Beyond Text: Unveiling Multimodal Proficiency of Large Language Models with MultiAPI Benchmark

Xiao Liu **Jianfeng Lin** **Jiawei Zhang**
IFM Lab, University of California, Davis
xiao@ifmlab.org, jfglin@ucdavis.edu, jiawei@ifmlab.org

Abstract

The proliferation of Large Language Models like ChatGPT has significantly advanced language understanding and generation, impacting a broad spectrum of applications. However, these models predominantly excel in text-based tasks, overlooking the complexity of real-world multimodal information. This study introduces **MultiAPI**, a pioneering comprehensive large-scale API benchmark dataset aimed at expanding LLMs' proficiency in multimodal contexts. Developed collaboratively through ChatGPT, **MultiAPI** consists of 187 diverse API calls and 1,799 contextual prompts, offering a unique platform evaluation of tool-augmented LLMs handling multimodal tasks. Through comprehensive experiments, our findings reveal that while LLMs demonstrate proficiency in API call decision-making, they face challenges in domain identification, function selection, and argument generation. What's more, we surprisingly notice that auxiliary context can actually impair the performance. An in-depth error analysis paves the way for a new paradigm to address these challenges, suggesting a potential direction for future LLM research.

1 Introduction

Large Language Models (LLMs), such as ChatGPT, have emerged as powerful tools in understanding and generating human language (Li et al., 2023c; Touvron et al., 2023; OpenAI, 2023), playing a pivotal role in diverse open-domain tasks and leaving a significant impact on both industry and academia (Bubeck et al., 2023; Yao et al., 2023; Touvron et al., 2023; Laskar et al., 2023). However, their performance is often confined to the text-based domains and tasks they were trained on, overlooking the multimodal and dynamic nature of real-world information. As people increasingly rely on LLMs to address their daily challenges, the demand for enhancing the task-handling capabilities of these models grows ever more pressing. In addition to addressing many of people's emerging needs in the real world, enhancing LLMs with multimodal problem-solving skills could be a significant step towards the realization of AGI in an idealized future (Bubeck et al., 2023).

Reflecting this demand and vision, recent studies have embarked on two primary approaches to integrate multimodal processing capabilities into existing LLMs (Li et al., 2023a): 1) Joint training or finetuning LLMs with components for multimodal encoding and generation (Wu et al., 2023; Maaz et al., 2023; Zhang et al., 2023a); 2) Introducing auxiliary API tools via natural language interfaces (Patil et al., 2023; Shen et al., 2023; Qin et al., 2023), positioning LLMs as the central decision-making entity determining the appropriate tools to employ for the inquiry. Joint training of multimodal LLMs, despite creating more unified models, faces challenges with computational demands and potential loss of the generalization ability (Bubeck et al., 2023). On the other hand, evolving API functions, which are modularly designed, allow LLMs to adapt to new tasks by simply altering the API configuration.

Despite the significant potential and flexibility the tool-augmented LLMs express on multimodal tasks, their quantitative performance of multimodal tasks when integrated with API tools still remains insufficiently examined. Recent studies are very inadequate and merely focus on and gleaning insights from open-domain tasks such as mathematical computations, database searches, and graph reasoning (Li et al., 2023b; Zhuang et al., 2023; Qiu et al., 2023). This gap in leveraging API tools to achieve multimodal tasks can be attributed to two primary obstacles: 1) the unavailability of high-quality API-prompt datasets, and 2) the absence of established metrics specifically designed to evaluate the efficacy of LLMs in multimodal tasks.

In this paper, we address the aforementioned challenges by constructing a large-scale API

Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024), pages 32–44

instruction-function dataset and evaluates LLMs' multimodal performance, called **MultiAPI**. Based on the HuggingFace dataset (Patil et al., 2023), we extracted models with high-quality descriptions across 9 domains along with their instructions. These models were initially encapsulated as API functions using ChatGPT prompts, followed by meticulous human refinements to ensure executability and consistent arguments across domains. This help create the **MultiAPI** benchmark dataset with 187 functional API calls and 1,799 instructions.

