A study on Classification based Concurrent API Calls and Optimal Model Combination for Tool Augmented LLMs for AI Agent

HeounMo Go[1] & SangHyun Park[1]*

[1] Department of Computer Science, Yonsei University, Yonsei-ro 50, Seodaemun-gu, Seoul 03722, Republic of Korea. Yonsei Univ., Republic of Korea

* Corresponding authors: SangHyun Park (sanghyun@yonsei.ac.kr)

Abstract

AI Agents have evolved to not only recommend content but also facilitate information retrieval and task processing. Developing AI Agents using general-purpose LLM models necessitates integration with external tools, leading to tool-augmented LLM studies. Despite the availability of multiple tools for the same purpose, existing research has not fully leveraged this diversity. This study categorizes external tools by type and proposes a method to simultaneously call tools of the same type. This allows for the utilization of diverse external tools in LLM inference, thereby achieving a higher accuracy compared to when only a single tool for one task is used. Experimental results show an accuracy improvement of 4.4% to 9.3% over existing studies. Furthermore, when utilizing tool-augmented LLM, a multi-step reasoning approach that divides the process into stages such as planning and tool invocation is widely employed. With the rapid advancement of LLMs, enhanced models continue to emerge. Considering the trade-offs between performance and cost in models, it is crucial to find an optimal combination of models in each stage of tool augmented LLM. In this study, we propose a novel method for efficiently utilizing both enhanced LLM models and existing models, which reduces response errors by up to 9%.

Introduction

The success of ChatGPT has heightened interest in natural language-based AI Agents. AI Agents have evolved to handle tasks such as personalized content recommendations, information retrieval, action completion including restaurant reservations, and product purchases through familiar conversational interfaces. Consequently, many companies are leveraging AI Agents to enhance their business competitiveness. For instance, SK Telecom offers the AI Agent service, "A.," to enable customers to use artificial intelligence more easily and conveniently in their daily lives. A. provides a "Daily" feature that integrates personal tasks such as to-dos, schedules, and records, allowing users to manage calendars, tasks, routines, and sleep in a unified manner. This service supports users in managing their daily activities effortlessly by allowing them to communicate with A. as if speaking to a personal assistant, thereby storing and managing appointments, meetings, and tasks without the hassle of manual input.

The AI assistant experience is significantly enhanced by offering personalized suggestions that consider various factors such as weather, traffic, and the user's preferences and tastes when performing schedules and appointments. Additionally, A. provides a natural conversational experience based on a large language model and offers a variety of agent-based services, including music, media, stock trading, and movie reservations. Through the multi-LLM agent, users can simultaneously receive and compare responses from seven of the latest LLMs, including ChatGPT, Perplexity, Claude, and A.X. Microsoft has integrated MS Copilot into its Office suite to boost productivity in document tasks, and AI Agents are being actively developed to improve the efficiency of call centers, helping and potentially replacing human agents. AI Agents are also being explored for productivity and cost reduction in fields like marketing and coding.

These AI Agents are built on LLM models, which can be proprietary, or external models like ChatGPT given the high costs of developing and maintaining proprietary LLMs. While general-purpose LLM models like ChatGPT perform well in open-domain Q&A due to extensive training data, integrating with various external tools is essential for real-world applications involving domain-specific knowledge and task execution, which is called tool augmented LLM. For example, an AI Agent must integrate with a weather information API to provide local weather forecasts or with a booking API to handle accommodation reservations.

Typically, there are several kinds of tools even for same purpose. For example, google search API and Bing web search API are available for information search, and there are multiple weather forecast APIs in each country. However, existing research has not fully leveraged this diversity for more accurate responses. This study categorizes external tools by type for real-world AI Agent applications and proposes a method to simultaneously call tools of the same type, inputting their results into the LLM model to enhance inference accuracy. This allows for the utilization of the execution results from a more diverse range of external tools in LLM inference, thereby achieving a higher response accuracy compared to when only a single tool for one task is used. Furthermore, when utilizing tool-augmented LLM, a multi-step reasoning approach that divides the process into stages such as planning and tool invocation is widely employed. With the rapid advancement of LLMs, increasingly enhanced models continue to emerge. However, high-performance models require significant token costs, making it crucial to find an optimal combination of models that considers both performance and cost. In this study, we propose a novel method for efficiently combining and utilizing both enhanced LLM models and existing models.

