

# StreamBench: Towards Benchmarking Continuous Improvement of Language Agents

*NeurIPS 2024 Track on Datasets and Benchmarks*

Cheng-Kuang Wu<sup>1\*</sup>, Zhi Rui Tam<sup>1\*</sup>, Chieh-Yen Lin<sup>1</sup>,  
Yun-Nung Chen<sup>2</sup>, Hung-yi Lee<sup>2</sup>

<sup>1</sup>Appier AI Research

<sup>2</sup>National Taiwan University

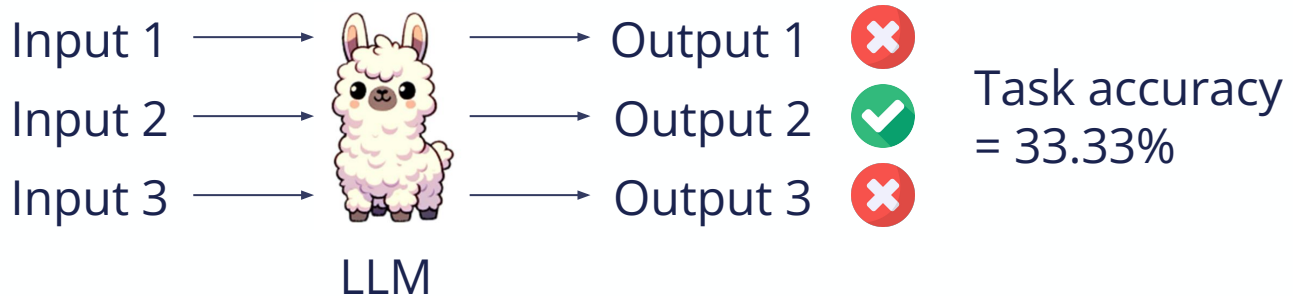
Appier

Paper





# Background

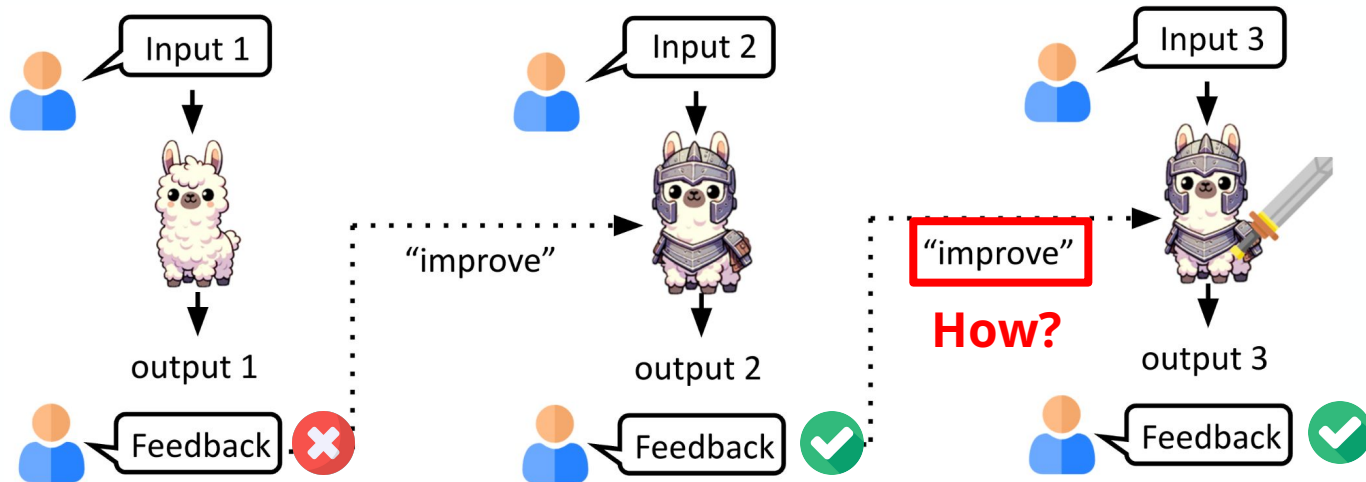
- Most benchmarks measure LLMs' *innate capabilities* (out-of-the-box performance) on a **batch** of task instances



- They do NOT measure LLMs' *ability to improve* over time when exposed to a **sequence** of task instances

# Benchmark Setting: Input-Feedback Sequence

- An online streaming setting, which exposes the LLM to an **input-feedback sequence**
- Input: a natural language instruction / question
- Feedback: correctness  