We subsequently conducted experiments on both API-based LLMs and open-sourced LLMs, exploring strategies that were previously proven effective in improving LLM prompting such as in-context learning (Brown et al., 2020) and chain-of-thought (Wei et al., 2023). Our investigation spanned single-step API call (only 1 API is required to resolve the instruction) and sequential API chain (multiple APIs are required) settings, evaluating 4 intuitive aspects: 1) invocation assessment; 2) domain match; 3) function match; and 4) argument match. Results revealed that while models accurately make decisions to invoke API functions, they often suffer from selecting the right function and parameters from the correct domain. Furthermore, we surprisingly noticed that adding auxiliary context could harm the API call performance. Extensive error analyses were conducted to understand the potential cause of such errors, leading us to propose two simple yet effective solutions to mitigate these errors. The experimental results validate the effectiveness of our method.

We summarize the contributions of this paper as follows:

- We constructed a pioneering large-scale multimodal instruction-function benchmark dataset, **MultiAPI**, with 187 executable API functions and 1,799 prompts. This data underwent rigorous human refinement to ensure its robustness and relevance in the context of LLM evaluations.
- Our experimental framework comprehensively assesses both API-based and open-sourced LLMs, revealing their strengths in API call decisions but highlighting challenges in domain and function selection, as well as argument generation.
- A thorough error analysis leads us to mitigate these errors and set a new direction for future LLM research within the multimodal context.

2 Related Work

2.1 Evaluation of Large Language Models

Performance evaluation of LLMs has become a particularly prominent field postdate of the introduction of ChatGPT, providing valuable insights for enhancing future model iterations and assisting the industry in developing more resilient applications. Extensive research has been undertaken to assess the competencies of LLMs (Yin et al., 2023a; Laskar et al., 2023; Zhang et al., 2023d). These works demonstrated LLMs expressed near-human performance on open-domain tasks such as mathematics, coding, law, and psychology. However, their proficiency with tool use has not been thoroughly explored.

Li et al. (2023b) introduced a benchmark for assessing LLMs' tool-use proficiency through a set of APIs. However, the amount of APIs of this dataset is constrained by its reliance on human implementation and primarily evaluates LLMs on general tasks like setting alarms or scheduling meetings.

In contrast, our study pivots to evaluate LLMs' ability to handle multimodal tasks via the use of tool APIs. We have harnessed ChatGPT's code generation capabilities based on the provided code template, followed by meticulous human refinement, to construct **MultiAPI**, a high-quality and large-scale multimodal API dataset. This novel dataset enables us to dive into the multimodal task performance of LLMs, marking a significant advancement in the field.

2.2 Large Language Model Augmentation

Although large language models recently demonstrated superior zero-shot language understanding (OpenAI, 2023; Touvron et al., 2023; Zhang et al., 2023b) capability, the task scope they could handle is highly tethered with their pretraining data. To adapt LLMs to diverse inputs and tasks, recent studies have primarily followed two avenues. The first involves joint fine-tuning of LLMs with pertinent neural network components. In this approach, the hidden representations of novel modalities are aligned with the LLM's latent space (Awais et al., 2023; Wu et al., 2023; Patil et al., 2023; Lyu et al., 2023). The second avenue integrates tools such as API functions as external modules (Schick et al., 2023; Zhang, 2023; Song et al., 2023). The strategy offers enhanced flexibility, allowing API functions

