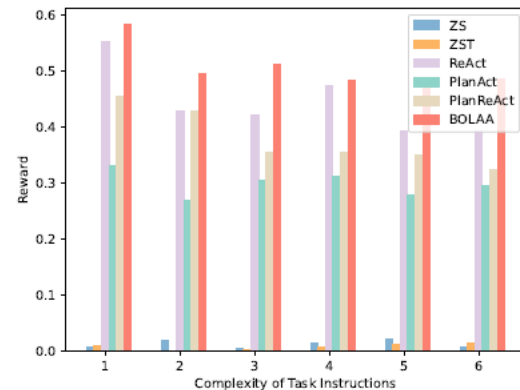
(b) Llama-2-70b

LLM	Len.	ZS	ZST	ReAct	PlanAct	PlanReAct
gpt-3.5-turbo	4k	0.3340	0.3254	0.3226	0.2762	0.2762
gpt-3.5-turbo-16k-0613	16k	0.3027	0.2264	0.1859	0.2113	0.2113

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 LAA architecture.

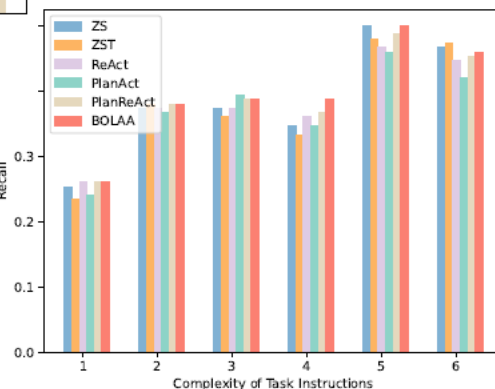


(a) text-davinci-003

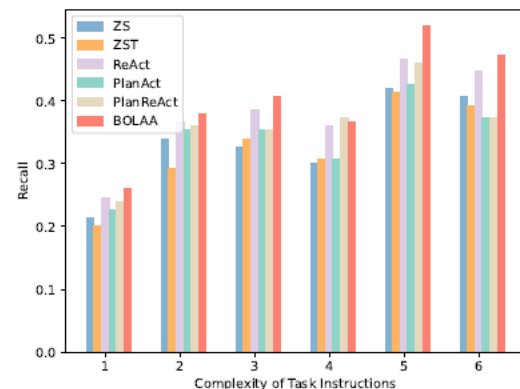


(b) Llama-2-70b

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



(a) text-davinci-003



(b) Llama-2-70b

Figure 5: The recall w.r.t. task complexity in WebShop. Each bar represents one LAA.

BOLAA
0.3867
0.3522
0.3700
0.3956
0.3856
0.4078
0.4011
0.3600
0.3900
0.3800
0.3811
0.3789
0.3956
0.3929
0.3933

CHATEVAL: TOWARDS BETTER LLM-BASED EVALUATORS THROUGH MULTI-AGENT DEBATE

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Zhiyuan Liu*

Department of Computer Science and Technology

Tsinghua University

zorowin123@gmail.com

Jie Fu, Wei Xue

Hong Kong University of Science and Technology

Shanghang Zhang

Peking University

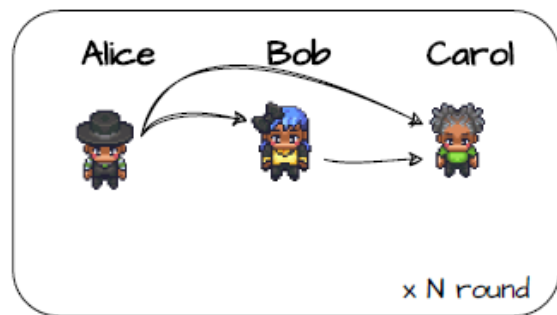
Motivation

The multi-agentbased approach enables a group of LLMs to synergize with an array of intelligent counterparts, harnessing their distinct capabilities and expertise to enhance efficiency and effectiveness in handling intricate tasks.

