AI Agents: Mastering Complex Task Solving

A Survey

August 18, 2023 - **Machine Learning Reading Group**

INTERNAL – SAP Only

THE BEST RUN SAP

What are AI Agents?

“Artificial Intelligence Agent (AI Agent) is defined as a program that employs artificial intelligence techniques to perform tasks that typically require human-like intelligence”
[TPTU: Task Planning and Tool Usage of Large Language Model-based AI Agents]

What are AI Agents?

“Artificial Intelligence Agent (AI Agent) is defined as a program that employs artificial intelligence techniques to perform tasks that typically require human-like intelligence”
[TPTU: Task Planning and Tool Usage of Large Language Model-based AI Agents]

Are AI agents necessarily LLM-based agents?

No, but LLMs **are a paradigm shift for AI agents** because they are the **best and most generic world models** we have at our disposal.
I guess they will be replaced by some form of **multimodal models at some point**.

What

"Artificial
intelliger
[TPTU: Task P

Are A

No, but L
most ge
I guess t

rtificial
ligence"

and

point.

BLOG ›

Google Research, 2022 & beyond: Robotics

TUESDAY, FEBRUARY 14, 2023

Posted by Kendra Byrne, Senior Product Manager, and Jie Tan, Staff Research Scientist, Robotics at Google

"Advances in large models across the field of AI have spurred a leap in capabilities for robot learning. This past year, we've seen the sense of context and sequencing of events captured in LLMs help solve long-horizon planning for robotics and make robots easier for people to interact with and task. ..."

[https://ai.googleblog.com/2023/02/google-research-2022-beyond-robotics.html]

“Tool Learning with Foundational Models”

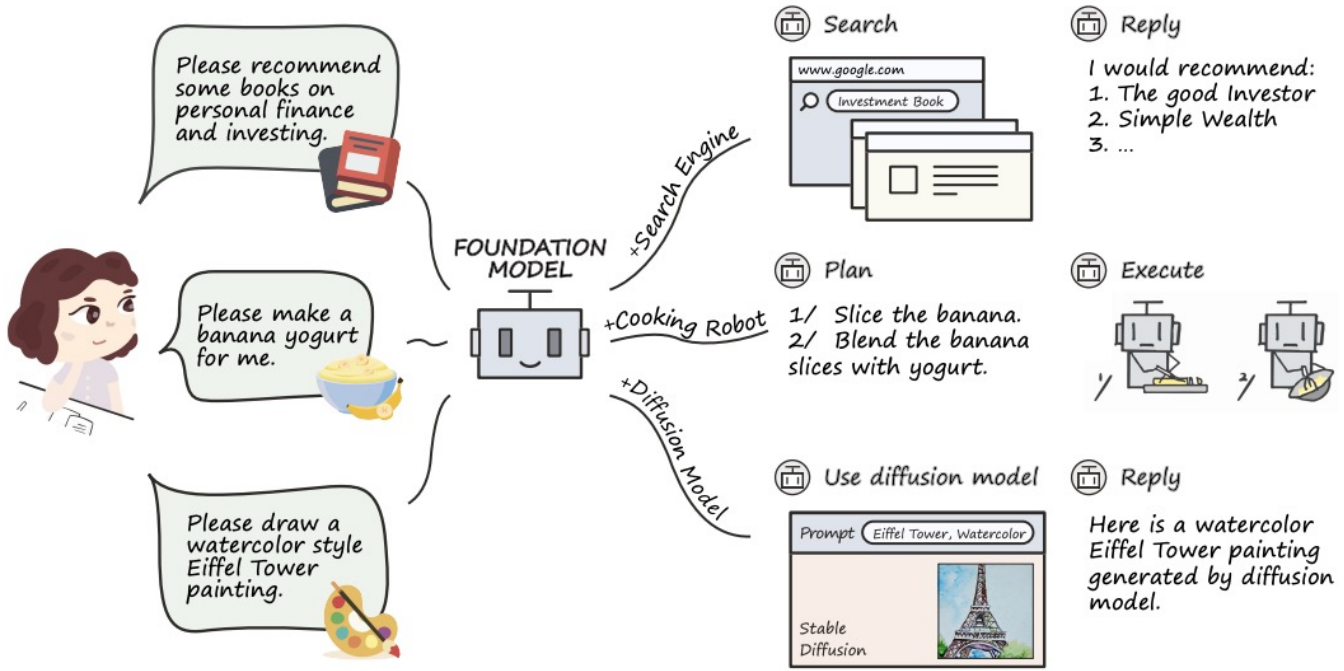
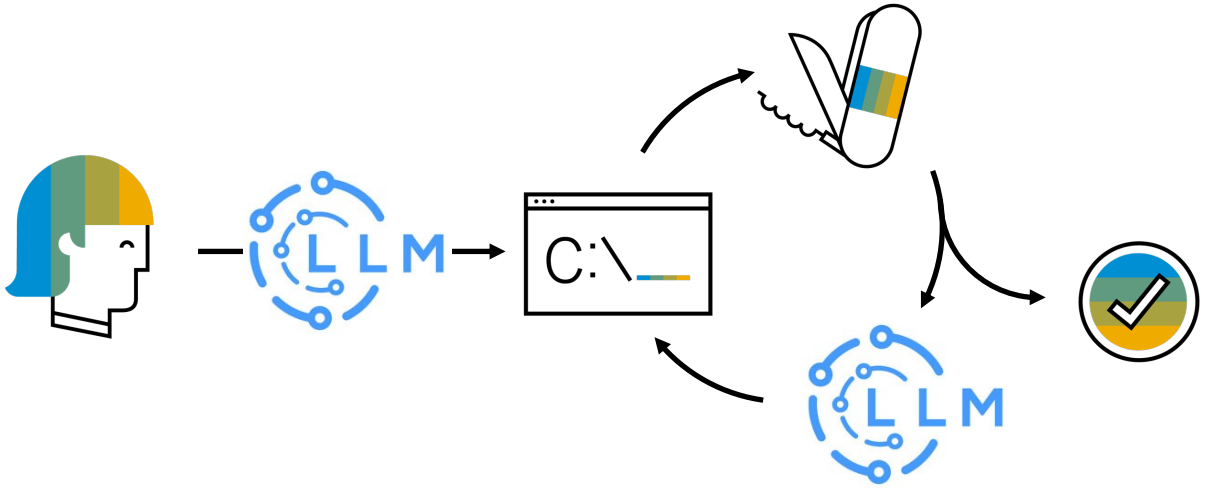


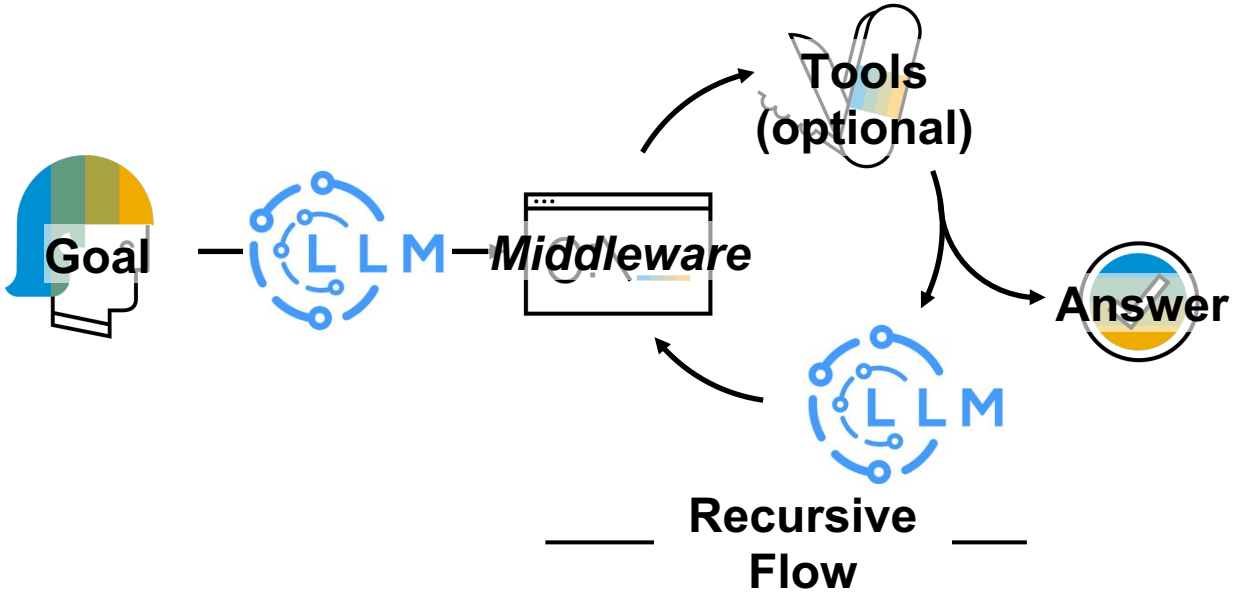
Figure 1: Tool learning paradigm aims to combine the strengths of specialized tools and foundation models.

["Tool Learning with Foundational Models"]

What are AI Agents?



What are AI Agents?



What are AI Agents?

AI Agent Framework:

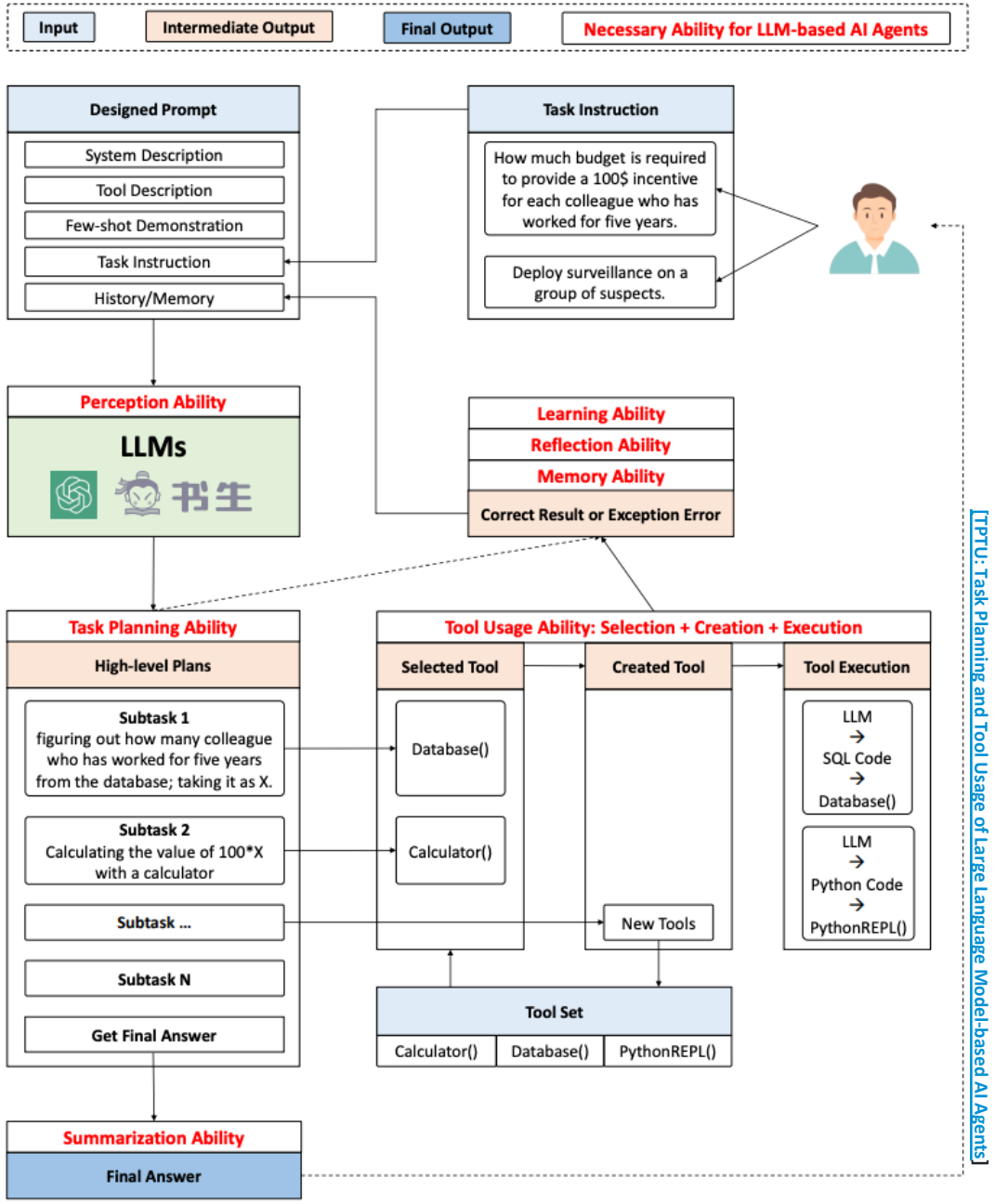


Figure 1: The proposed framework for LLM-based AI Agents.

What are AI Agents?

AI Agent Framework:

- Components:
  - Task Instruction
  - Designed Prompt
  - Tool Set
  - LLM
  - Intermediate Output/State
  - Final Answer

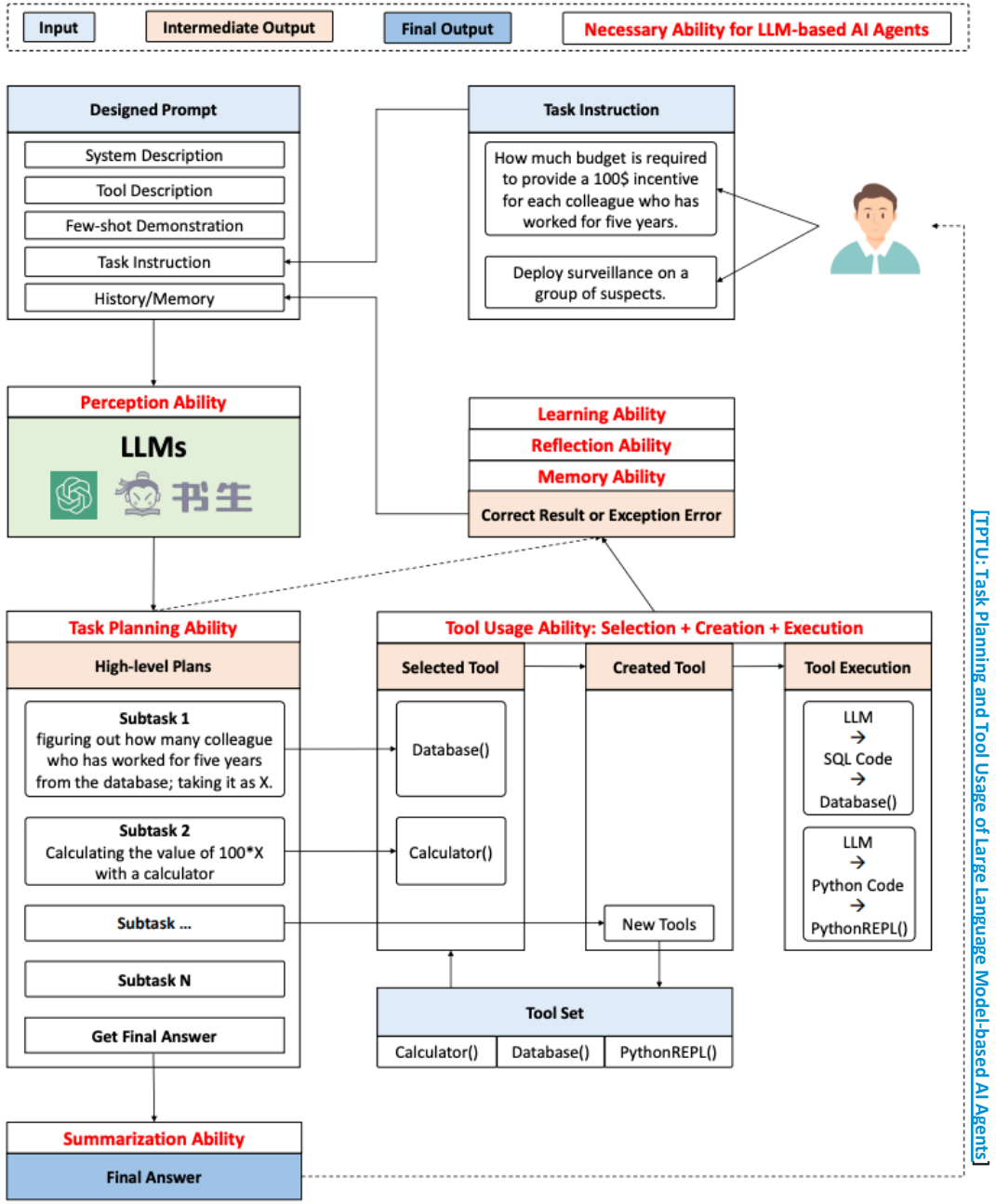


Figure 1: The proposed framework for LLM-based AI Agents.