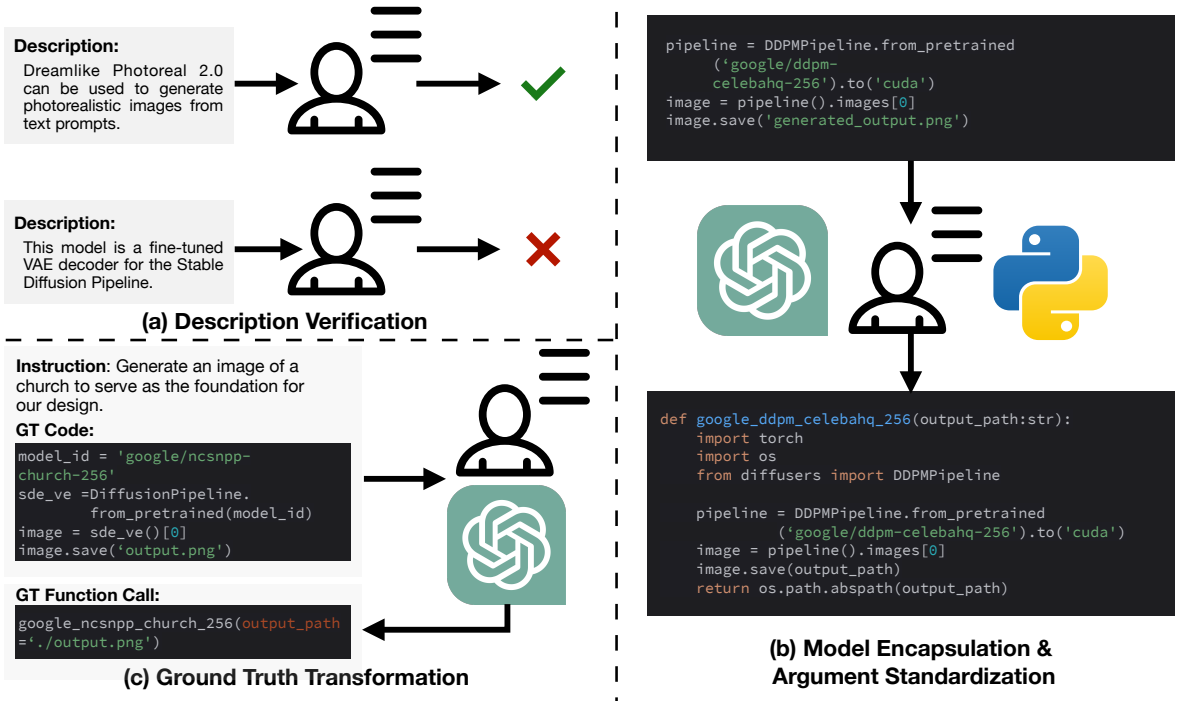

Figure 1: Workflow for adapting the HuggingFace dataset for MultiAPI collaboration with GPT model: (a) the Description Verification process where model descriptions are assessed for precision and detail. (b) the Model Encapsulation and Argument Standardization procedure, transitioning from an 'example code' format to an argument-standardized Python function and ensuring the function is executable. (c) the Ground Truth Transformation, showing the conversion of instruction-code pairs into instruction-function pairs.

to be seamlessly incorporated into textual contexts, irrespective of whether the LLM is API-based or open-sourced.

Several studies have examined combining large language models with external resources. Shen et al. (2023) notably linked ChatGPT with HuggingFace, enhancing its decision-making range. However, this integration struggled with producing precise code due to inconsistencies in the ground truth code and insufficient documentation. In our study, we mitigated these limitations by utilizing human annotators to integrate each HuggingFace model as a function call. We further unified function arguments within the same domain, simplifying the evaluation process and reducing the complexity of model interactions during assessments.

3 MultiAPI Benchmark Dataset

3.1 Data Collection

In this section, we detail the process of constructing **MultiAPI** leveraging the HuggingFace instruction-code dataset introduced by Patil et al. (2023). The original dataset consists of a model definition file including model descriptions along with its corresponding example code template; and an instruction-code pair file linking models to self-generated instructions(Wang et al., 2023).

We first filtered out all the models that could potentially assist multimodal tasks from 9 unique domains, as shown in Table 5, and their corresponding instruction-code pairs. The subsequent data processing comprises four steps: 1) Description Verification, 2) Model Encapsulation, 3) Argument Standardization, and 4) Ground Truth Transformation. The primary procedures are illustrated in Figure 1. It's noteworthy that the first three steps are applied to the model definition and the last is applied to the instruction-code pair.