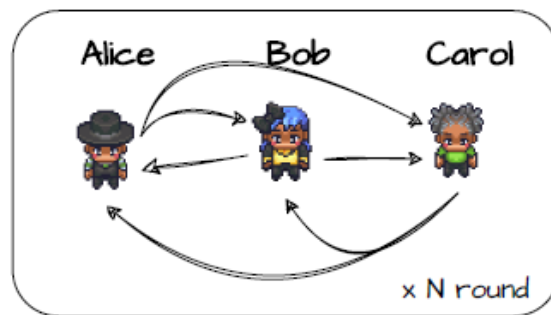
While a single powerful LLM can already tackle various missions, emerging studies suggest that multiple LLMs can further improve one another through debate and cooperation (Li et al., 2023a; Liang et al., 2023). By incorporating multiple LLMs into an integrated group and designing specific interaction mechanisms, different LLMs can engage in proposing and deliberating unique responses and thought processes across several rounds.

Furthermore, to enrich the evaluation dynamics, every agent within ChatEval is endowed with a unique persona.

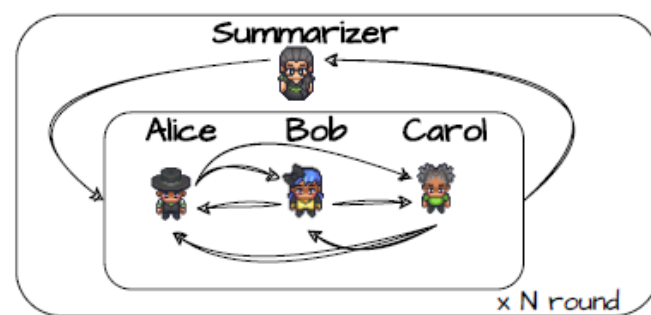
Communication Strategy



(a) One-by-One



(b) Simultaneous-Talk



(c) Simultaneous-Talk-with-Summarizer

Figure 2: The overall schematic diagram of our proposed three different kinds of communication strategy. The direction of the arrows represents the flow of information, meaning that what this person says will be appended to the chat history of the person pointed to by the arrow. Full algorithm description of the above communication strategies can be found in Appendix [B](#).

AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework

Qingyun Wu¹, Gagan Bansal², Jieyu Zhang³, Yiran Wu¹, Shaokun Zhang¹,
Erkang Zhu², Beibin Li², Li Jiang², Xiaoyun Zhang², and Chi Wang²

¹*Pennsylvania State University*

²*Microsoft*

³*University of Washington*

Motivation

Considering the evolving range of real-world tasks that could benefit from LLMs and the intrinsic weaknesses of using a single agent [29, 16, 8], a promising direction for future LLM applications is to have multiple agents work together to solve complex tasks.

To support the rapid development of next-gen LLM applications, an ideal framework would support the latest LLMs and augmentations, allow (potentially automated) cooperation between LLMs, tools, and seamless and flexible human involvement to leverage their expertise and intelligence.