What are AI Agents?

AI Agent Framework:

- Components:
  - Task Instruction
  - Designed Prompt
  - Tool Set
  - LLM
  - Intermediate Output/State
  - Final Answer
- Abilities
  - Perception
  - Task Planning
  - Tool Usage
  - Learning/Reflection/Memory
  - Summarization

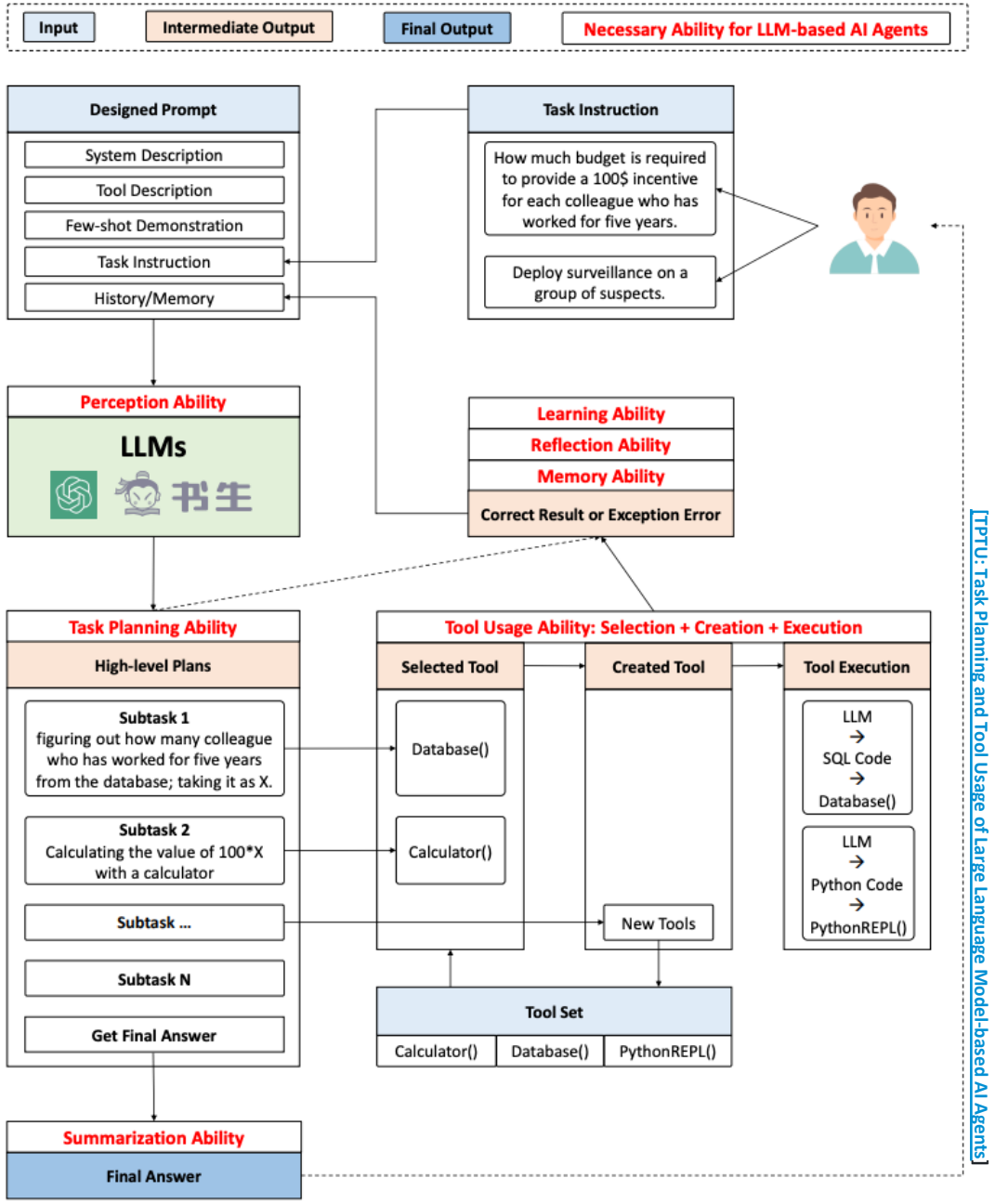


Figure 1: The proposed framework for LLM-based AI Agents.

What are AI Agents?

AI Agent Framework:

- Components:
  - Goal
    - Task Instruction
    - Designed Prompt
  - Recursive Flow/ Middleware
    - Tool Set
    - LLM
    - Intermediate Output/State
  - Answer
    - Final Answer
- Abilities
  - Goal
    - Perception
  - Recursive Flow/ Middleware
    - Task Planning
    - Tool Usage
    - Learning/Reflection/Memory
  - Answer
    - Summarization

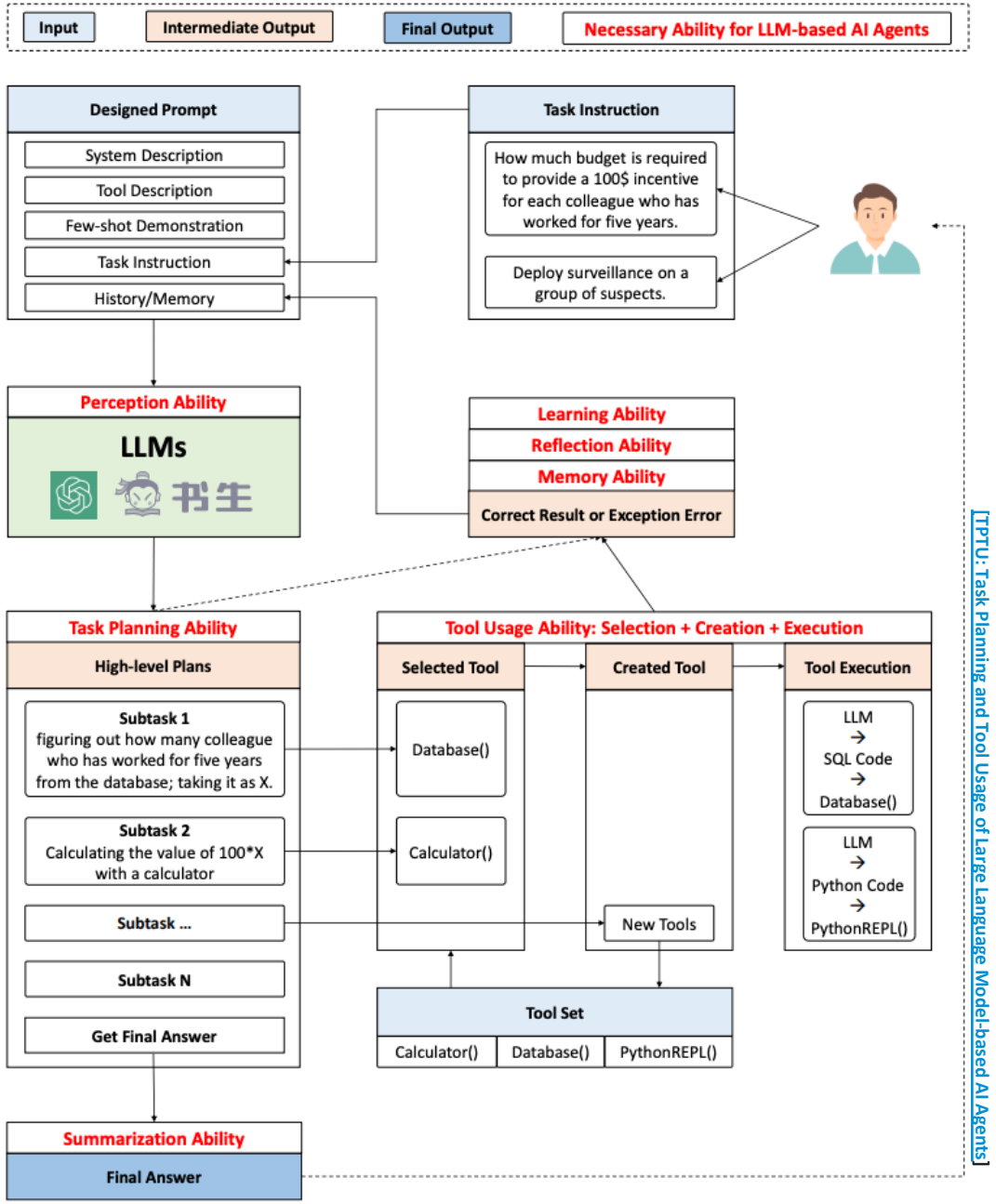


Figure 1: The proposed framework for LLM-based AI Agents.

What are AI Agents?

AI Agent Framework:

- Components:
  - Task Instruction
  - Designed Prompt
  - Tool Set **including specialed Agents**
  - LLM
  - Intermediate Output/State
  - Final Answer
- Abilities
  - Perception
  - Task Planning
  - Tool Usage
  - Learning/Reflection/Memory
  - Summarization

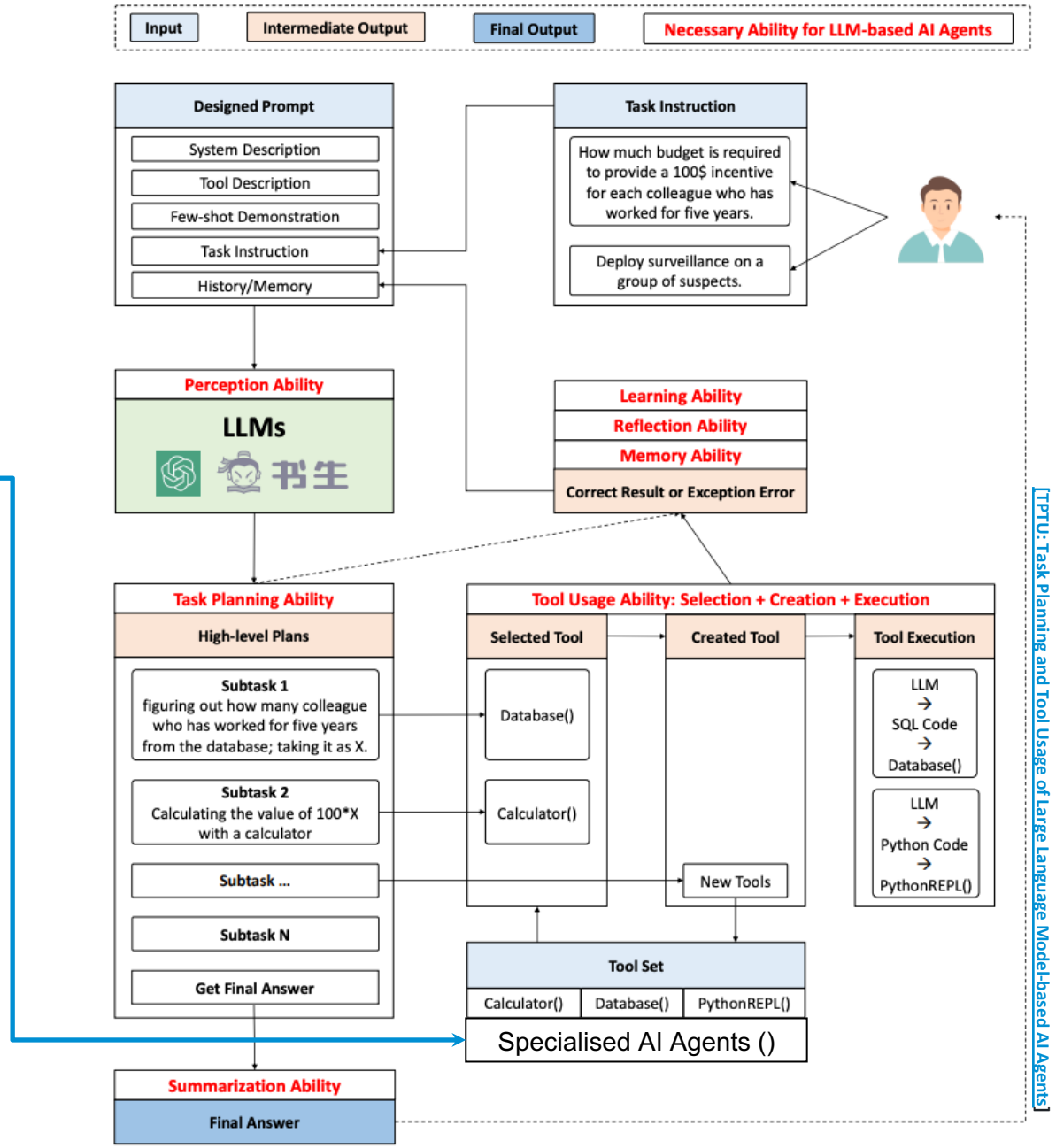
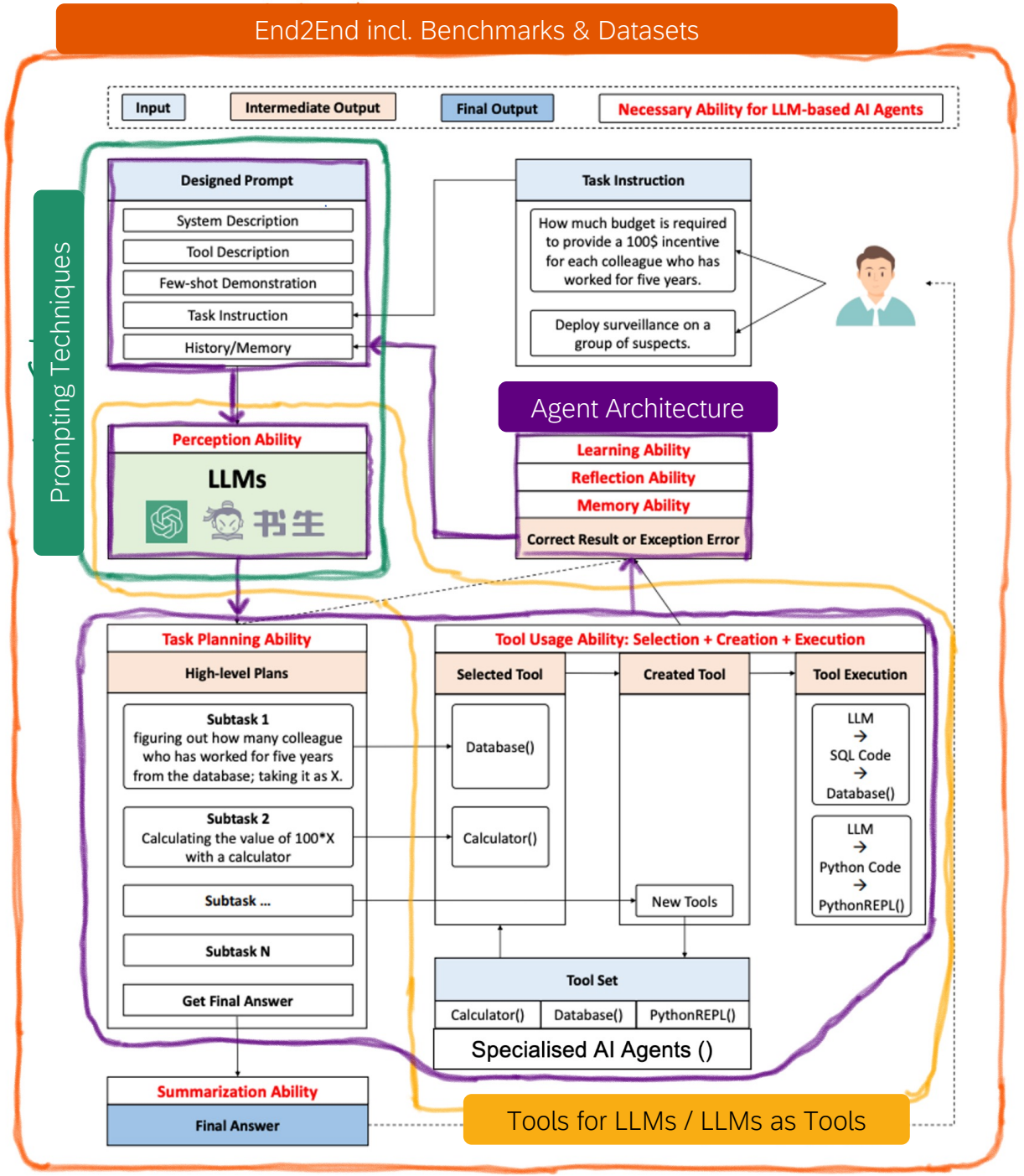


Figure 1: The proposed framework for LLM-based AI Agents.