Related Work

While LLMs demonstrate high performance across various domains, research on tool augmented LLM, which is integrating external tools with LLM, has been active for enhancing reasoning performance and accuracy.

Initial studies focused on enhancing LLM inference capabilities to improve responses to common-sense questions, mathematical queries, and symbolic reasoning [1]. Research has also explored using external tools to enhance natural language processing and image recognition performance, such as Tool Former [2] and Visual GPT [3]. Additionally, studies like ReAct [4], ART [5], MM-ReAct [6], TaskMatrix.AI [7], and ReWOO [8] have investigated multi-step reasoning and external tool integration to improve problem-solving capabilities.

Addressing the hallucination problem, where LLMs provide incorrect answers in domain-specific knowledge tasks, has also been a focus [9][10][11]. Solutions include integrating external tools during action execution [12][13][14][15][16], with particular attention to real-world API utilization [17][18][19][20][21]. Research has explored various APIs to enhance LLM performance, such as search APIs [22][23][24], calculator APIs for error correction [25], code interpreter APIs for code generation quality [26], and math prover APIs for mathematical problem proofs [27].

Typically, there are several kinds of tools even for a same purpose. For example, google search API and Bing web search API are available for information search, and there are multiple weather forecast APIs in each country. Despite the availability of multiple APIs for the same purpose, existing studies have not fully leveraged this diversity for more accurate responses, which means that if we allow for the utilization of the execution results from a more diverse range of external tools in LLM inference, we can achieve a higher response accuracy compared to when only a single tool for one task is used as previous studies do. This study categorizes external APIs by purpose and type for real-world AI Agent applications and proposes a method to simultaneously call APIs of the same type, inputting their results into the LLM model to enhance inference accuracy. This process includes exception handling to minimize unnecessary token costs.

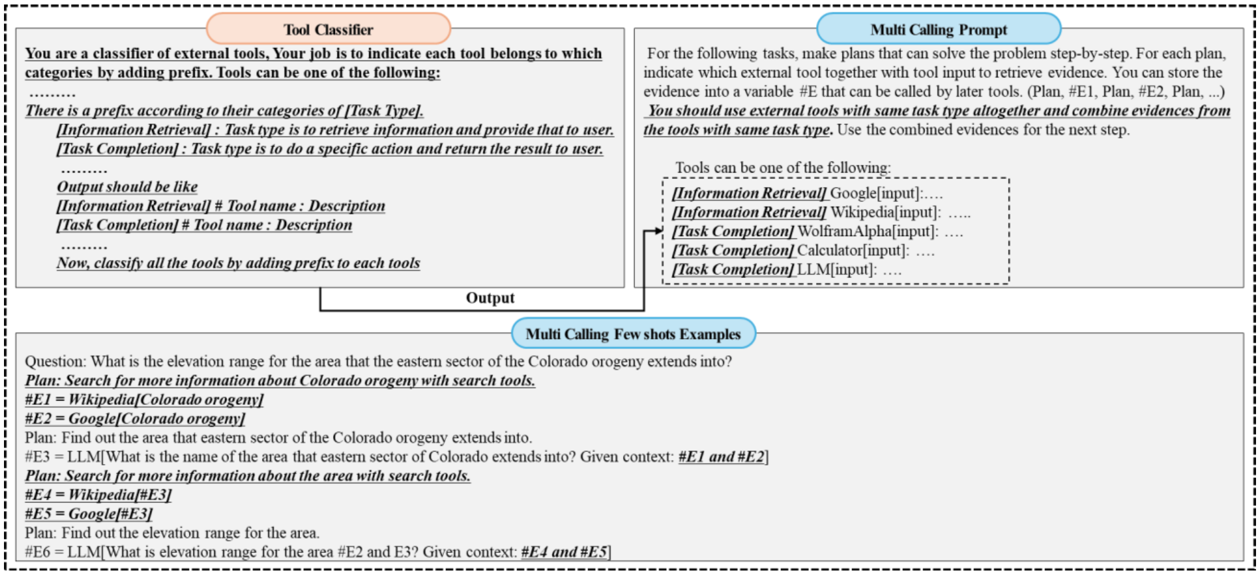
Additionally, as LLM models continue to evolve, there are various versions of models available even in a same type of LLM, differentiated by performance and price. From the perspective of companies developing real world AI agent services, it is necessary to efficiently combine these various models according to the purpose and characteristics of the service. This study proposes a method to efficiently combine new and high-performance models with existing models when using external tools.