ARCHITECTURES

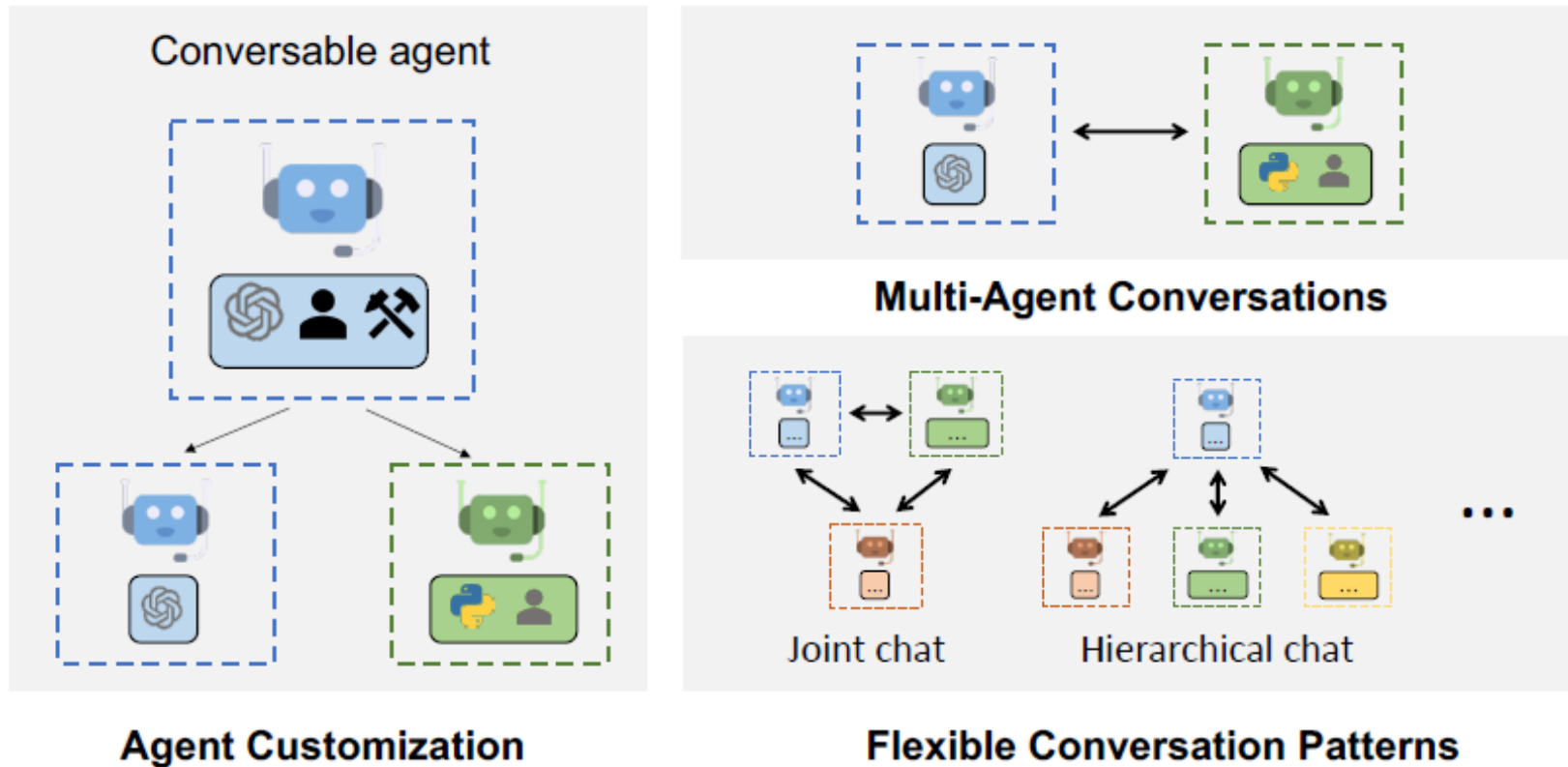
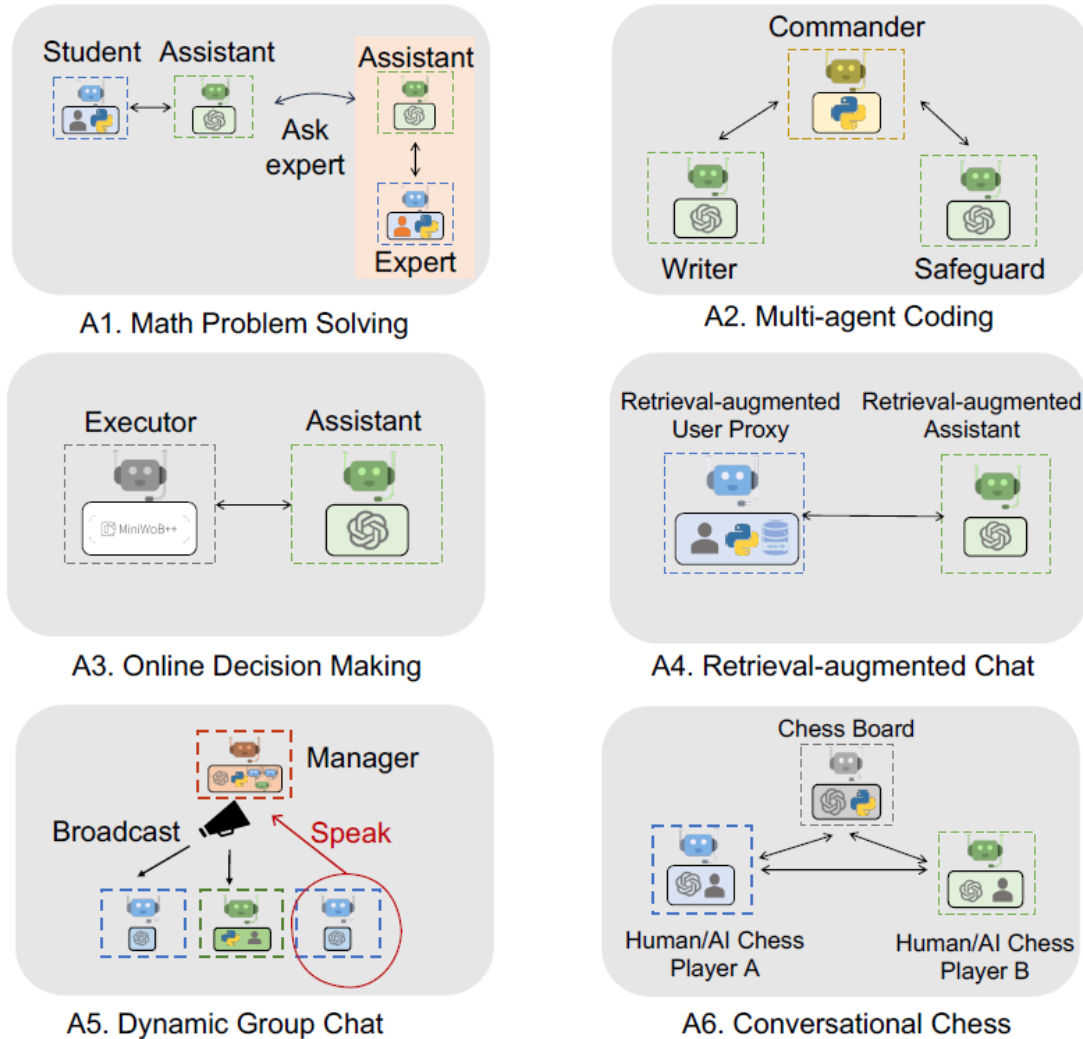


Figure 1: AutoGen enables complex LLM-based workflows using multi-agent conversations. (Left) AutoGen agents are customizable and can be based on LLMs, tools, humans and even a combination of them. (Top-right) Agents can converse to solve tasks. (Bottom-right) The framework supports many additional complex conversation patterns.

ARCHITECTURES



- A1: Math problem solving solves math problems in three scenarios
- A2: Multi-agent coding uses three agents to solve challenging supply chain optimization problems
- A3: Online decision making uses agents to solve web interaction tasks in the MiniWoB++ benchmark
- A4: Retrieval-augmented chat uses retrieval augmented agents to solve code generation and question-answering problems
- A5: Dynamic group chat showcases how to build a versatile group chat with AutoGen
- A6: Conversational chess implements a fun conversational chess game where players can creatively express moves

Figure 5: We present six examples of diverse applications built using AutoGen. Their conversation patterns show AutoGen's flexibility and power.