AI Agent Landscape

Paper Categories:

End2End incl. Benchmarks & Datasets

Prompting Techniques

Agent Architecture

Tools for LLMs / LLMs as Tools

„Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models“ [Link]

1.Singapore Management University
2.Southwest Jiaotong University
3.Singapore University of Technology and Design
4.East China Normal University

Prompting Techniques

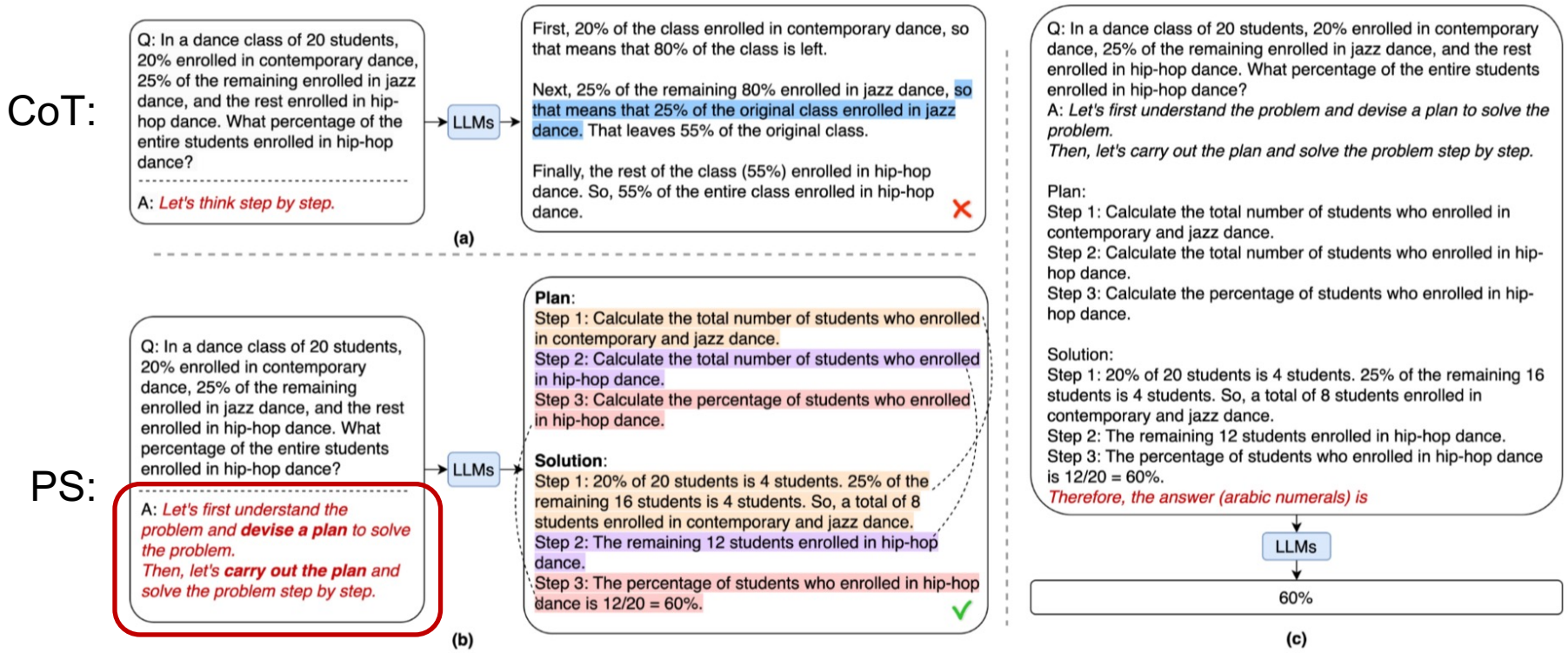


Figure 2: Example inputs and outputs of GPT-3 with (a) Zero-shot-CoT prompting, (b) Plan-and-Solve (PS) prompting, and (c) answer extraction prompting. While Zero-shot-CoT encourages LLMs to generate multi-step reasoning with "*Let's think step by step*", it may still generate wrong reasoning steps when the problem is complex. Unlike Zero-shot-CoT, PS prompting first asks LLMs to devise a plan to solve the problem by generating a step-by-step plan and carrying out the plan to find the answer.

„Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models“ [Link]

1.Singapore Management University
2.Southwest Jiaotong University
3.Singapore University of Technology and Design
4.East China Normal University

Prompting Techniques

Table 2: Accuracy comparison on six math reasoning datasets. The best and second best results are boldfaced and underlined respectively.

Setting	Method (text-davinci-003)	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Average
Zero-Shot	CoT	83.8	56.4	85.3	38.9	88.1	69.9	70.4
	PoT	**92.2**	57.0	85.1	<u>43.9</u>	<u>91.7</u>	70.8	<u>73.5</u>
	PS (ours)	87.2	<u>58.2</u>	<u>88.1</u>	42.5	89.2	<u>72.0</u>	72.9
	PS+ (ours)	<u>91.8</u>	**59.3**	**92.2**	**46.0**	**94.7**	**75.7**	**76.7**
Few-Shot	Manual-CoT	93.6	58.4	91.6	48.4	93.5	80.3	77.6
	Auto-CoT	95.5	57.1	90.8	41.7	92.1	78.1	75.9

Table 3: Accuracy on commonsense reasoning datasets.

Method	CSQA	StrategyQA
Few-Shot-CoT (Manual)	78.3	71.2
Zero-shot-CoT	65.2	63.8
Zero-shot-PS+ (ours)	**71.9**	**65.4**

Table 4: Accuracy on symbolic reasoning datasets.

Method	Last Letter	Coin Flip
Few-Shot-CoT (Manual)	70.6	100.0
Zero-shot-CoT	64.8	96.8
Zero-shot-PS+ (ours)	**75.2**	**99.6**

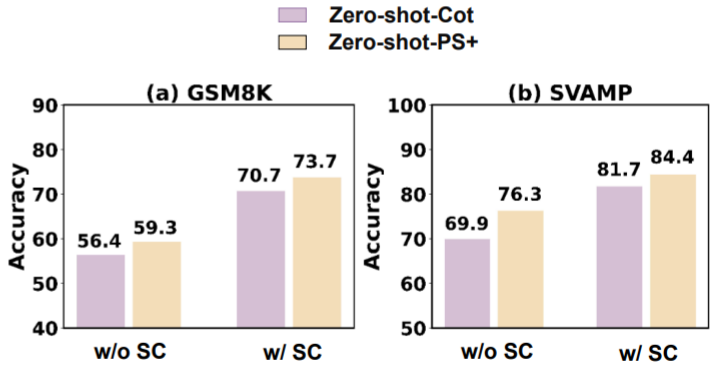


Figure 4: Results of methods with and without self-consistency (SC) on GSM8K and SVAMP.

„Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models“ [Link]

Prompting Techniques

Table 2: Accuracy comparison on six math reasoning datasets. The best and second best results are boldfaced and underlined

Setting	Average
Zero-Shot	70.4
	73.5
	72.9
	76.7
Few-Shot	77.6
	75.9

Formulating a complete plan before solving and giving the model a **chance to reflect** on it seems to **help the model reason significantly**.

Similar findings in:

“TPTU: Task Planning and Tool Usage of Large Language Model-based AI Agents” [Link]
“Reflexion: Language Agents with Verbal Reinforcement Learning” [Link]
“ReAct: Synergizing Reasoning and Acting in Language Models“ [Link]

Table 3: Ac

Metho		
Few-Shot-CoT (Manual)	78.3	71.2
Zero-shot-CoT	65.2	63.8
Zero-shot-PS+ (ours)	**71.9**	**65.4**

Table 4: Accuracy on symbolic reasoning datasets.

Method	Last Letter	Coin Flip
Few-Shot-CoT (Manual)	70.6	100.0
Zero-shot-CoT	64.8	96.8
Zero-shot-PS+ (ours)	**75.2**	**99.6**

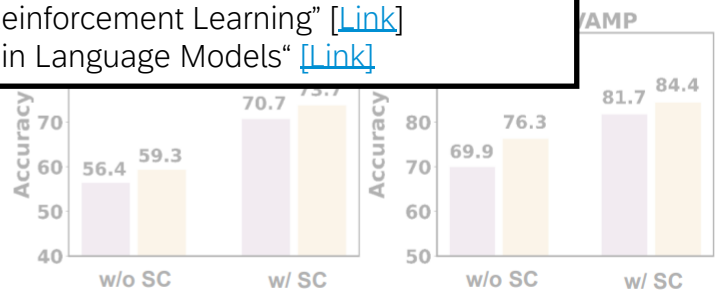
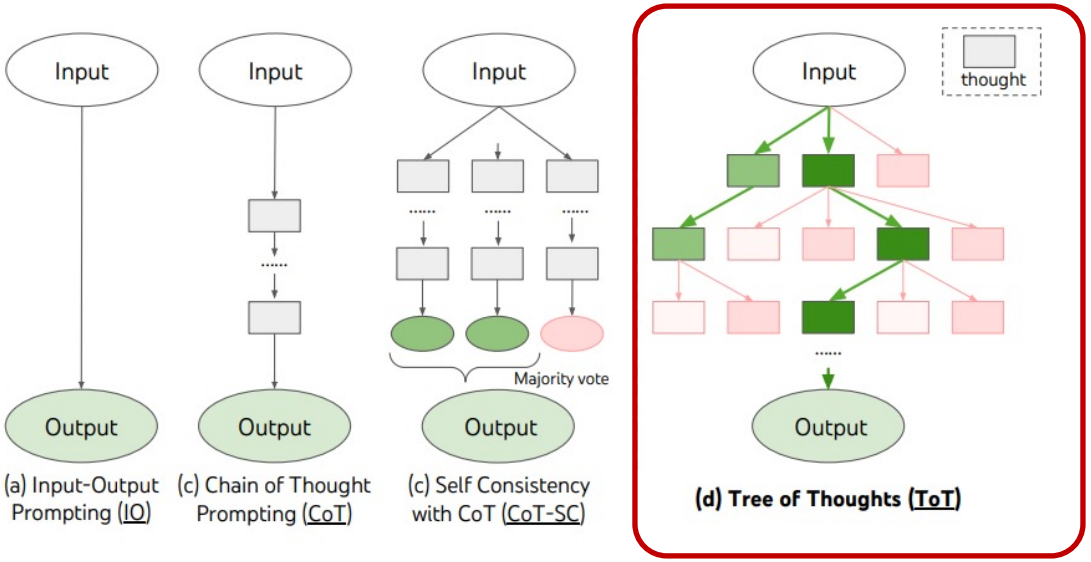


Figure 4: Results of methods with and without self-consistency (SC) on GSM8K and SVAMP.

„Tree of Thoughts: Deliberate Problem Solving with Large Language Models“ [Link]

1.Princeton
2.Google DeepMind

Prompting Techniques



Approach:

1. **Though decomposition + generator**
2. **Scoring/Ordering Nodes**
3. **Search algorithm (BFS vs. DFS)**

„Tree of Thoughts: Deliberate Problem Solving with Large Language Models“ [Link]

1.Princeton
2.Google DeepMind

Prompting Techniques

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

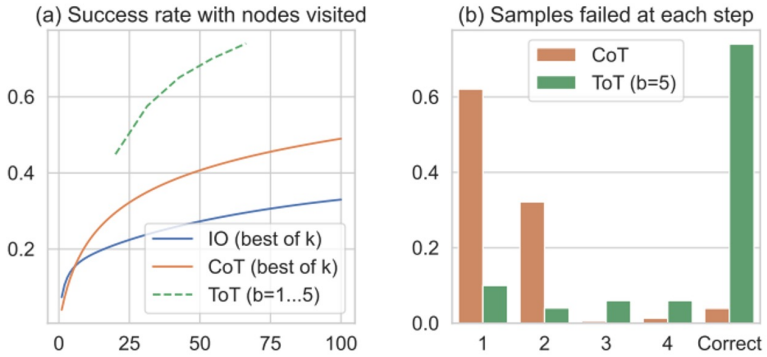


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

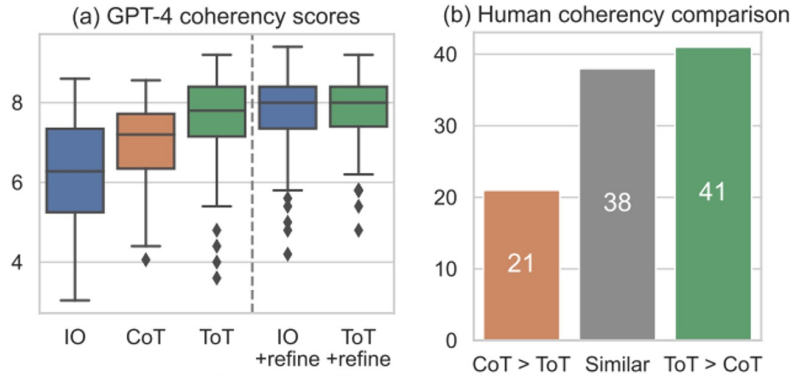


Figure 5: Creative Writing results.

Method	Success Rate (%) Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	**78**	**60**	**20**
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

„Tree of Thoughts: Deliberate Problem Solving with Large Language Models“ [Link]

Prompting Techniques

Method	Success
IO prompt	7.3%

Table 2: Game of 24 Results.

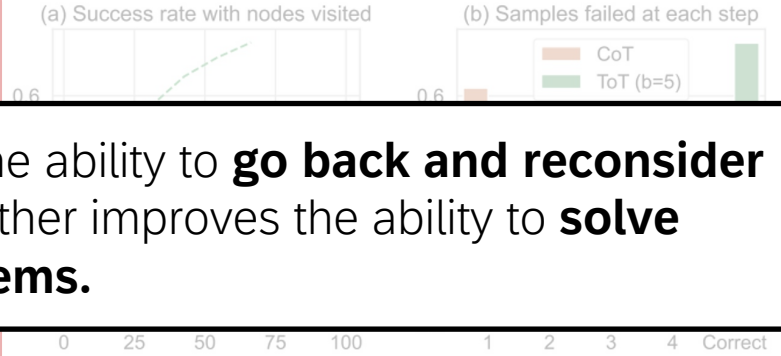


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Giving an LLM the ability to **go back and reconsider** old decisions further improves the ability to **solve complex problems.**

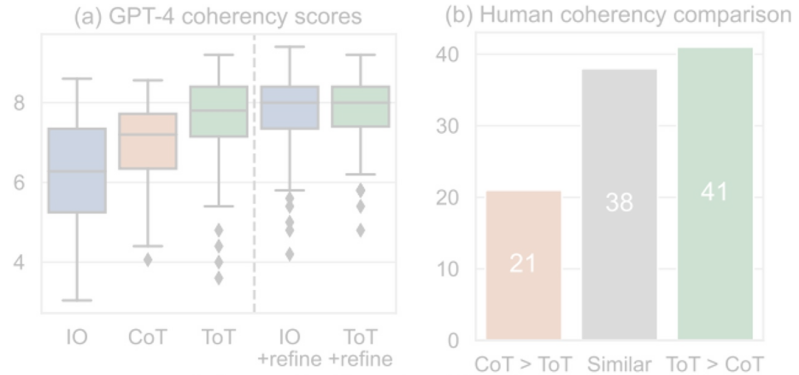


Figure 5: Creative Writing results.