Methods

Work Flow

This study uses ReWOO [8] as the base methodology. ReWOO enhances performance by utilizing external tools through multiple inference steps and establishes an overall inference plan in a single LLM call during the Plan stage, avoiding multiple steps like CoT [28] and ReAct [4].

The workflow consists of three main stages: Planner, Worker, and Solver. The Planner stage establishes an overall plan to solve the given problem, as illustrated in Figure 1.


Tool Classifier

You are a classifier of external tools, Your job is to indicate each tool belongs to which categories by adding prefix. Tools can be one of the following:
....
There is a prefix according to their categories of [Task Type].
[Information Retrieval] : Task type is to retrieve information and provide that to user.
[Task Completion] : Task type is to do a specific action and return the result to user.
....
Output should be like
[Information Retrieval] # Tool name : Description
[Task Completion] # Tool name : Description
....
Now, classify all the tools by adding prefix to each tools

Multi Calling Prompt

For the following tasks, make plans that can solve the problem step-by-step. For each plan, indicate which external tool together with tool input to retrieve evidence. You can store the evidence into a variable #E that can be called by later tools. (Plan, #E1, Plan, #E2, Plan, ...) ***You should use external tools with same task type altogether and combine evidences from the tools with same task type.*** Use the combined evidences for the next step.

Tools can be one of the following:
[Information Retrieval] Google[input]:....
[Information Retrieval] Wikipedia[input]: .....
[Task Completion] WolframAlpha[input]: ....
[Task Completion] Calculator[input]: ....
[Task Completion] LLM[input]: ....

Multi Calling Few shots Examples

Question: What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
Plan: Search for more information about Colorado orogeny with search tools.
#E1 = Wikipedia[Colorado orogeny]
#E2 = Google[Colorado orogeny]
Plan: Find out the area that eastern sector of the Colorado orogeny extends into.
#E3 = LLM[What is the name of the area that eastern sector of Colorado extends into? Given context: ***#E1 and #E2***]
Plan: Search for more information about the area with search tools.
#E4 = Wikipedia[#E3]
#E5 = Google[#E3]
Plan: Find out the elevation range for the area.
#E6 = LLM[What is elevation range for the area #E2 and E3? Given context: ***#E4 and #E5***]

Figure 1. Planner Workflow – Tool Classifier designed to categorize external tools based on their specific purposes, Multi Calling Prompt and Few shot examples that allows for the simultaneous invocation of these tools

The first step in the Planner stage is to classify the given external tools by purpose using a Tool Classifier, which is done through a prompt to the LLM. The Tool Classifier serves the function of categorizing external tools into different types based on their purpose-specific similarities, by adding a prefix to the tool description. To achieve this, it provides the Large Language Model (LLM) with a prompt that includes candidate types, their descriptions, and examples of the output format with the prefix.

Figure 1 shows an example. Here, there are two categories: Information Retrieval and Task Completion. The reason why we choose these two categories as an example is described in the following section.

Multi Calling Prompt calls tools of the same classification simultaneously with the categorization prefix from Tool Classifier. Few-shot examples also call tools of the same classification simultaneously, passing their results to the next stage. Figure 1 shows an example where Wiki and Google tools, classified for Information Retrieval, are called simultaneously, passing both results to the next stage. Calling multiple tools with the same purpose simultaneously increases response accuracy by utilizing results from other tools if one tool fails to provide the desired outcome. The Tool Classifier and Multi Calling Prompt for simultaneous tool calls are key contributions of this study, highlighted in bold and underlined in Figure 1. In the next stage, the Worker calls the actual external tools according to the plan established by the Planner, retrieving evidence for each step. If the external tool's result is deemed unsatisfactory, exception handling is used to prevent unnecessary tokens from being input into the LLM, which is a contribution of this study. For example, if a search tool's result includes "not find," it is replaced with null to avoid unnecessary token costs.