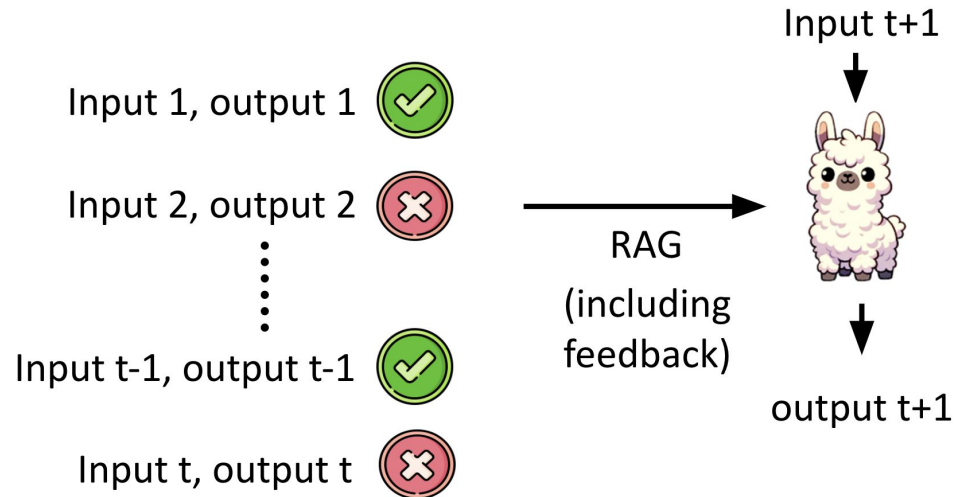


# How to “Improve” the LLMs?

- To enhance LLMs’ capabilities over time, we can design an LLM “agent”: an LLM parameterized by  $\theta$  and augmented with **additional components**
  - Prompts
  - RAG memory
  - RAG retriever
  - Other creative components
- Design **update algorithm** to improve these components

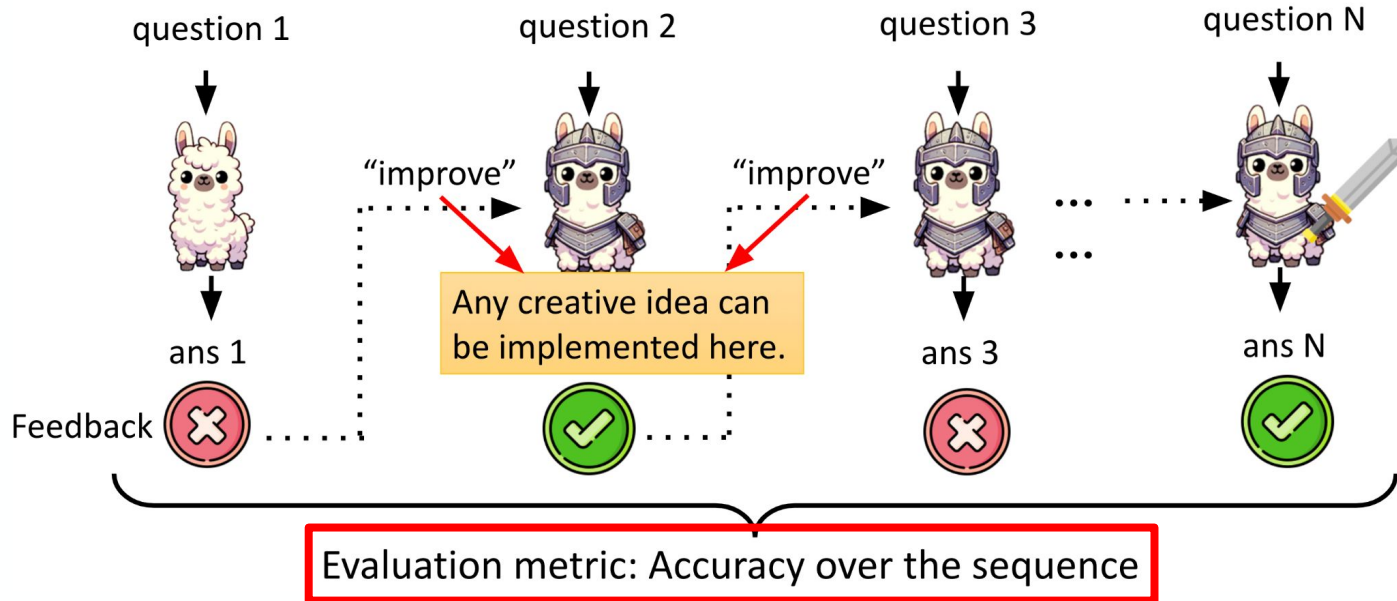
# Example: A RAG-Based Simple Baseline

- Idea: **store past experience** in RAG **memory  $M$** 
  - At each time step  $i$ , store (input, output, feedback) in  $M$
- **Retrieve them in the future** for in-context learning



# Goal: Design Methods to Improve Agents!

- Update parameters  $\theta$ , RAG memory, RAG retriever, prompts, ... or **other creative components** you can come up with!