Method	Success Rate (%) Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	**78**	**60**	**20**
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

„Toolformer: Language Models Can Teach Themselves to Use Tools“ [Link]

1.Meta AI
2.Universitat Pompeu Fabra

End2End

Tools for LLMs

Presentation by Michael in this series 17th March’23

The New England Journal of Medicine is a registered trademark of [QA(“Who is the publisher of The New England Journal of Medicine?”) → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from “la tortuga”, the Spanish word for [MT(“tortuga”) → turtle] turtle.

The Brown Act is California’s law [WikiSearch(“Brown Act”) → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

Fine-tuning of GPT-J (6.7B) to use tools.
Tool evaluation during generation.

„Tool Documentation Enables Zero-Shot Tool-Usage with Large Language Models“ [Link]

1.University of Washington
2.National Taiwan University
3.Google Cloud AI Research
4. Google Research

Tools for LLMs

Prompting Techniques

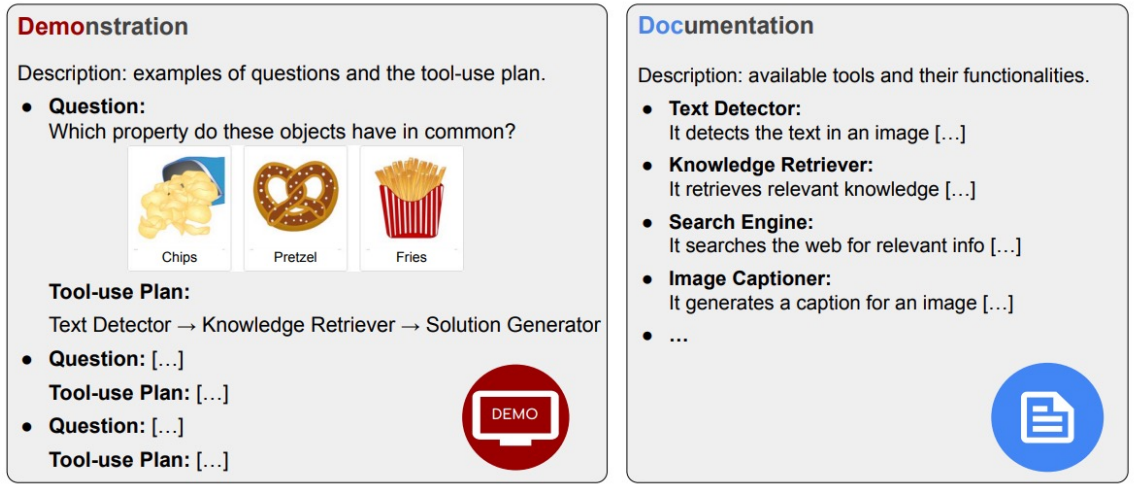


Figure 2: Two types of knowledge for prompting LLMs for tool-use: Demonstrations (demos) and Documentations (docs). Demos consist of <input, output> pairs on input instructions and their corresponding output tool-use plans. They require manual efforts for careful curation on every new task, and the model performance can be sensitive to which demos are used [81, 12]. Many demos may also be necessary for good coverage when the number of tools scales up. On the other hand, docs provide descriptions for the tool functionality, and are usually organically available for tools.

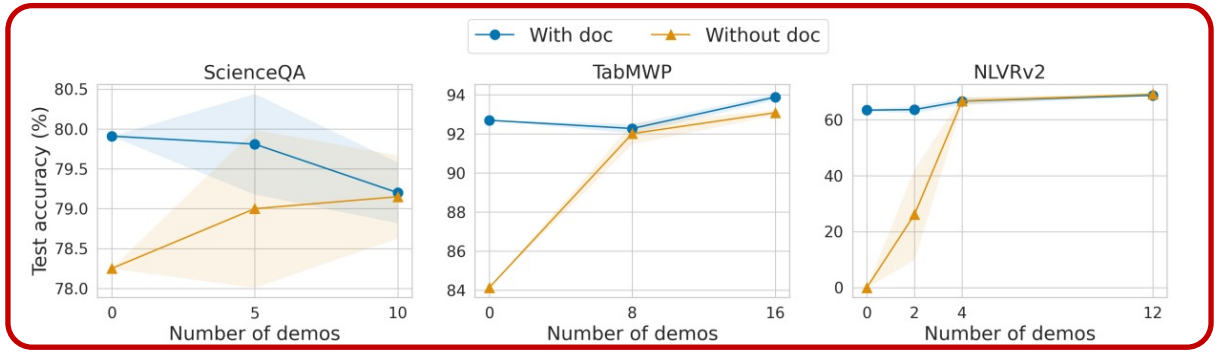


Figure 4: Tool-using performance with `gpt-3.5-turbo` on different benchmarks, which covers from langauge to vision modalities. We report results with and without documentation (doc) and demonstations (demo), and their combinations. Clearly, with documentation only (upper-left blue dot) shows competitive performance across all datasets.

Compare demo-based vs documentation-based tool-augmented LLMs for complex task solving

„Tool Documentation Enables Zero-Shot Tool-Usage with Large Language Models“ [Link]

1.University of Washington
2.National Taiwan University
3.Google Cloud AI Research
4. Google Research

Tools for LLMs

Prompting Techniques

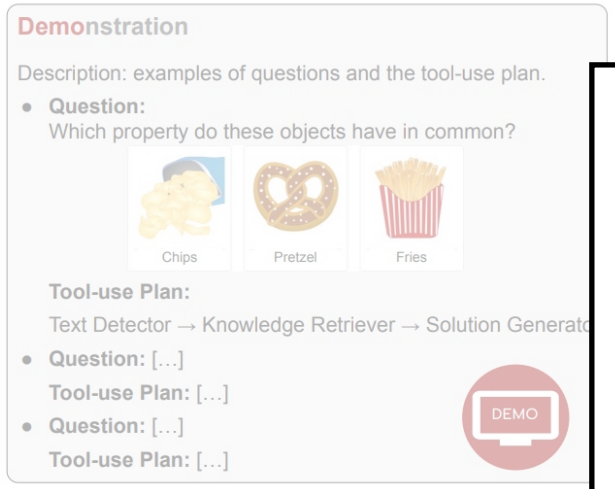
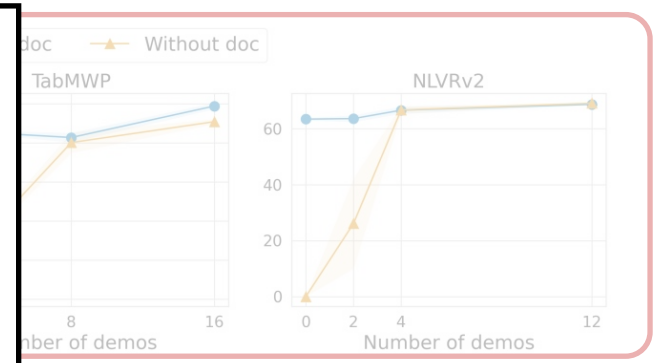


Figure 2: Two types of knowledge for prompting L
Documentations (docs). Demos consist of <inpu
corresponding output tool-use plans. They require
task, and the model performance can be sensitive
may also be necessary for good coverage when the number of tools scales up. On the other hand,
docs provide descriptions for the tool functionality, and are usually organically available for tools.

Tools can be reasonably well controlled **by in-context learning with LLMs >100B**.

The **usefulness of few-shot examples** seems to be **limited** and sometimes even limits the model's ability to use the tools.

(In constrast to the Toolformer: In-context learning required to decouple tool evaluation and LLM generation)



5-turbo on different benchmarks, which covers
results with and without documentation (doc) and
Clearly, with documentation only (upper-left blue
datasets.

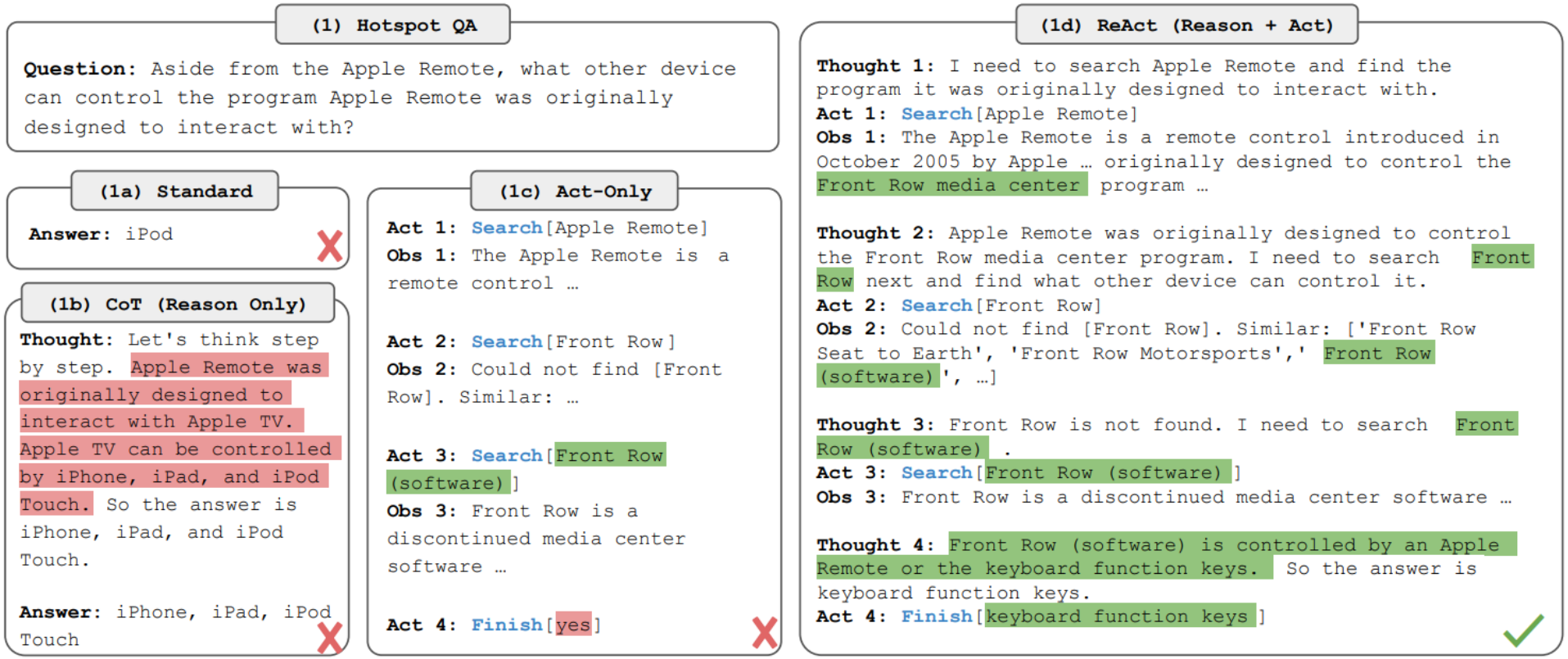
Compare demo-based vs documentation-based tool-augmented LLMs for complex task solving

„ReAct: Synergizing Reasoning and Acting in Language Models“ [Link]

Prompting Techniques

Agent Architecture

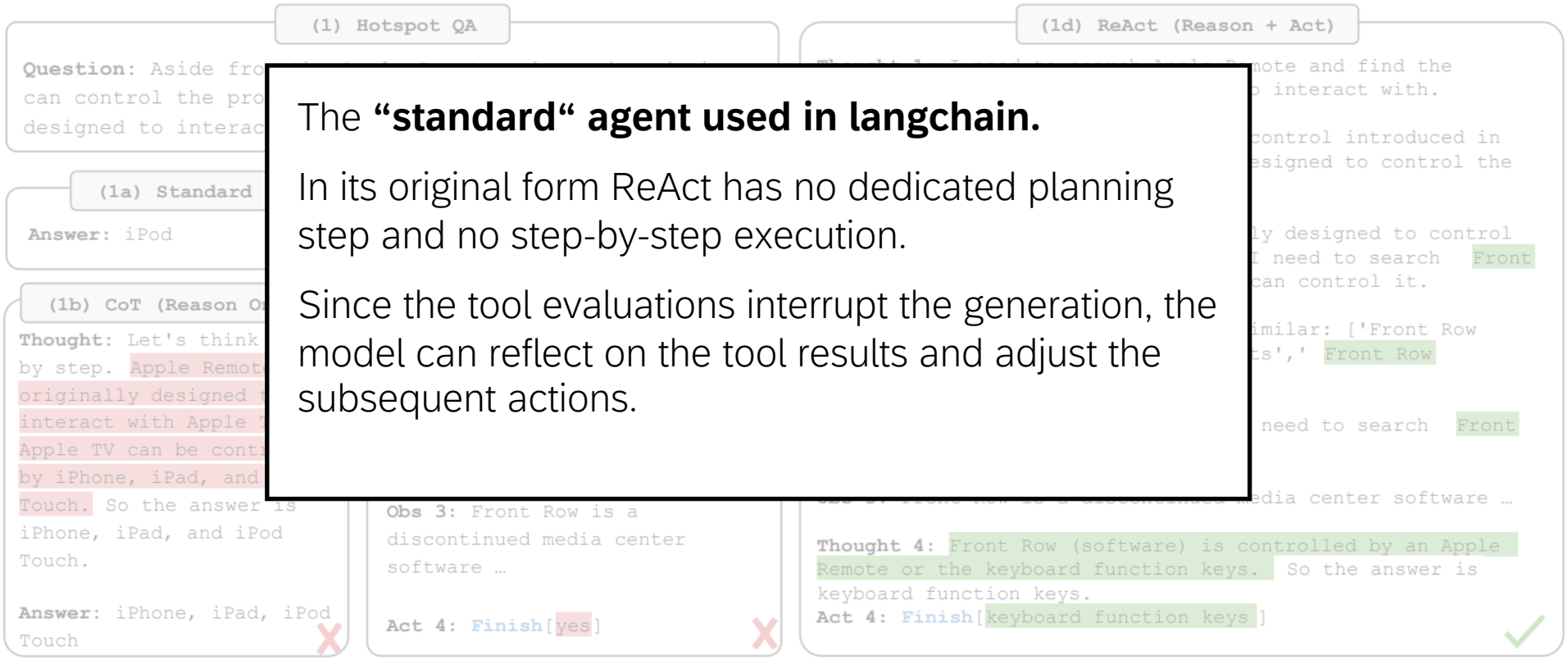
1.Princeton
2.Google Research



„ReAct: Synergizing Reasoning and Acting in Language Models“ [Link]

Prompting Techniques

Agent Architecture



The **“standard“ agent used in langchain.**

In its original form ReAct has no dedicated planning step and no step-by-step execution.

Since the tool evaluations interrupt the generation, the model can reflect on the tool results and adjust the subsequent actions.

„BOLAA: Benchmarking and Orchestrating LLM-augmented Autonomous Agents“ [Link]

1.Salesforce

Agent Architecture

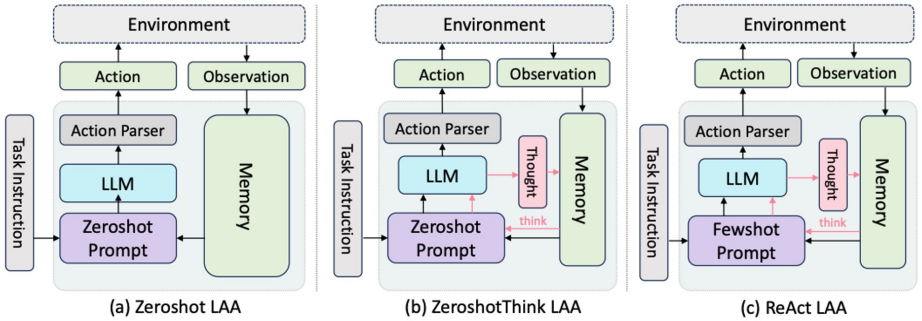


Figure 1: The LAA architectures for Zeroshot-LAA (ZS-LAA), ZeroshotThink LAA (ZST-LAA) and ReAct LAA. ZS-LAA generates actions from LLM with zeroshot prompt. ZST-LAA extends ZS-LAA with self-think. ReAct LAA advances ZST-LAA with fewshot prompt. They all resolve a given task by interacting with environment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

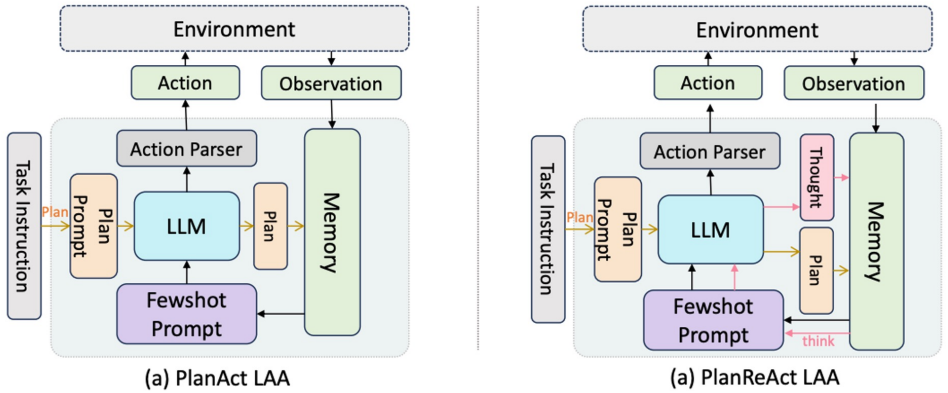


Figure 2: The LAA architectures for PlanAct LAA and PlanReAct LAA.