Finally, the Solver inputs the collected evidence into the LLM according to the plan to derive the answer to the problem.

Classification Criteria

For real-world applications, optimizing the classification of external tools by purpose is necessary. A broader classification may result in calling too many external tools simultaneously, potentially exceeding the LLM's context window and increasing token costs. Conversely, a narrower classification may diminish the effect of calling multiple tools simultaneously.

This study verifies the effectiveness of the proposed method by considering two types: "Information Retrieval" for widely used search tools like Google and Wiki, and "Task Completion" for specific tasks like arithmetic operations. The two classification types presented in this study as examples are the most commonly used in AI Agent service use cases. For the most representative AI Agent, ChatGPT, the most typical use case is providing information retrieval results in a question-and-answer format. Recently, there has been an effort to expand its capabilities towards task completion through features such as function calls. Although this study only applies classification by purpose, domain-specific classifications could be added for real-world applications. For example, domains like "Contents" for music and videos, or "Accommodation" and "Airline" for travel reservations, and "Restaurant" for dining recommendations. If domain classifications are added, the Tool Classifier's input prompt and examples can be modified to include two-step prefixes, such as [Information Retrieval] [Contents], and included in the Multi Calling Prompt. This will be addressed in future work.

LLM Combination Optimization

With the rapid advancement of LLM models, more evolved models continue to emerge. For instance, ChatGPT has released versions like 3.5 turbo, 4o, 4o-mini, and o1. While newer models generally offer better performance, they also tend to have higher token costs. Although the latest models would be applied for performance, it would require more token cost.

In addition, considering the high degree of freedom in the responses of Large Language Models, the adoption of new models could significantly impact the service, which means that companies should consider additional verification and operational costs when switching from optimized services based on existing models. Considering these aspects, it is necessary to efficiently combine the existing model with the new model rather than completely replacing it with a new LLM all at once.

This study investigates the most efficient combination of LLM models for the Planner and Solver stages, using the widely used ChatGPT 3.5 turbo model and the advanced GPT 4o model. The reason for selecting the ChatGPT 3.5 turbo model and the GPT 4o model for this experiment is that the ChatGPT 3.5 turbo is one of the oldest and most affordable models provided by OpenAI, whereas the GPT 4o is one of the most expensive and latest models. This study aims to analyze the performance variation trends between a combination of a low-cost, older model and a high-cost, latest model, which is why these two models were chosen.

Experiment

Datasets

The datasets used in this study are listed in Table 1.

Task	Dataset	Description
Common Knowledge and Reasoning	HotpotQA [29]	Multi-hop reasoning QA task over diversified domains
	StrategyQA[30]	Open domain QA task where answers require steps of reasoning.
	Sports Understanding[31]	Factual QA benchmark from BigBench[32] over sports domain
Scientific Reasoning	PhysicsQuestion[33]	High school physics questions.
Curated	SOTUQA [8]	QA dataset over State of the Union Address 2023

Table 1. Tasks and Datasets

In selecting the dataset used for this study, the following factors were considered. First, we chose publicly available data widely used in the commonsense and reasoning domains of open domain applications where Large Language Models (LLMs) are extensively utilized (Hotpot QA and Strategy QA). Specifically, we aimed to select datasets that could verify multi-hop reasoning abilities. (Hotpot QA) To test response capabilities in specific domains outside the open domain, we included the Sports Understanding data. Additionally, to verify response capabilities for different types of tasks, such as arithmetic calculations rather than general reasoning or question-answering, we incorporated the Physics Question data. Lastly, to assess the performance of the proposed method on commonly used curated data, we aimed to evaluate its performance using the SOTUQA data

Baselines

To verify the effectiveness of the proposed approach, comparisons is made with the Direct method, which inputs questions directly into the LLM without step-by-step prompts, and ReWOO [8], one of the latest multi-step reasoning methods using external tools.