Description Verification: While most models come equipped with a description field that provides the basic information, the quality of these descriptions varies widely, largely depending on community contributors. Previous studies verified that a precise and detailed model description plays a critical role in aiding the model to identify the appropriate tool (Hsieh et al., 2023). Such specificity could also enhance the accuracy and reliability of evaluation outcomes. To this end, we engaged two human annotators with expertise in NLP to manually review all descriptions. They were tasked with

removing the model whose descriptions only offer a general overview, lacking a delineated use case, as depicted in lower Figure 1 (a).

Model Encapsulation: The primary utility of the original dataset was to facilitate the training or finetuning of LLMs to autonomously generate the API call code. Consequently, models were invoked using the `example_code` field present in the dataset, as illustrated in the upper section of Figure 1(b). To adapt the existing example codes to the API function-calling framework, we prompt *gpt-3.5-turbo* to transform the example code template into an API function and subsequently extract the potential arguments. In addition, we identify and include the import statements inside the function to ensure the function is independently executable.

Argument Standardization: Upon encapsulating the functions, we observe that while *gpt-3.5-turbo* transformed essential codes into function form, it exhibited challenges in accurately extracting function arguments. Further analysis suggests that the variation in argument names and the number of arguments pose a significant challenge (Yin et al., 2023b), potentially introducing the risk of hallucination, ambiguity and complicating the parsing process during argument evaluations. To address the aforementioned discrepancies, we introduce an argument standardization process. Consider a function set F_d within a given domain d. We define a standardized argument set A_d by manually reviewing all functions within d to determine the commonly recurring arguments intrinsic to the domain's functionality. As a result, for any functions within d, we require:

$$\forall f_1, f_2 \in F_d, \quad \text{args}(f_1) = \text{args}(f_2) = A_d \quad (1)$$

For instance, within the *Text to Image* domain, functions generate images in response to user prompts. Consequently, the indispensable arguments for this domain are `prompt` and `output_path`. The detailed mappings between domains and required arguments are listed in Table 5 in Appendix A.

Using this collated reference table, human experts are introduced to refine the generated functions ensuring: 1) Each function includes the minimum required arguments, named in line with the reference table. 2) Other arguments are listed as default arguments with default values. 3) Each function is executable within Python environments.

Ground Truth Transformation: As shown in the upper segment of Figure 1(c), instruction-code pairs represent specific instructions with their corresponding code blocks. To maintain consistency with our previous steps, we use a similar human-supervised approach to transform these pairs into instruction-function pairs. The results are depicted in the bottom code block of Figure 1(c). This ensures a consistent framework for both model definitions and their corresponding instructions.

3.2 Evaluation Metrics

The outputs of multimodal tasks are dependent on varying input modalities, leading to unpredictable results even with identical inputs (Rombach et al., 2022; Saharia et al., 2022). This variability makes direct evaluation of the output unreliable. Moreover, crafting robust evaluation metrics for each individual domain poses significant challenges for future versatility.

However, benefiting from diligent data collection steps, we bypass these issues by assessing the LLM's tool usage ability based on the function calls selected. In function-calling context, user's requirement would be fulfilled if the model correctly selects the appropriate function and fills in the accurate arguments. This approach streamlines the evaluation into a universal domain-agnostic text-matching task with some necessary adaptions.

Inspired by Li et al. (2023b), we design a stepwise, four-level evaluation framework for a comprehensive assessment of LLMs' tool usage in multimodal tasks. This framework includes:

1. **Invocation Assessment:** Tests if LLMs can discern when a user instruction necessitates an auxiliary function.

2. **Domain Match:** Evaluates the LLMs' ability to match the function's domain to the ground truth by leveraging domain annotations in our dataset.

3. **Function Match:** Conducts a detailed assessment to confirm whether the LLM correctly identifies the specific tool within the matched domain via their descriptions.