End2End Evaluation

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
fastchat-t5-3b	2k	0.0252	0.0067	0.0692	**0.1155**	0.0834
vicuna-7b	2k	**0.1339**	0.0797	0.0318	0.0868	0.0956
vicuna-13b	2k	0.1541	0.0910	**0.2637**	0.1754	0.2075
vicuna-33b	2k	0.2180	0.2223	**0.2602**	0.1333	0.2016
llama-2-7b	4k	0.0395	0.0207	**0.2624**	0.1780	0.1417
llama-2-13b	4k	0.1731	0.2313	**0.2521**	0.2192	0.2177
llama-2-70b	4k	0.2809	0.3207	**0.3558**	0.1424	0.1797
mpt-7b-instruct	8k	0.0982	0.0483	**0.1707**	0.1147	0.1195
mpt-30b-instruct	8k	0.1562	0.2141	**0.3261**	0.2224	0.2315
xgen-8k-7b-instruct	8k	0.1502	0.1244	**0.1937**	0.1116	0.1096
longchat-7b-16k	16k	0.0791	0.0672	**0.2161**	0.1296	0.0971
longchat-13b-16k	16k	0.1083	0.0562	**0.2387**	0.1623	0.1349
text-davinci-003	4k	0.3430	0.3304	**0.4503**	0.3577	0.4101
gpt-3.5-turbo	4k	**0.3340**	0.3254	0.3226	0.2762	0.3192
gpt-3.5-turbo-16k-0613	16k	**0.3027**	0.2264	0.1859	0.2113	0.2251

„BOLAA: Benchmarking and Orchestrating LLM-augmented Autonomous Agents“ [Link]

1.Salesforce

Agent Architecture

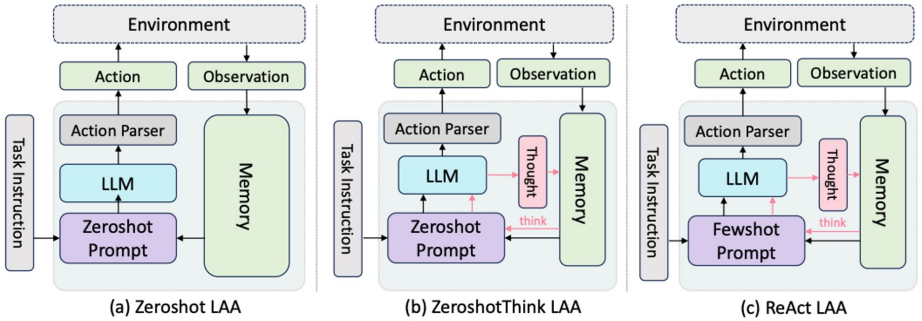


Figure 1: The LAA architectures for Zeroshot-LAA (ZS-LAA), ZeroshotThink LAA (ZST-LAA) and ReAct LAA. ZS-LAA generates actions from LLM with zeroshot prompt. ZST-LAA extends ZS-LAA with self-think. ReAct LAA advances ZST-LAA with fewshot prompt. They all resolve a given task by interacting with environment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

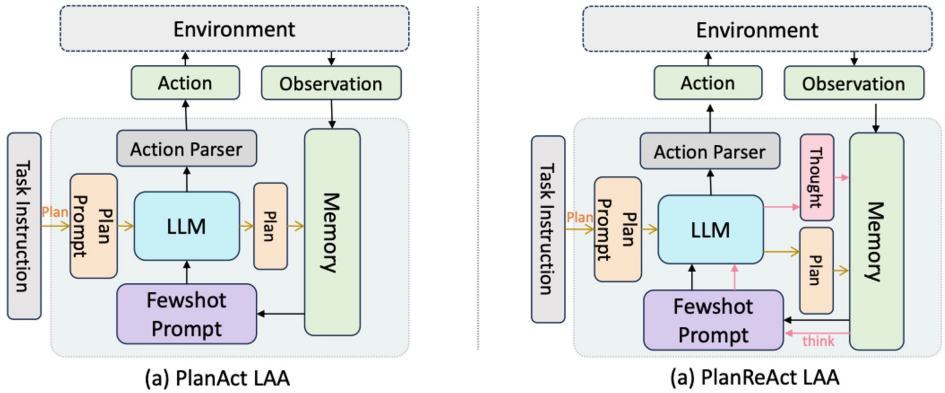


Figure 2: The LAA architectures for PlanAct LAA and PlanReAct LAA.

End2End Evaluation

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	4k	0.0042	0.0068	0.1248	0.3156	0.2761	**0.4648**
llama-2-13b	4k	0.0662	0.0420	0.2568	**0.4892**	0.4091	0.3716
llama-2-70b	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**

„BOLAA: Benchmarking and Orchestrating LLM-augmented Autonomous Agents“ [Link]

Agent Architecture

End2End Evaluation

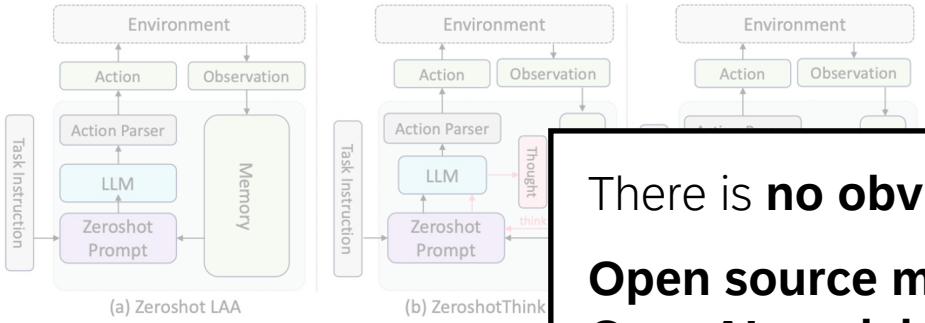


Figure 1: The LAA architectures for Zeroshot-LAA (ZS- and ReAct LAA. ZS-LAA generates actions from LLM w ZS-LAA with self-think. ReAct LAA advances ZST-LAA given task by interacting with environment via actions to c

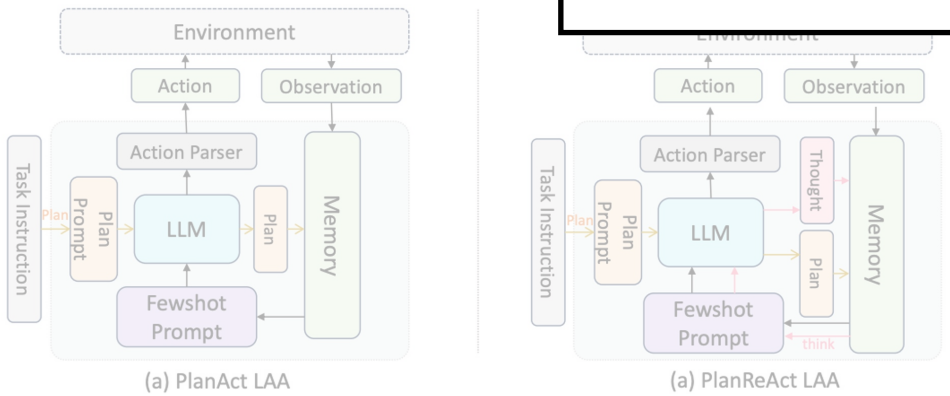


Figure 2: The LAA architectures for PlanAct LAA and PlanReAct LAA.

There is **no obvious best agent architecture**.

Open source models can‘t compete with the OpenAI models when used for AI agents.

Similar findings in:

“AgentBench: Evaluating LLMs as Agents” [Link]

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 . best LLM regarding one LAA architecture.

		LAA Architecture				
		eAct	PlanAct	PlanReAct	BOLAA
		3098	0.3837	0.1507	0.5169
		1033	0.0555	0.0674	0.0604
		1509	0.3120	0.4127	0.5350
		1887	0.3692	0.3125	0.5612
		1248	0.3156	0.2761	0.4648
		2568	**0.4892**	0.4091	0.3716
		4426	0.2979	0.3770	0.5040
		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

„AgentBench Evaluation LLMs as AI Agents“ [Link]

1.Tsinghua University
2.Ohio State University
3.UC Berkeley

End2End Evaluation

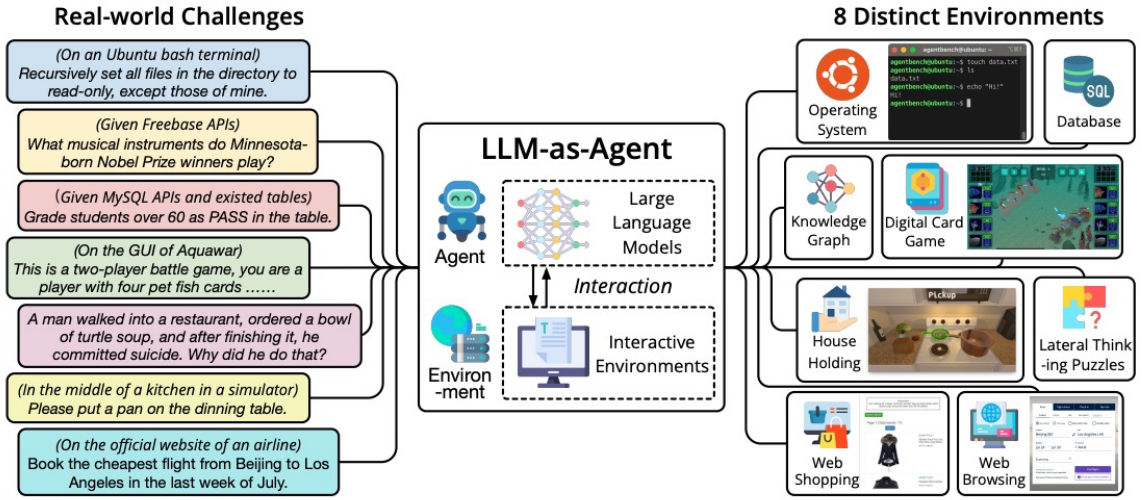
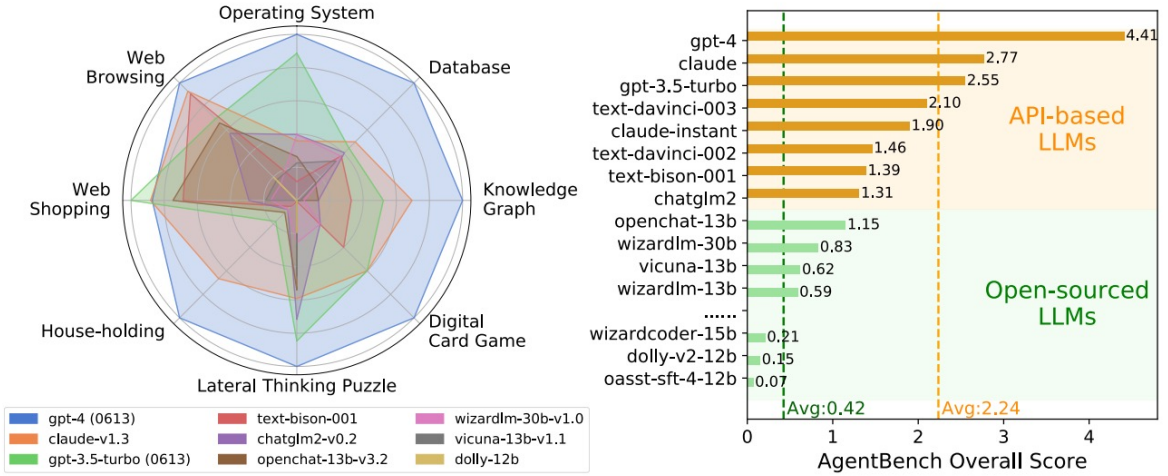


Figure 2: AgentBench is the first systematic benchmark to evaluate LLM-as-Agent on a wide array of real-world challenges and 8 distinct environments. In total, 25 LLMs are examined in its first edition.



(a) Typical LLMs' AgentBench performance (relative) against the best in each environment.

(b) Overall scores of AgentBench across 8 environ -ments. Dashed lines for two LLM types' average.

Figure 1: An overview of LLMs on AgentBench. While LLMs begin to manifest their proficiency in LLM-as-Agent, gaps between models and the distance towards practical usability are significant.

„AgentBench Evaluation LLMs as AI Agents“ [Link]

End2End Evaluation

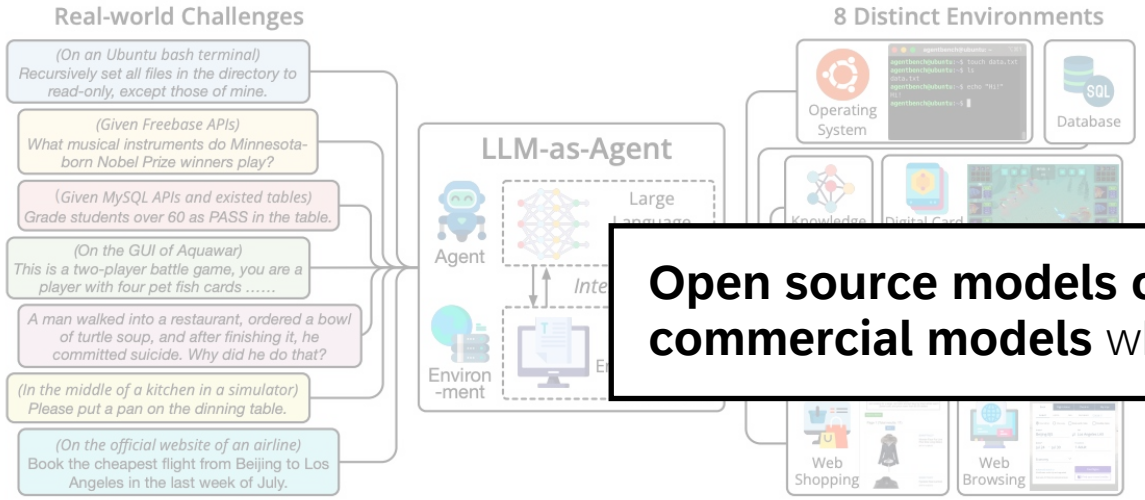
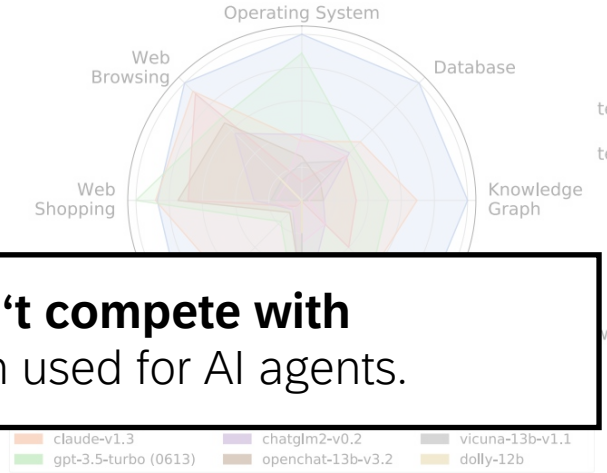
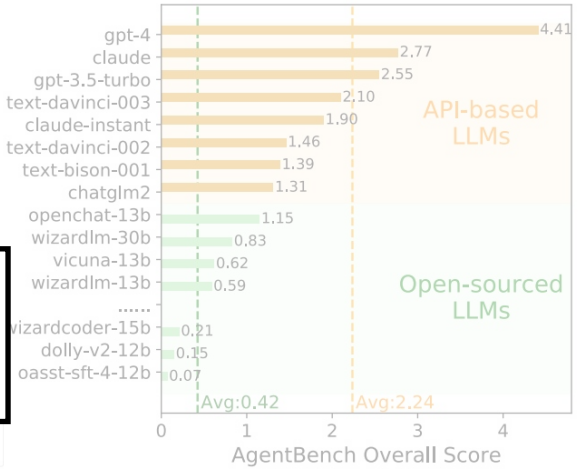


Figure 2: AgentBench is the first systematic benchmark to evaluate LLM-as-Agent on a wide array of real-world challenges and 8 distinct environments. In total, 25 LLMs are examined in its first edition.