Models

The gpt-3.5-turbo-1106 model, widely used in LLM-based services, is primarily used, and the gpt-4o-2024-05-13 model is used to find the optimal combination

External Tools

The external tools used in this study are listed in Table 2.

Type (prefix)	Tool	Description
Information Retrieval	Wikipedia	Search engine for Wikipedia
	Google	Search result from Google SERP API
Task Completion	WolframAlpha	Computation result from Wolfram Alpha API
	Calculator	Program-aided LLM
	LLM	Separate single LLM

Table 2. External Tools

Metrics

Exact match (EM) and character-level F1 scores are used for performance comparisons, which are widely used in natural language processing. Additionally, accuracy is calculated through GPT-based answer comparisons to measure semantic accuracy, which is the main metric for evaluation. Efficiency is compared by calculating the number of tokens and costs used by the planner, worker, solver, and external tools, as well as the number of inference steps. For inference steps, the direct method is counted as 1, and the proposed method and ReWOO are counted as the number of steps proposed by the Planner +1, considering the Solver stage as a single step.

Results

Comparison with baselines

Table 3 shows that the proposed method generally achieves higher accuracy than the comparison methods, as illustrated in Figure 2.

Dataset	Method	Acc	F1	EM	#Tokens	#Steps	$Cost(1k)
HotpotQA	Direct	59.0	42.0	34.0	47.6	1.00	0.05
	ReWOO	55.0	38.3	29.0	1749.7	3.55	1.75
	Our Approach	**64.5**	34.9	23.7	3011.6	3.43	3.01
StrategyQA	Direct	64.0	63.0	63.0	40.5	1.00	0.04
	ReWOO	70.7	70.7	70.7	1033.1	3.06	1.03
	Our Approach	**76.7**	76.7	76.7	1263.7	3.03	1.26
SportsU	Direct	71.0	67.0	67.0	44.3	1.00	0.04
	ReWOO	66.7	60.4	60.4	1840.8	3.83	1.84
	Our Approach	**77.1**	63.8	62.9	1491.2	3.23	1.49
PhysicsQA	Direct	68.0	17.9	-	105.9	1.00	0.11
	ReWOO	66.7	18.2	-	1161.0	2.51	1.16
	Our Approach	**71.0**	15.9	-	1485.7	3.13	1.49
SOTUQA	Direct	**88.0**	33.5	-	64.7	1.00	0.06
	ReWOO	77.1	25.4	-	2119.6	3.44	2.12
	Our Approach	80.4	29.7	-	2755.8	3.00	2.76