# Tasks and Datasets

- We choose a variety of tasks and datasets
- For each dataset, **we randomly assign a time step to each data instance** to build the streaming sequence

Table 1: Input, output, evaluation metrics, and number of testing instances of selected datasets.

Task	Text-to-SQL			Python	Tool Use	Medical	QA
Dataset	Spider	CoSQL	BIRD	DS-1000	ToolBench	DDXPlus	HotpotQA
Input ( $x_t$ )	Data requirements			Question	User query	Symptoms	Question
Output ( $y_t$ )	SQL code			Code	API calls	Diagnosis	Answer
Metric	Execution accuracy			Pass@1	Accuracy	Accuracy	Exact Match
Test size ( $T$ )	2,147	1,007	1,534	1,000	750	1,764	1,500

# Experiments

- We try various non-streaming and streaming methods
- General finding: **streaming** > non-streaming

Table 2: Averaged performance of three LLM agents across different baselines and datasets.

Task	Text-to-SQL			Python	Tool Use	Medical	QA
	Spider	CoSQL	BIRD	DS-1000	ToolBench	DDXPlus	HotpotQA
<i>Non-streaming</i>							
Zero-Shot	67.89	50.55	29.60	37.70	61.38	52.85	48.49
Few-Shot	68.55	50.61	30.40	33.33	68.58	60.98	53.11
CoT	61.53	46.01	27.23	25.93	58.98	58.20	52.47
Self-Refine	67.75	49.49	29.62	36.30	60.67	52.89	43.53
<i>Streaming</i>							
GrowPrompt	69.90	51.97	30.35	33.77	65.07	55.10	51.38
MemPrompt	70.78	53.29	31.99	35.47	64.31	54.02	52.62
Self-StreamICL	74.63	55.05	35.31	41.30	71.33	70.56	54.80
MAM-StreamICL	<b>75.69</b>	<b>55.17</b>	<b>36.38</b>	<b>43.10</b>	<b>75.87</b>	<b>83.50</b>	<b>55.20</b>



# What Makes Effective Streaming Strategies?

- We try various non-streaming and streaming methods
- General finding: **streaming** > non-streaming

Table 2: Averaged performance of three LLM agents across different baselines and datasets.

Task	Text-to-SQL			Python	Tool Use	Medical	QA
	Spider	CoSQL	BIRD	DS-1000	ToolBench	DDXPlus	HotpotQA

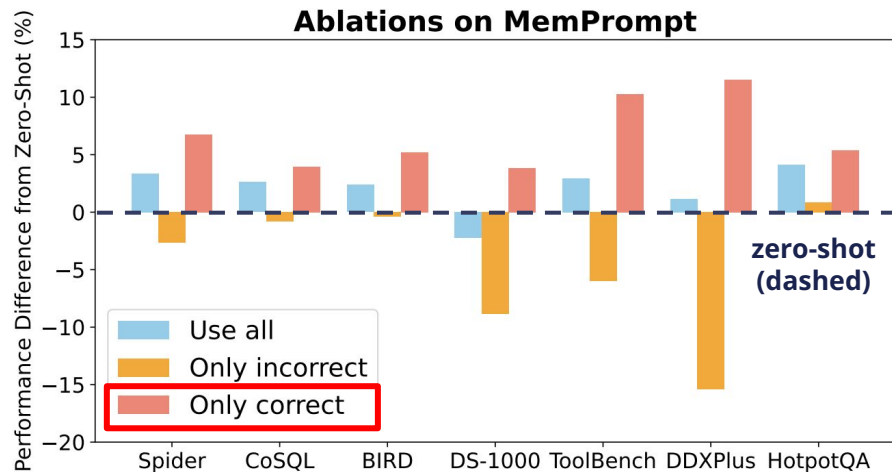
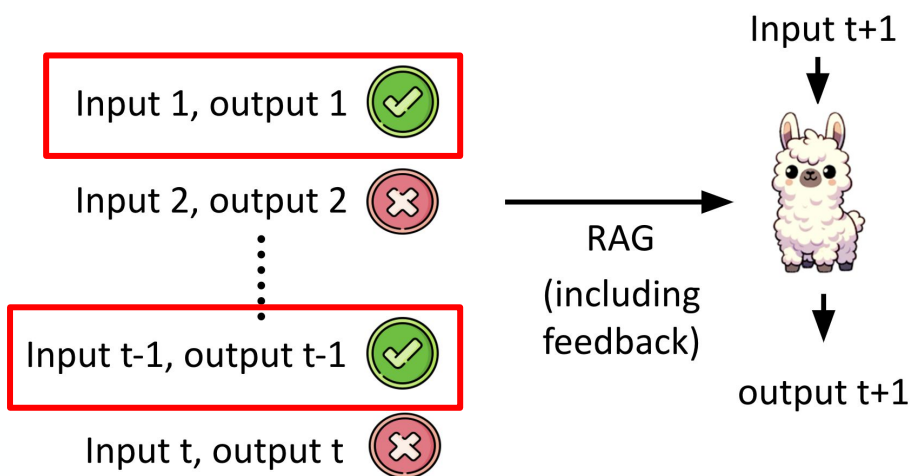
## What makes some streaming methods more effective than others?