Open source models can‘t compete with commercial models when used for AI agents.



(a) Typical LLMs' AgentBench performance (relative) against the best in each environment.



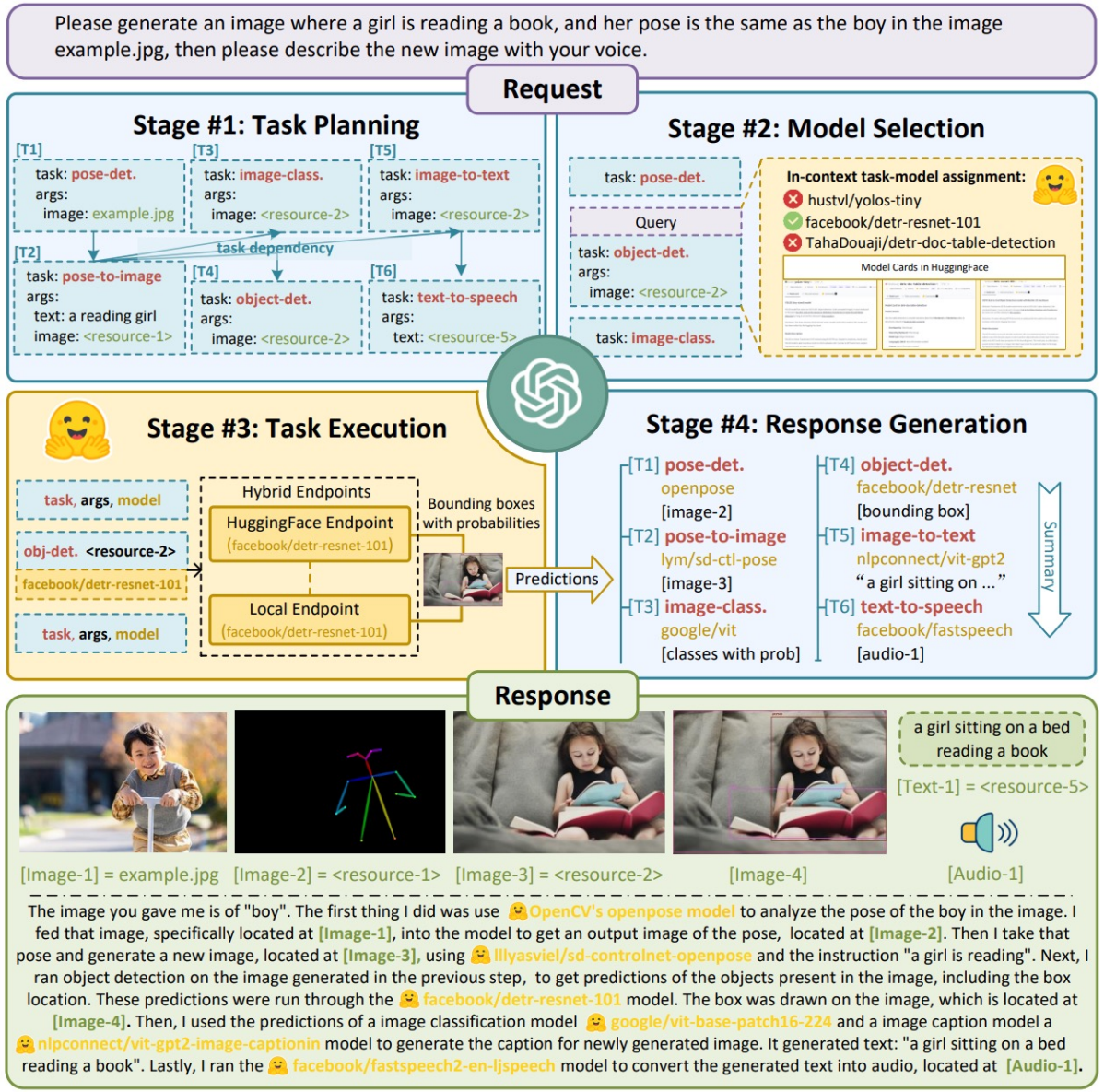
(b) Overall scores of AgentBench across 8 environ -ments. Dashed lines for two LLM types' average.

Figure 1: An overview of LLMs on AgentBench. While LLMs begin to manifest their proficiency in LLM-as-Agent, gaps between models and the distance towards practical usability are significant.

„HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face “ [Link]

Tools for LLMs / LLMs as Tools

1.Zhejiang University
2.Microsoft Research Asia

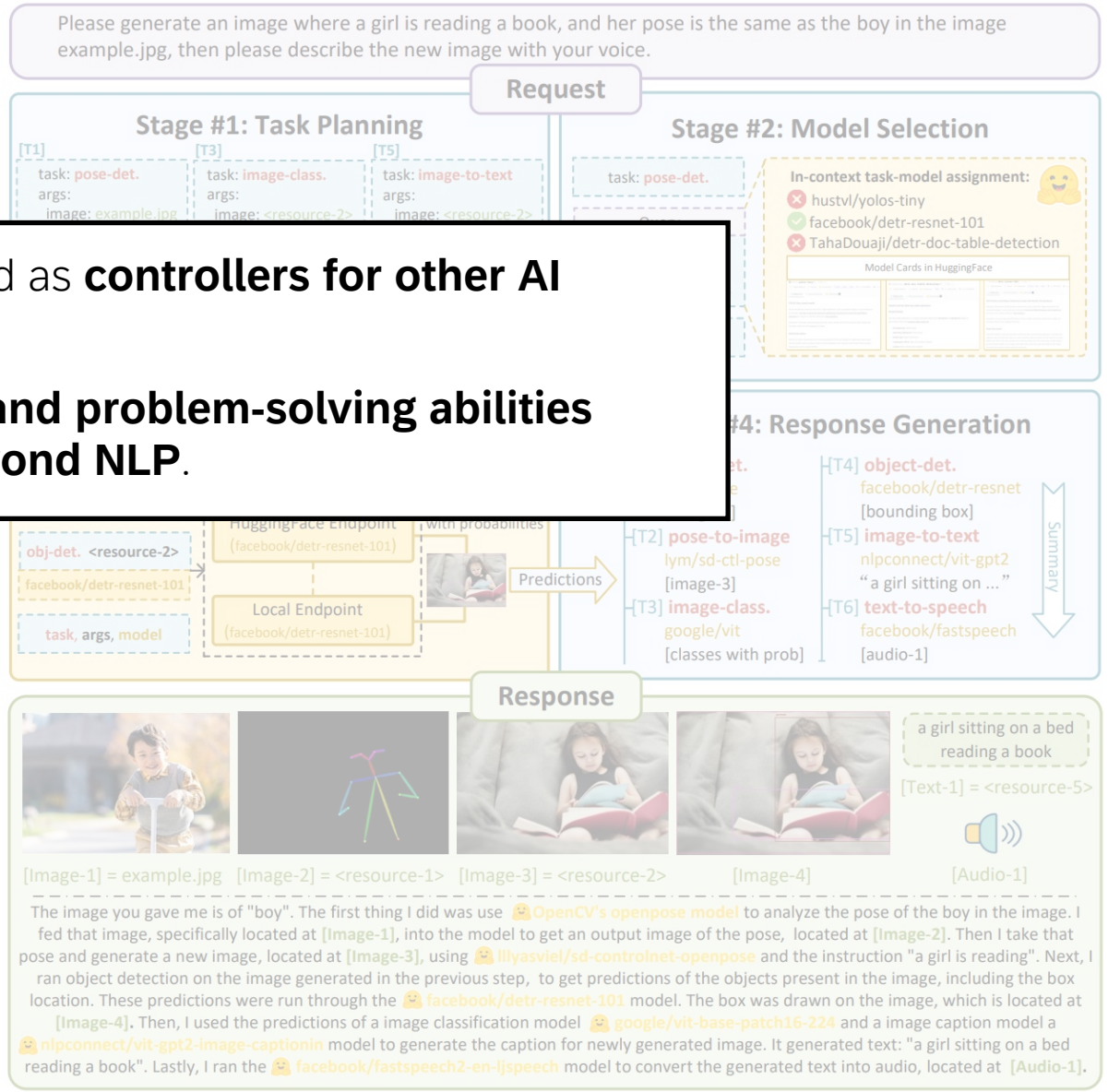


„HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face “ [Link]

Tools for LLMs / LLMs as Tools

LLMs can be used as **controllers for other AI models**.

Their **reasoning and problem-solving abilities** work in areas **beyond NLP**.



„Voyager: An Open-Ended Embodied Agent with Large Language Models“

[Link]

1.NVIDIA
2.Caltech
3.UT Austin
4.Stanford
5.ASU

Agent Architecture

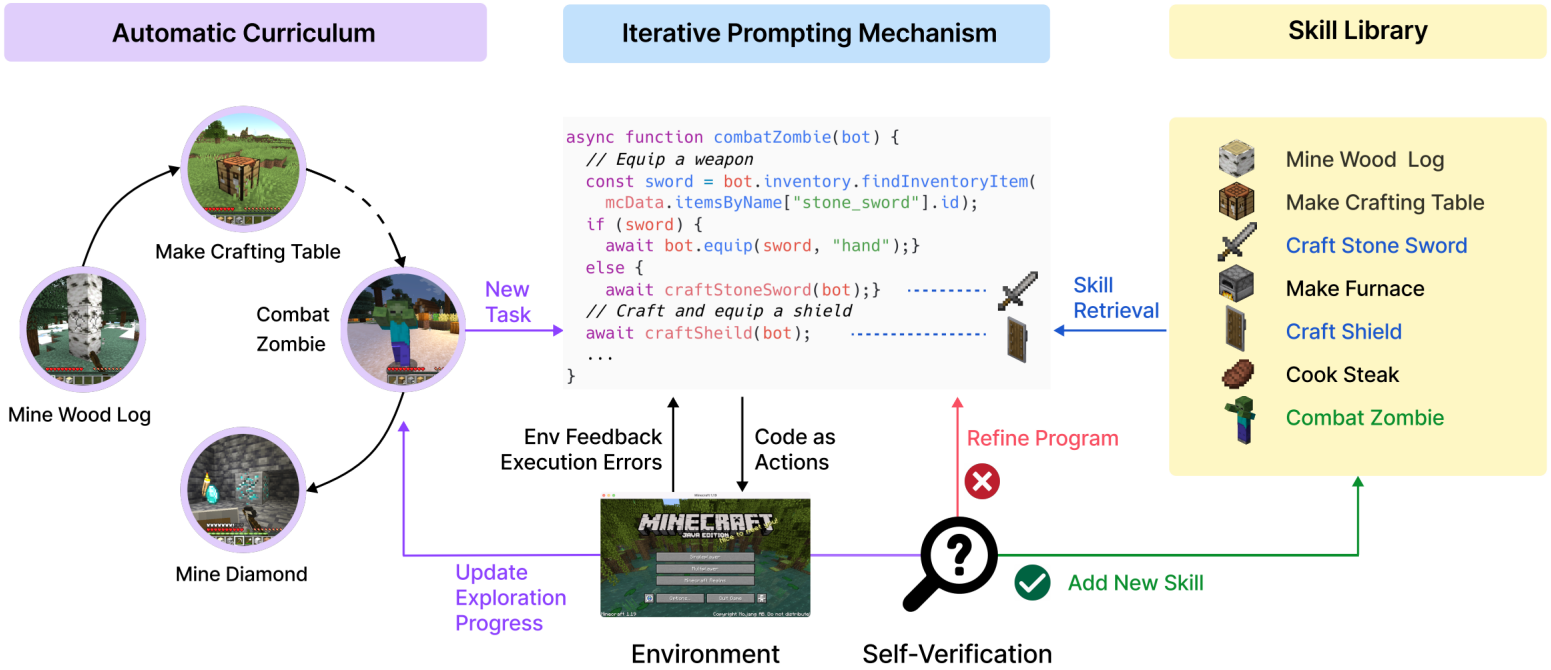


Figure 2: VOYAGER consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

„Voyager: An Open-Ended Embodied Agent with Large Language Models“

[Link]

1.NVIDIA
2.Caltech
3.UT Austin
4.Stanford
5.ASU

Agent Architecture

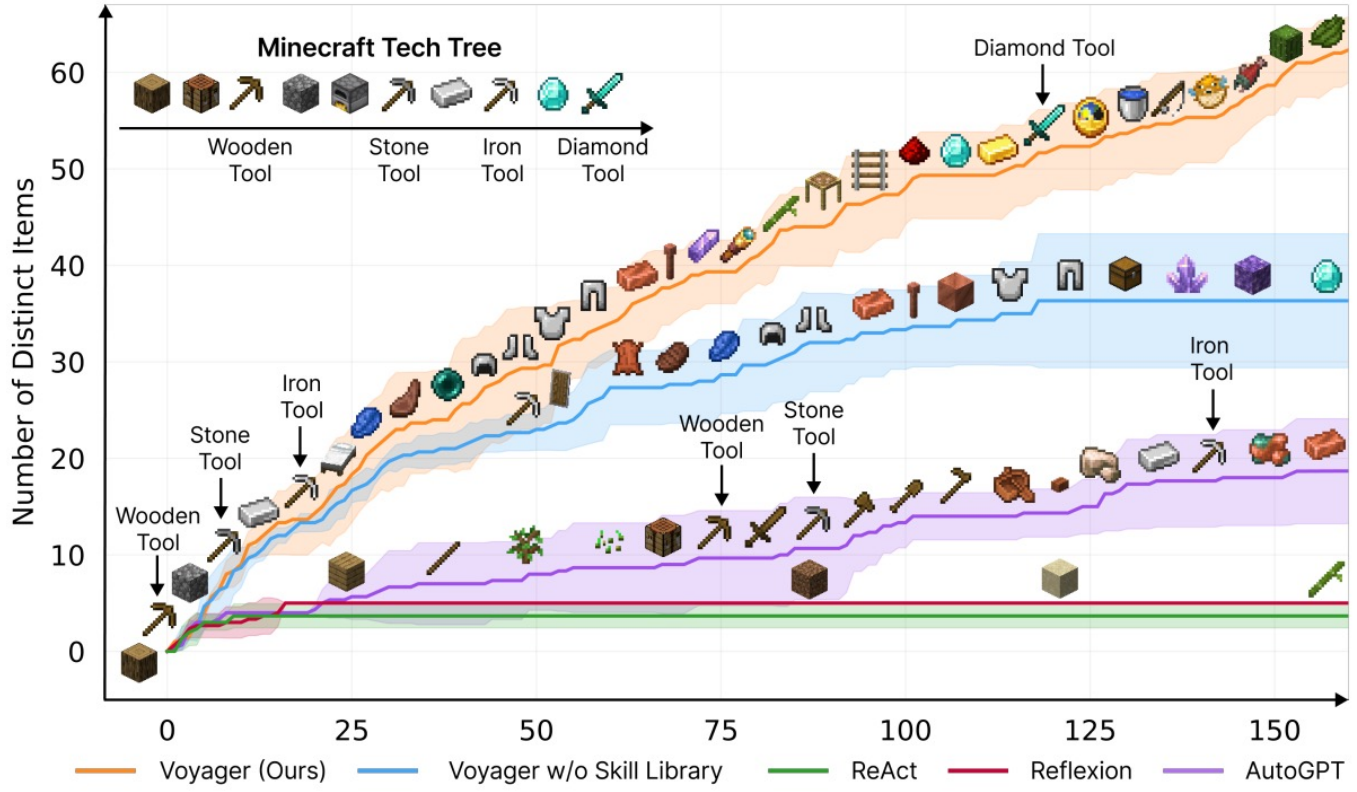


Figure 1: VOYAGER discovers new Minecraft items and skills continually by self-driven exploration, significantly outperforming the baselines. X-axis denotes the number of prompting iterations.