Table 3. Comparison with baselines

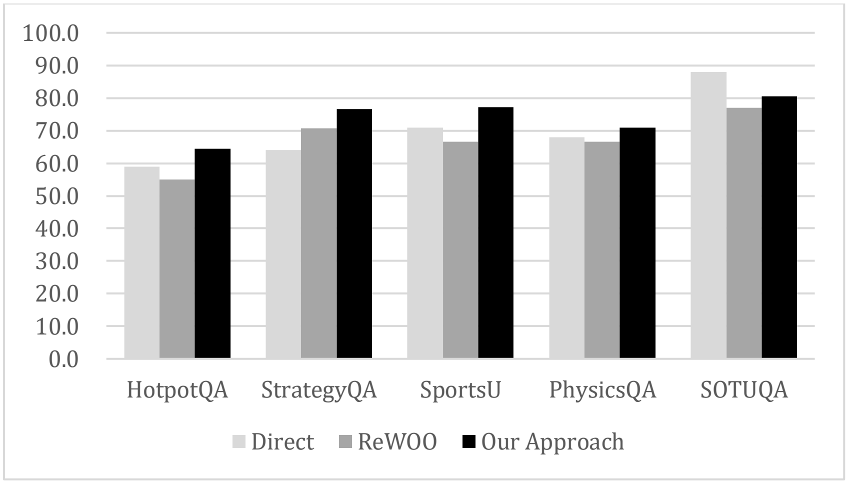


Figure 2. Comparison with baselines

Compared to the highest-performing baseline, the proposed method shows an accuracy improvement of +9.3% for HotpotQA, +8.5% for StrategyQA, +8.7% for Sports Understanding, and +4.4% for PhysicsQA. Character-level F1 and Exact Match show inconsistent trends, indicating that word-level comparisons alone are insufficient for measuring accuracy in complex questions. Although token count and cost are higher than ReWOO due to simultaneous tool calls, the variation is dataset-dependent, with Sports Understanding showing a lower token count. For complex problems like HotpotQA, simultaneous tool calls result in more tokens, leading to higher token increases. However, for simpler datasets like StrategyQA and PhysicsQA, token increases are relatively lower, 22% and 28% respectively. Interestingly, Sports Understanding shows a 19% decrease in token count, likely due to filtering external tool results such as nullifying "not find" results, minimizing unnecessary token costs. This suggests that appropriate filtering can minimize token cost increases even with simultaneous tool calls.

The number of steps is similar to ReWOO. Overall, the proposed method shows accuracy improvements and potential for minimizing token cost increases with proper filtering. In the case of the number of steps, as expected, the direct approach shows a count of 1, while both ReWOO and our approach exhibit a similar count of approximately 3. The reason for the number of steps not increasing in our approach is presumed to be due to the handling of simultaneous calls to the same type of API in a single step, rather than treating them as separate steps.

For SOTUQA, the direct LLM call without external tools shows the highest accuracy, likely due to the LLM model's continuous updates incorporating relevant training data.

In summary, the experiment result demonstrates an improvement in accuracy ranging from 4.4% to 9.3% compared to existing studies, based on experimental results. Although the token cost increased by 22% (Strategy QA) to 72% (Hotpot QA) with additional API usage comparing with baseline methods, the token cost in domain specific dataset (Sports Understanding) shows a reduction of up to 19% through filtering. This suggests that more precise filtering of results from multiple API calls, or even the use of a dedicated Small Language Model (SLM) for this purpose, could further reduce token costs. The proposed methodology not only shows meaningful performance improvement but also indicates the potential for token cost optimization which will be addressed in future research.

Model Combination Optimization

The accuracy and efficiency of different ChatGPT model combinations for the Planner and Solver stages for each dataset are compared in Table 4.

Dataset	Model		Acc	F1	EM	#Tokens	#Steps	$Cost(1k)
	Planner	Solver						
HotpotQA	GPT 3.5t	GPT 3.5t	64.5	34.9	23.7	3011.6	3.43	3.01
	GPT 3.5t	GPT 4o	**78.5**	67.2	57.0	2979.5	3.43	9.19
	GPT 4o	GPT 3.5t	57.0	37.4	28.0	1773.3	1.20	8.01
	GPT 4o	GPT 4o	72.0	59.9	51.0	1844.3	1.45	9.22
StrategyQA	GPT 3.5t	GPT 3.5t	71.3	70.1	70.1	1268.1	3.02	1.27
	GPT 3.5t	GPT 4o	**83.8**	83.8	83.8	1270.9	3.01	3.58
	GPT 4o	GPT 3.5t	81.3	81.3	81.3	1283.8	3.39	4.14
	GPT 4o	GPT 4o	**84.0**	84.0	84.0	1316.9	3.28	6.58
SportsU	GPT 3.5t	GPT 3.5t	77.1	63.8	62.9	1491.2	3.23	1.49
	GPT 3.5t	GPT 4o	**86.5**	86.5	86.5	1769.1	3.43	5.89
	GPT 4o	GPT 3.5t	73.9	69.3	68.5	1382.8	2.11	4.80
	GPT 4o	GPT 4o	79.6	79.6	79.6	1156.2	1.73	5.78
PhysicsQA	GPT 3.5t	GPT 3.5t	71.0	15.9	-	1485.7	3.13	1.49
	GPT 3.5t	GPT 4o	**82.8**	14.4	-	1560.1	3.07	4.23
	GPT 4o	GPT 3.5t	68.1	16.2	-	1387.9	1.57	5.54
	GPT 4o	GPT 4o	81.6	17.8	-	1369.0	1.43	6.84
SOTUQA	GPT 3.5t	GPT 3.5t	80.4	29.7	-	2755.8	3.00	2.76
	GPT 3.5t	GPT 4o	91.5	31.4	-	2812.8	3.04	8.42
	GPT 4o	GPT 3.5t	88.0	34.4	-	1714.9	1.00	7.98
	GPT 4o	GPT 4o	**94.0**	36.9	-	1770.6	1.16	8.85