### Streaming

GrowPrompt	69.90	51.97	30.35	33.77	65.07	55.10	51.38
MemPrompt	70.78	53.29	31.99	35.47	64.31	54.02	52.62
Self-StreamICL	74.63	55.05	35.31	41.30	71.33	70.56	54.80
MAM-StreamICL	<b>75.69</b>	<b>55.17</b>	<b>36.38</b>	<b>43.10</b>	<b>75.87</b>	<b>83.50</b>	<b>55.20</b>

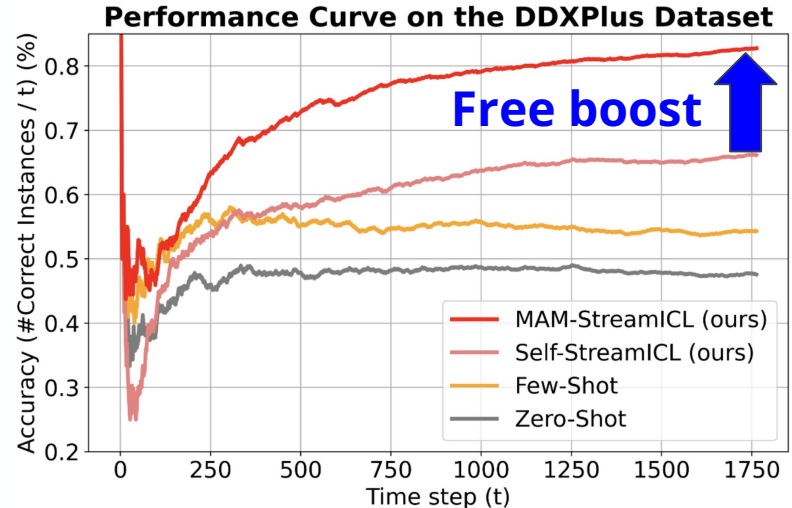
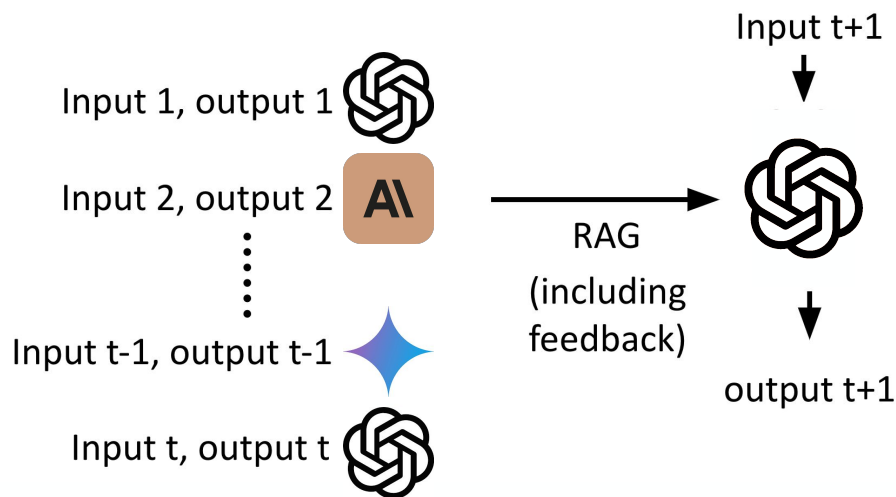
# Only Save Correct LLM Output to RAG Memory

- Saving **only the correct LLM self-generated outputs** to RAG memory is more beneficial for performance boost
- Collecting incorrect self-output even hurts performance



# Sharing Memory Across Multiple LLMs

- Make **different LLMs take turns to solve problems** at each time step, and share the RAG memory together
- **Same** averaged inference **cost** of using a single LLM!



# For Details, Check Out our Paper and Code!

**Paper**



<https://arxiv.org/abs/2406.08747>

**Code**



<https://github.com/stream-bench/stream-bench>