„Voyager: An Open-Ended Embodied Agent with Large Language Models“ [Link]

Agent Architecture

LLMs‘ **reasoning and problem-solving abilities** can even work for embodied agents! [Google‘s pivot in robotics]

Skill library for agents with **live long learning** capabilities.

Curriculum combined with a skill library enables self-improving agents.

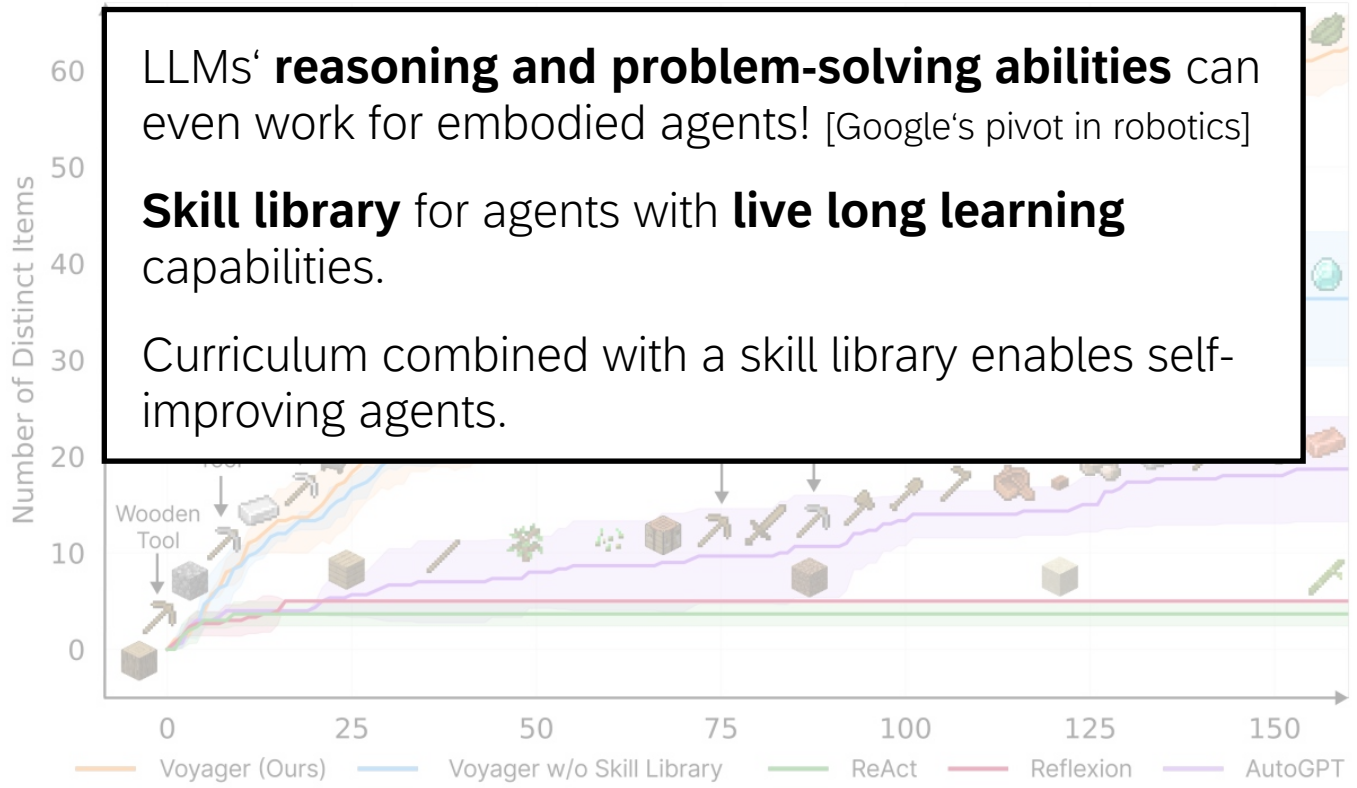


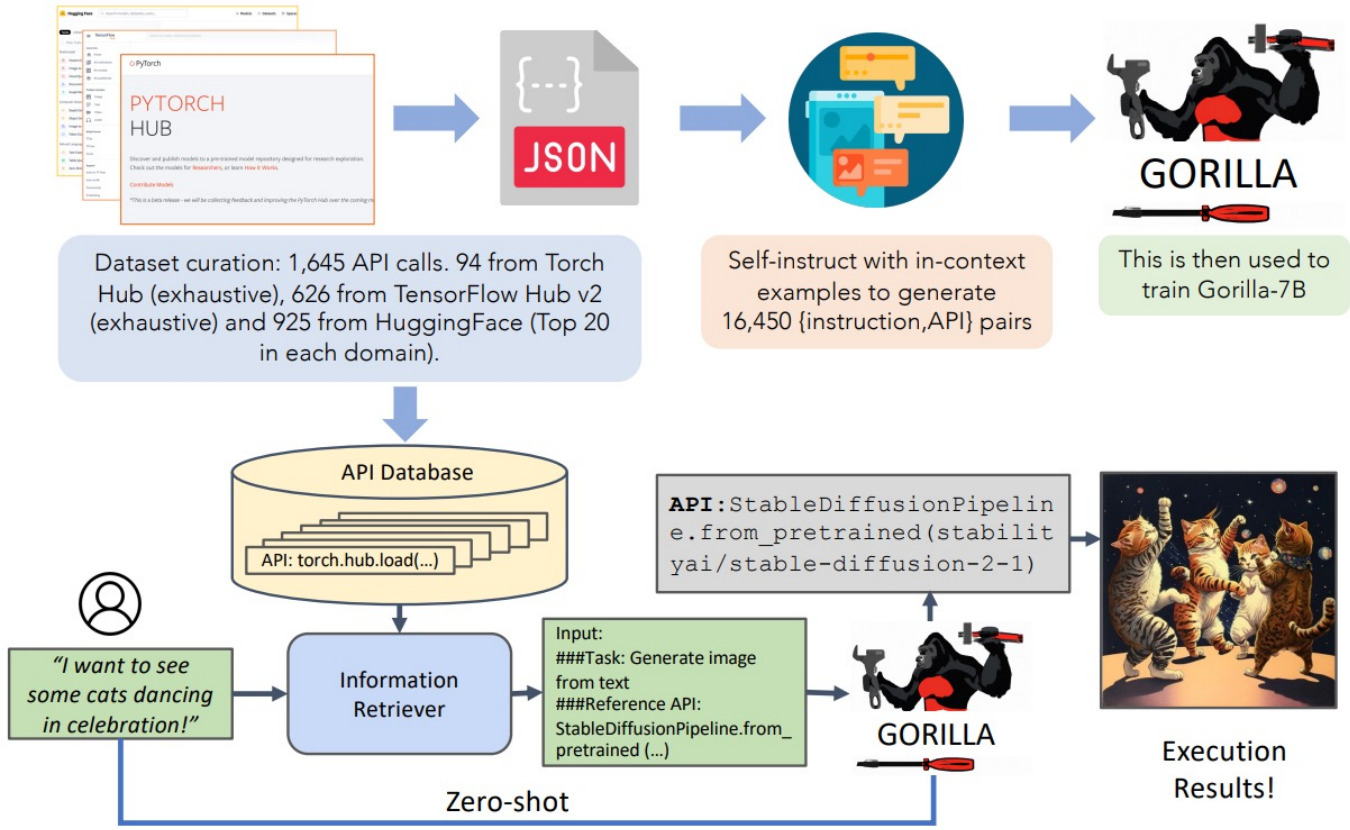
Figure 1: VOYAGER discovers new Minecraft items and skills continually by self-driven exploration, significantly outperforming the baselines. X-axis denotes the number of prompting iterations.

„Gorilla: Large Language Model Connected with Massive APIs“ [Link]

1.UC Berkeley
2.Microsoft Research

End2End

Tools for LLMs / LLMs as Tools



Fine-tuning LLaMA (7B) for specific APIs with API retriever used during training

„Gorilla: Large Language Model Connected with Massive APIs“ [Link]

1.UC Berkeley
2.Microsoft Research

End2End

Tools for LLMs / LLMs as Tools

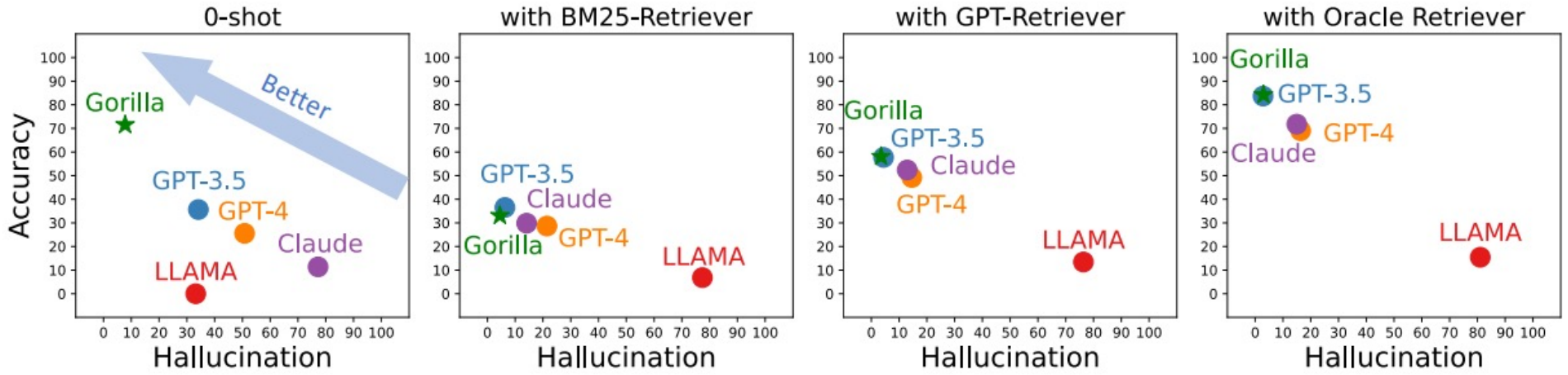
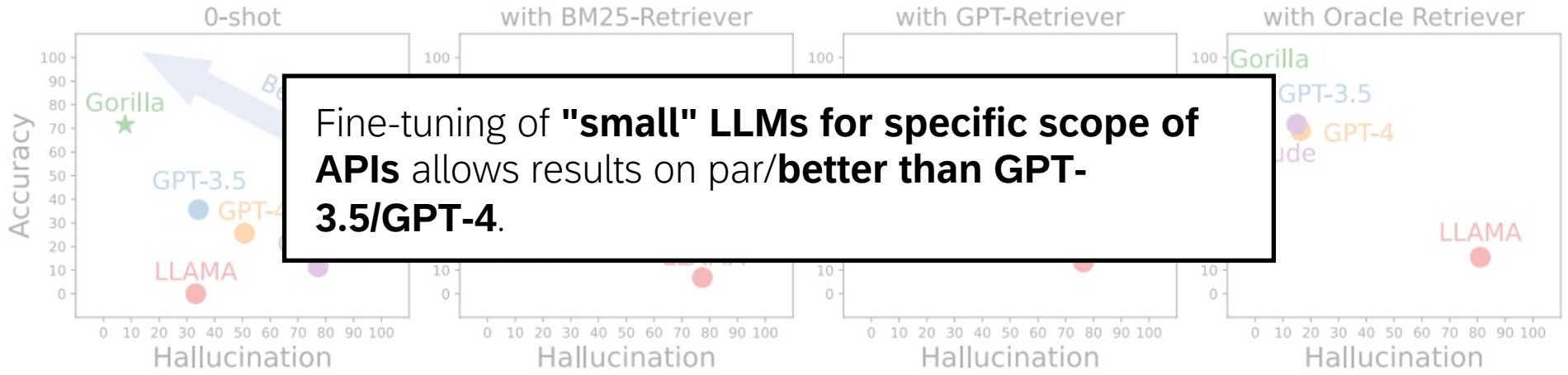


Figure 2: **Accuracy (vs) hallucination** in four settings, that is, *zero-shot* (i.e., without any retriever), and *with retrievers*. `BM25` and `GPT` are commonly used retrievers and the `oracle` retriever returns relevant documents at 100%, indicating an upper bound. Higher in the graph (higher accuracy) and to the left is better (lower hallucination). Across the entire dataset, our model, Gorilla, improves accuracy while reducing hallucination.

„Gorilla: Large Language Model Connected with Massive APIs“ [Link]

End2End

Tools for LLMs / LLMs as Tools



Fine-tuning of **"small" LLMs for specific scope of APIs** allows results on par/**better than GPT-3.5/GPT-4**.

Figure 2: **Accuracy (vs) hallucination** in four settings, that is, *zero-shot* (i.e., without any retriever), and *with retrievers*. BM25 and GPT are commonly used retrievers and the `oracle` retriever returns relevant documents at 100%, indicating an upper bound. Higher in the graph (higher accuracy) and to the left is better (lower hallucination). Across the entire dataset, our model, Gorilla, improves accuracy while reducing hallucination.

„ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs“ [Link]

End2End

Tools for LLMs / LLMs as Tools

1.Tsinghua University ModelBest Inc.
2.Renmin University of China
3.Yale University
4.WeChat AI, Tencent Inc.
5.Zhihu Inc.

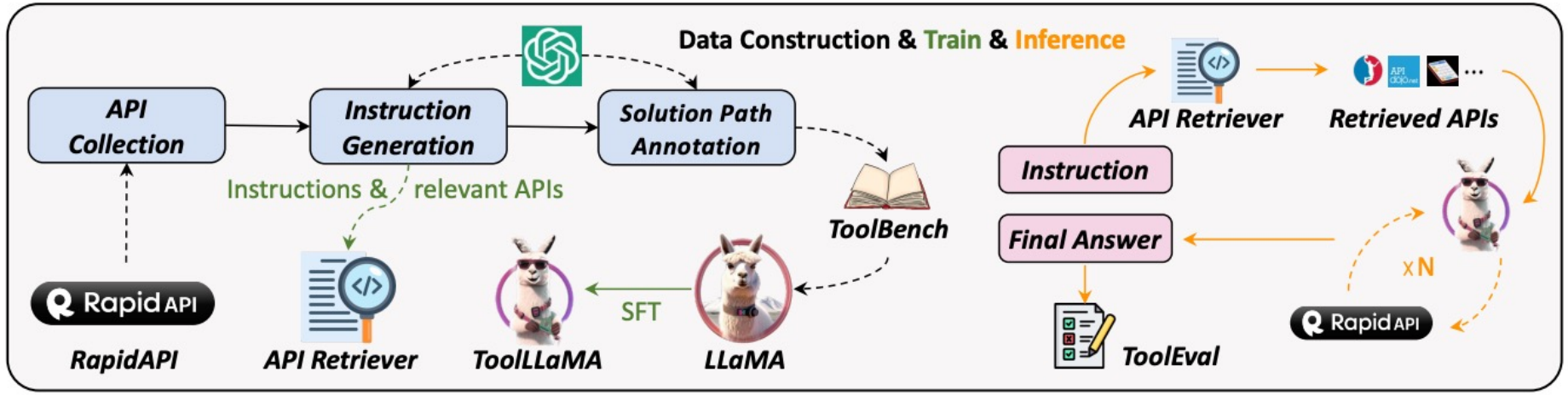


Figure 1: Three phases of constructing ToolBench and how we train our API retriever and ToolLLaMA. During inference of an instruction, the API retriever recommends relevant APIs to ToolLLaMA, which performs multiple rounds of API calls to derive the final answer. The whole reasoning process is evaluated by ToolEval.

„ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs“ [Link]

End2End

Tools for LLMs / LLMs as Tools

1.Tsinghua University ModelBest Inc.
2.Renmin University of China
3.Yale University
4.WeChat AI, Tencent Inc.
5.Zhihu Inc.