4. **Argument Match:** Verifies the LLM's proficiency in translating user instructions into precise arguments for successful function invocation. The distinction in evaluating multimodal task functions lies in the API arguments. We

classify arguments defined in Table 5 into two distinct categories: exact-match arguments and concept-match arguments. Exact-match arguments, such as file paths, demand precise, verbatim replication. Any deviation in these arguments can impede the successful invocation of the function. On the other hand, concept-match arguments, like generative prompts, offer more flexibility in wording, though they must maintain fidelity in conveying the intended meaning. Inaccuracies in generating concept-match arguments, while not hindering the function invocation, can lead to outputs that diverge from the expected results.

In our experiments, exact-match arguments undergo text matching for exact path alignment, while concept-match prompts are semantically evaluated using ROUGE F-scores (Lin, 2004) and cosine similarity (Lahitani et al., 2016) for both statistical and vectorized analysis.

4 Experiments

In this section, we extensively test our **MutilAPI** benchmark to evaluate LLMs' multimodal task handling via tool integration, covering API-based and open-source models. We explore various prompt configurations to find the most effective settings for multimodal tasks.

4.1 Task Formulation

Given a multimodal task instruction i, the model's objective is to generate an API function f from a set of available functions F and its corresponding set of arguments A_f. Formally, for $f \in F$ the generation process can be represented as:

$$p(f, A|i, F) = p(f|i, F) \times p(A|f, i) \quad (2)$$

4.2 Models and Prompt Configurations

Current LLMs can be categorized into API-based models and open-sourced models. Our evaluation performs on both categories. For API-based models, we use *gpt-3.5-turbo-0613* as the candidate. For open-sourced models, we leverage Llama2-13B (Touvron et al., 2023) provided by HuggingFace[1]. Furthermore, previous research proved prompt configurations can significantly affect the performance of LLMs (Zhang et al., 2023c; Wei et al., 2022). To investigate whether these configurations remain effective on our task. We implemented the following prompt configurations in our experiments:

In-context Learning: Previous research demonstrated the few-shot performance of language models can be significantly boosted by providing exemplar input-ground truth pairs (Brown et al., 2020). In our in-context setting, we provide 2 instruction-function call pairs to assist the model in reasoning the predictions.

Chain-of-Thought: Chain-of-Thought (Wei et al., 2023) adapts the concept of divide-and-conquer. It allows LLMs to address problems in a step-by-step paradigm, by deconstructing the primary task into smaller, manageable queries. This approach not only simplifies the task but also enhances the reasoning capabilities of the models. We apply this framework by breaking down the task into 4 questions aligned with our evaluation metrics introduced in 3.2. Those questions are listed in Appendix C.

Function Calling: Recently introduced by OpenAI[2], Function Calling is a feature tailored for GPT models. The models are finetuned on a specialized function-call dataset. The intent is to enable the models to better recognize scenarios necessitating function calls, thereby facilitating the generation of more structured outputs.

4.3 Context Token Limitation

Given the constraint of a maximum context window of 4,096 tokens for those LLMs used in our experiments, we face a limitation in the number of functions that can be included within this token budget. Our calculations suggest that approximately 25 functions can be accommodated. To effectively manage this constraint, we initially shuffle the entire dataset. Subsequently, we divide it into 10 segments, each containing 25 functions, except for the final segment. For each experiment configuration, we conduct separate trials on each of these 10 splits. The overall results are then derived by calculating the average across these 10 segments.