Table 4. Model Combination Optimization - Comparison between Models of Planner and Solver

GPT 3.5t means gpt-3.5-turbo-1106, and GPT 4o means gpt-4o-2024-05-13 model which is more advanced model. Figure 3 illustrates the accuracy results, with each case representing the Planner/Solver model combination.

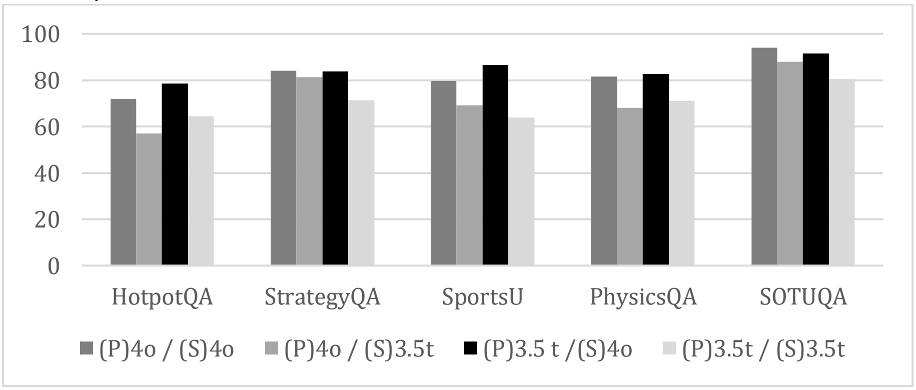


Figure 3. Planner and Solver Model Comparison

Contrary to expectations, using GPT 3.5t for the Planner and GPT 4o for the Solver generally yields the highest accuracy. Compared to the highest-performing baseline, this combination shows an accuracy improvement of +9.0% for HotpotQA, +3.0% for StrategyQA, +8.7% for Sports Understanding, and +1.4% for PhysicsQA. F1 and EM are inconsistent, indicating that word-level comparisons are insufficient for measuring accuracy. Given the significant token cost difference between GPT 3.5t and GPT 4o, token costs rather than token counts should be considered for measuring cost efficiency.

For HotpotQA and Sports Understanding, the total token cost of GPT 3.5t/GPT 4o is similar to the cost of GPT 4o/GPT 4o. Meanwhile, for StrategyQA and PhysicsQA, GPT 3.5t/GPT 4o shows 46% and 38% token cost savings compared to using GPT 4o/GPT 4o respectively. Overall, compared to using GPT 4o for both stages, the proposed combination of GPT 3.5t/GPT 4o shows similar costs with 6.5% and 6.9% higher accuracy for HotpotQA and Sports Understanding respectively, and similar accuracy with 46% and 38% token cost savings for StrategyQA and PhysicsQA. For SOTUQA, the Planner using GPT 3.5t and Solver using GPT 4o shows similar accuracy and costs to using GPT 4o for both stages, likely due to the LLM model's continuous updates incorporating relevant training data, making the Solver model's performance more critical.

To analyze why the Planner using GPT 3.5t and Solver using GPT 4o yields the highest performance, examples of plans generated by each model are included in the Appendix. The GPT 4o plan, unlike the few-shot example, applies the Wiki and Google tools for different purposes, likely due to the model's advanced reasoning capabilities, and it results in incorrect answers. The GPT 3.5t plan follows the few-shot example format, yielding correct answers. This suggests that higher LLM model sophistication does not always guarantee better performance, and in cases requiring more controllability than reasoning performance, existing models may be more suitable.