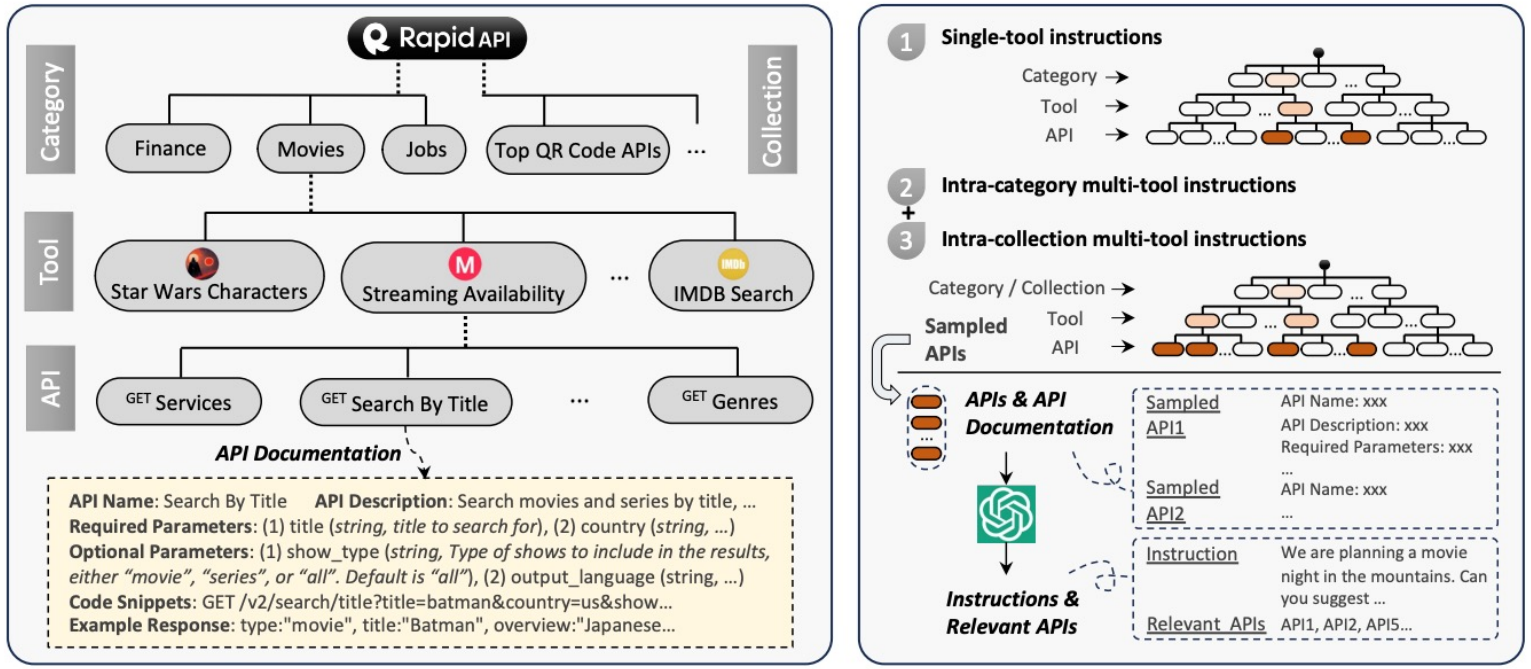


Figure 3: The hierarchy of RapidAPI (left) and the process of instruction generation (right).

„ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs“ [Link]

End2End

Tools for LLMs / LLMs as Tools

1.Tsinghua University ModelBest Inc.
2.Renmin University of China
3.Yale University
4.WeChat AI, Tencent Inc.
5.Zhihu Inc.

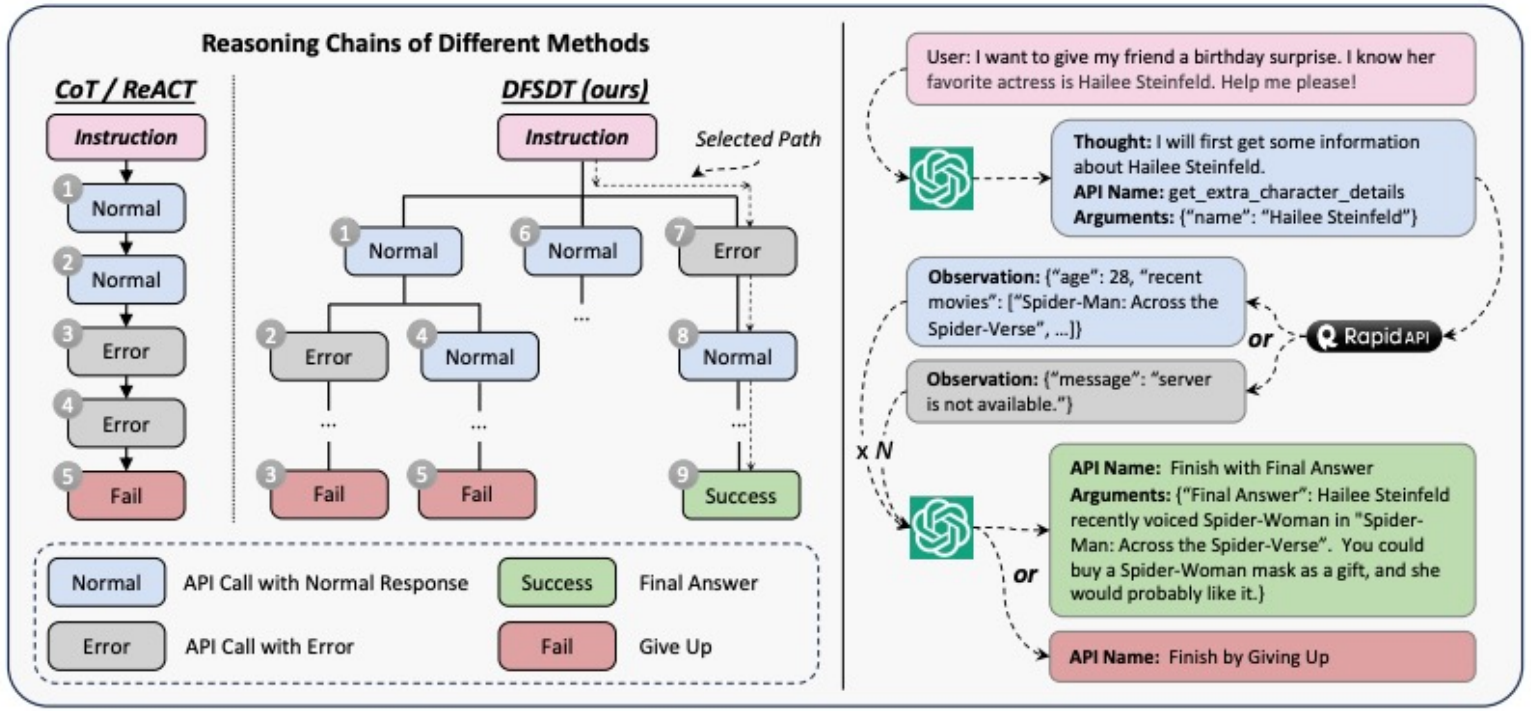


Figure 4: A comparison of our DFSDT and conventional CoT or ReACT during model reasoning (left). We show part of the solution path annotation process using ChatGPT (right).

„ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs“ [Link]

End2End

Tools for LLMs / LLMs as Tools

1.Tsinghua University ModelBest Inc.
2.Renmin University of China
3.Yale University
4.WeChat AI, Tencent Inc.
5.Zhihu Inc.

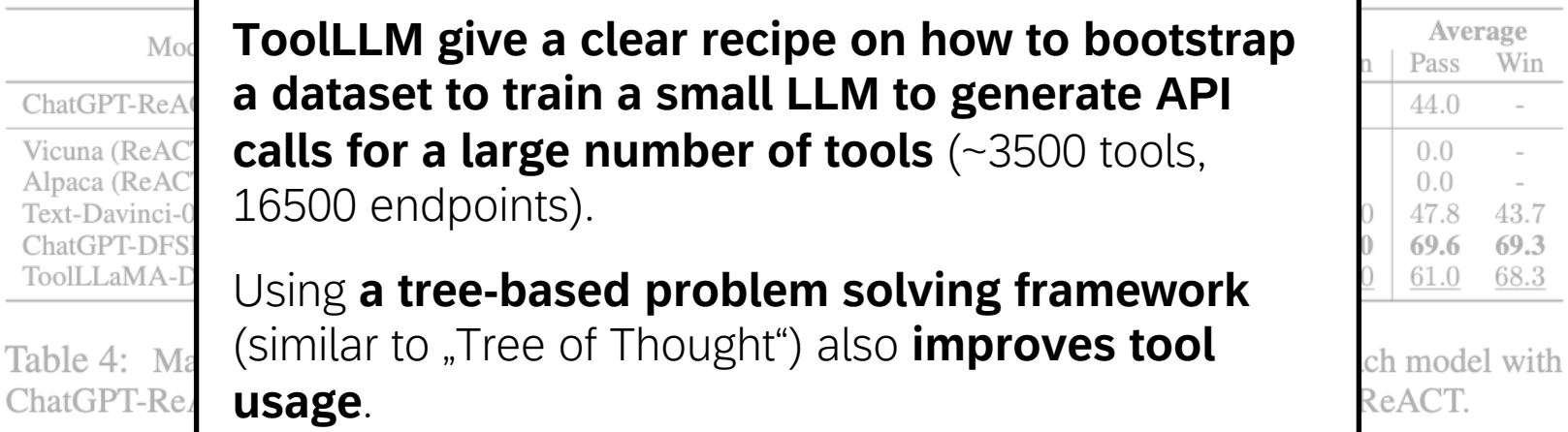
Model	I1-Inst.		I1-Tool		I1-Cat.		I2-Inst.		I2-Cat.		I3-Inst.		Average	
	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win
ChatGPT-ReACT	56.0	-	62.0	-	66.0	-	28.0	-	22.0	-	30.0	-	44.0	-
Vicuna (ReACT & DFSDT)	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-
Alpaca (ReACT & DFSDT)	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-
Text-Davinci-003-DFSDT	53.0	46.0	58.0	38.0	61.0	39.0	38.0	46.0	38.0	45.0	39.0	48.0	47.8	43.7
ChatGPT-DFSDT	**78.0**	**68.0**	**84.0**	**59.0**	**89.0**	**57.0**	**51.0**	**78.0**	**58.0**	77.0	**57.0**	**77.0**	**69.6**	**69.3**
ToolLLaMA-DFSDT	68.0	**68.0**	80.0	**59.0**	75.0	56.0	47.0	75.0	56.0	**80.0**	40.0	72.0	61.0	68.3

Table 4: Main experiments on the test set of ToolBench. Win rate is calculated by comparing each model with ChatGPT-ReACT. A win rate higher than 50% means the model performs better than ChatGPT-ReACT.

„ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs“ [Link]

End2End

Tools for LLMs / LLMs as Tools



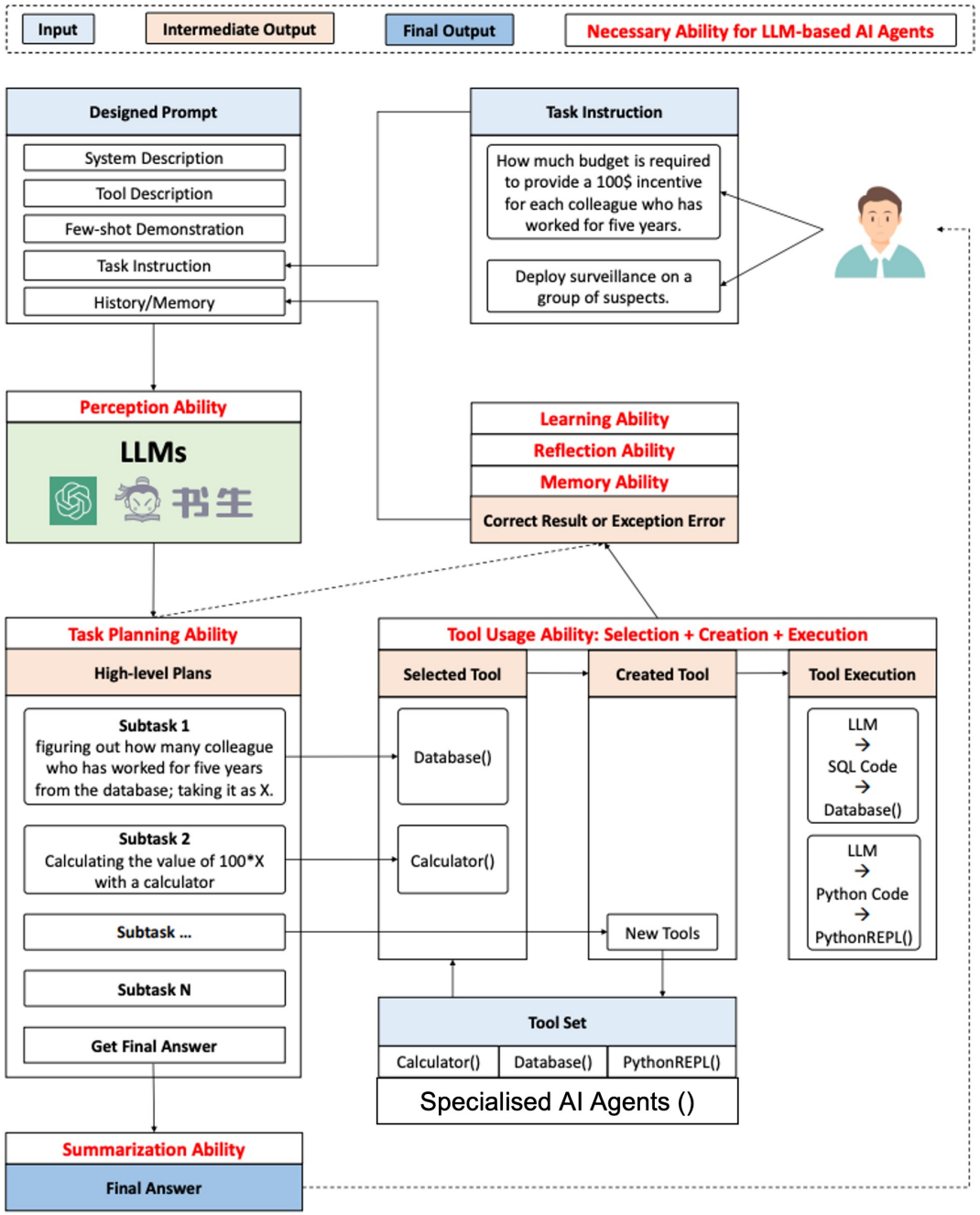
Mo		Average Pass	Win
ChatGPT-ReA		44.0	-
Vicuna (ReAC		0.0	-
Alpaca (ReAC		0.0	-
Text-Davinci-(	0	47.8	43.7
ChatGPT-DFS	0	**69.6**	**69.3**
ToolLLaMA-D	0	61.0	68.3

Table 4: Ma ch model with ChatGPT-Re ReACT.

ToolLLM give a clear recipe on how to bootstrap a dataset to train a small LLM to generate API calls for a large number of tools (~3500 tools, 16500 endpoints).

Using **a tree-based problem solving framework** (similar to „Tree of Thought“) also **improves tool usage**.

Take Aways



Take Aways

Formulating a complete plan before solving and giving the model a **chance to reflect** on it seems to **help the model reason significantly**.

Tools can be reasonably well controlled **by in-context learning (only providing documentation) with LLMs >100B**.

The **usefulness of few-shot examples** seems to be **limited** and sometimes even limits the model's ability to use the tools.

LLMs‘ **reasoning and problem-solving abilities** can even work for embodied agents! [Google‘s pivot in robotics]

Skill library for agents with **live long learning** capabilities.

Giving an LLM the ability to **go back and reconsider** old decisions further improves the ability to **solve complex problems.**

There is **no obvious best agent architecture**.

Open source models can‘t compete with commercial models when used for AI agents.

ToolLLM give a clear recipe on how to bootstrap a dataset to train a small LLM to generate API calls for a large number of tools (~3500 tools, 16500 endpoints).

Using **a tree-based problem solving framework** (similar to „Tree of Thought“) also **improves tool usage**.