4.4 Function Invocation

In this section, we focus on the function invocation aspect of LLMs to evaluate their ability to under-

[1]https://huggingface.co/docs/transformers/main/model_doc/llama2

[2]https://platform.openai.com/docs/guides/function-calling

Model	Invoke Accuracy	Domain Accuracy	Function Accuracy
GPT-3.5	99.82	**71.78**	**52.94**
GPT-3.5-cot	**99.95**	71.43	51.73
GPT-3.5-ict	99.47	68.07	48.35
GPT-3.5-ict-cot	98.77	64.00	48.16
GPT-3.5-fc	**99.11**	**75.52**	**55.53**
GPT-3.5-fc-cot	94.13	70.00	50.12
GPT-3.5-fc-ict	95.02	67.72	49.59
GPT-3.5-fc-ict-cot	98.41	69.91	51.62
Llama	85.87	**14.75**	**9.94**
Llama-cot	79.88	12.76	6.37
Llama-ict	83.59	10.70	5.72
Llama-ict-cot	**86.30**	10.56	5.00

Table 1: Experimental results for function selection across different LLM configurations, where '-cot' denotes the use of Chain-of-Thought prompting, '-incontext' signifies incontext learning, and '-fc' indicates that the function calling feature is enabled.

Model	Argument Accuracy	R1	R2	RL	Sim
GPT-3.5	**42.68**	25.05	17.94	24.64	46.61
GPT-3.5-ict	36.37	30.37	21.32	29.68	50.82
GPT-3.5-cot	41.12	24.84	17.81	24.31	46.39
GPT-3.5-ict-cot	25.79	**32.45**	**22.78**	**31.95**	**53.97**
GPT-3.5-fc	**43.40**	24.17	15.42	23.39	44.64
GPT-3.5-fc-ict	32.26	**24.67**	**16.63**	**24.10**	44.05
GPT-3.5-fc-cot	38.26	24.53	15.45	23.85	**45.50**
GPT-3.5-fc-ict-cot	18.91	23.65	15.09	22.86	45.14

Table 2: Comparative evaluation of GPT-3.5 model configurations in argument generation. The first section shows the match accuracy of exact-match arguments while the second demonstrate the evaluation metrics of concept-match parameters. R1/2/L represents ROUGE-1/2/L scores respectively, and Sim represents cosine similarity.

stand user instructions and locate the proper tool function. The results are demonstrated in Table 1.

LLMs face challenges in multimodal domain selection: By observing across columns, we could conclude both GPT-3.5 and Llama models exhibit commendable accuracy in determining the necessity of function invocation based on user instructions. However, a significant drop in performance occurs when it comes to identifying the specific domain of multimodal tasks and selecting the precise function to effectively address these tasks. This finding implies that, while LLMs possess robust common-sense knowledge, they still struggle with accurately comprehending the nuances and definitions unique to each domain of multimodal tasks.

Function Calling enhancement performance varied by prompt configuration: Upon comparing the results in the first and second blocks of Table 1, it is evident that enabling Function Calling significantly enhances performance in the GPT-3.5 and GPT-3.5-ict-cot configurations, while it appears to slightly impede performance in settings where only a single prompt configuration is employed. This observation could potentially be attributed to the complex interplay between the Function Calling mechanism and the prompt configurations. Such findings underscore the importance of carefully considering the compatibility of various features and configurations when augmenting LLMs for specific tasks.

In-context learning impairs multimodal function invocation: Our analysis of the effectiveness of prompt configurations, conducted through a cross-row examination within each block, revealed consistent patterns across both GPT-3.5 and Llama models. A prominent observation is that the incorporation of contextual elements tends to negatively impact performance, a trend that is especially pronounced with the introduction of in-context learning. This significant impairment in performance is contrary to the widespread belief that providing reference context generally improves model performance across a variety of tasks. Such a result suggests that in multimodal function invocation scenarios, the addition of contextual information might inadvertently introduce complexity or irrelevant data, thus impairing the model's efficiency. This counterintuitive result suggests a need for more research into how context affects LLMs' function invocation, challenging current assumptions and opening new research avenues.

4.5 Argument Generation

The capabilities of LLMs in generating arguments for multimodal tasks are detailed in Table 2. It's noteworthy that Llama was excluded from this analysis due to its inferior performance in function locating. The results indicate a significant challenge for GPT models in accurately generating both exact-match and concept-match arguments based on user instructions. The success rate for matching exact-match arguments falls below 50%, and the semantic similarity of the generated concept-match arguments is similarly subpar. This suggests that argument generation set a more critical bottleneck hindering LLMs' ability to effectively invoke multimodal functions, compared to the function invocation ability in the previous sections.