When using GPT-4o as the Planner, the number of steps is approximately 1 to 2, whereas using GPT-3.5t results in about 3 steps. This indicates that the number of steps is significantly smaller when GPT-4o is employed as the Planner. As previously mentioned, GPT-4o tends to generate plans that differ from the few-shot examples due to its enhanced inference capabilities, which is presumed to result in fewer steps.

Conclusion

This study proposes a method to enhance the accuracy of AI Agent inference results by categorizing external tools by type and simultaneously calling tools of the same type, which allows for the utilization of the execution results from a more diverse range of external tools in LLM inference, thereby achieving a higher response accuracy compared to when only a single tool for one task is used. Experimental results show an accuracy improvement of 4.4% to 9.3% over existing studies. Furthermore, when utilizing tool-augmented LLM with multi-step reasoning approach, it is crucial to find an optimal combination of models in each step. We propose a new method combining enhanced LLM models with existing models efficiently reduces response errors by up to 9%. Future work will explore performance improvements through diversified classification criteria and domain-specific standards for real-world applications.

Data and Code availability

Data and code for this study are available at https://github.com/ilyhs79/Concurrent-API-Calls-and-Optimal-Model-Combination-for-Tool-Augmented-LLMs

References

1. Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. arXiv preprint arXiv:2302.07842, 2023.
2. Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. arXiv preprint arXiv:2302.04761, 2023.
3. Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671, 2023
4. 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, 2023.
5. Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. arXiv preprint arXiv:2303.09014, 2023.
6. Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. arXiv preprint arXiv:2303.11381, 2023.
7. Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. arXiv preprint arXiv:2303.16434, 2023.
8. Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, Dongkuan Xu. ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models arXiv preprint arXiv:2305.18323
9. Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. arXiv preprint arXiv:2305.11738, 2023
10. Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Xin Zhao, and Ji-Rong Wen. ChatCoT: Toolaugmented chain-of-thought reasoning on chat-based large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 14777–14790, Singapore, December 2023e. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.985. URL https://aclanthology.org/2023.findings-emnlp.985
11. Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. arXiv preprint arXiv:2309.10691, 2023f
12. Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT:Solving AI tasks with chatGPT and its friends in hugging face. In Thirty-seventh Conference on Neural Information Processing Systems, 2023b. URL https://openreview.net/forum?id=yHdTscY6Ci

13. Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In Thirtyseventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=HtqnVSCj3q
14. Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. arXiv preprint arXiv:2302.04761, 2023
15. Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. arXiv preprint arXiv:2305.18752, 2023b
16. Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. Easytool: Enhancing llm-based agents with concise tool instruction. arXiv preprint arXiv:2401.06201, 2024a
17. 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
18. Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 3102–3116, Singapore, December 2023g. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.187
19. 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
20. Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models, 2023b
21. Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. arXiv preprint arXiv:2311.18760, 2023c
22. 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, 2023.
23. 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.
24. Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering. arXiv preprint arXiv:2203.05115, 2022.
25. Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.
26. Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. arXiv preprint arXiv:2211.10435, 2022.
27. Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. arXiv preprint arXiv:2210.12283, 2022.
28. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903

29. 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. arXiv preprint arXiv:1809.09600, 2018.
30. Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. Transactions of the Association for Computational Linguistics, 9:346–361, 2021.
31. Ethan Kim. Sports understanding. https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/sports_understanding [Online; accessed 13-May-2023].
32. Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD international conference on Management of data, pages 1197–1208, 2013.
33. Ian D Beatty, William J Gerace, William J Leonard, and Robert J Dufresne. Designing effective questions for classroom response system teaching. American journal of physics, 74(1):31–39, 2006Schijen, M., Lakens, D. & Scheel, A. Positive Results in Standard vs Registered Reports in Psychology. OSF https://osf.io/dbhgr/ (2020).

Acknowledgements

This research was supported by the National Research Foundation (NRF) funded by the Korean government (MSIT) (No. RS-2023-00229822).

Author contributions

HM.GO wrote the main manuscript text. All authors reviewed the manuscript

Competing interests

The authors declare no competing interests.