Additionally, the data shows that while exact-

	GPT-3.5		GPT-3.5-fc	
Metric	**Func 1**	**Func 2**	**Func 1**	**Func 2**
Inv Acc	99.76	99.94	99.83	60.00
Dm Acc	76.67	36.67	66.67	40.00
Func Acc	53.33	30.00	46.67	40.00
Arg Acc	86.36	31.25	89.47	61.54
R1	63.17	67.25	69.68	68.57
R2	45.58	64.30	54.60	54.00
RL	61.15	67.25	69.67	67.69
Sim	83.28	75.69	86.68	70.45

Table 3: Sequential API invocation result on **MultiAPI-SEQ**. The metrics evaluated include Invocation Accuracy (Inv Acc), Domain Accuracy (Dm Acc), Function Accuracy (Func Acc), Argument Accuracy (Arg Acc), ROUGE-1/2/L (R1/2/L), Similarity (Sim).

match argument accuracy aligns with previous insights, adding context improves concept-match argument generation. This reveals that context enhances LLMs' semantic accuracy, indicating optimization potential, especially in improving concept-match handling in multimodal tasks without hindering exact-match performance.

4.6 Sequential API Invocation

In real-world applications, user instructions often require multiple API calls for resolution, especially in multimodal scenarios. This demands that LLMs understand each modality, its tasks, and their interactions. Analyzing sequential API invocation in models provides insights more representative of real-life applications and aids application development. To address this need, we introduce **MultiAPI-SEQ**, a dataset specifically designed for assessing sequential function invocation. This dataset has been carefully curated by human experts who have manually crafted 30 distinct instructions. Each of these instructions necessitates the sequential invocation of two functions from the **MultiAPI** dataset. By limiting each instruction to require just two functions, we aim to simplify the analysis process while still effectively evaluating the models' ability to handle multi-step task execution.

As shown in Table 3, both models exhibit high invocation accuracy initially, yet GPT-3.5-fc's accuracy notably diminishes during the second task. This indicates that while fine-tuning may enhance single-function call performance, it could adversely affect task planning in sequential API call tasks. Additionally, both models show a reduction in domain and function accuracy. The linguistic similarity metrics across functionalities indicate that GPT-3.5 demonstrates more consistent performance, hinting at its robustness in generating contextually appropriate responses throughout the task sequence.

5 Error Analysis

5.1 Domain Mismatch

Section 4.4 suggests LLMs struggle to differentiate multimodal task domains. We analyze model errors to identify these shortcomings. We summarize the result as a misclassification network indicating LLM's domain confusion in Figure 2.

For visual analysis APIs, the model demonstrates an inclination to misinterpret classification and segmentation tasks as object detection. Besides, it also frequently fails the identification between image classification and image segmentation. This pattern indicates a fundamental challenge in the LLM's ability to identify domains based on user instruction, particularly in discerning whether it should encompass the entire image or focus on the specific content within the image. The asymmetries in bidirectional error between these nodes further suggest that LLM bias towards local rather than global image analysis.

Additionally, image generation APIs often lead the model to confuse conditional and unconditional tasks, misidentifying text-to-image and image-to-image tasks as unconditional. It also struggles to recognize input modalities, confusing image-to-image with text-to-image tasks, indicating a possible lack of modality understanding due to they are trained on textual data.

5.2 Function Mismatch

To assess the LLMs' function selection accuracy, we randomly sampled 10 functions and corresponding instructions from each domain and prompted the model to choose the most appropriate function within that domain. As shown in Figure 2, the histogram reflecting function accuracy across domains, demonstrates the uneven function selection proficiency of LLMs in handling different multimodal tasks. Domains with more straightforward, visually dense tasks like image-to-image and object detection demonstrate relatively high accuracy, indicating that models perform better with tasks requiring less complex language-to-function mapping.

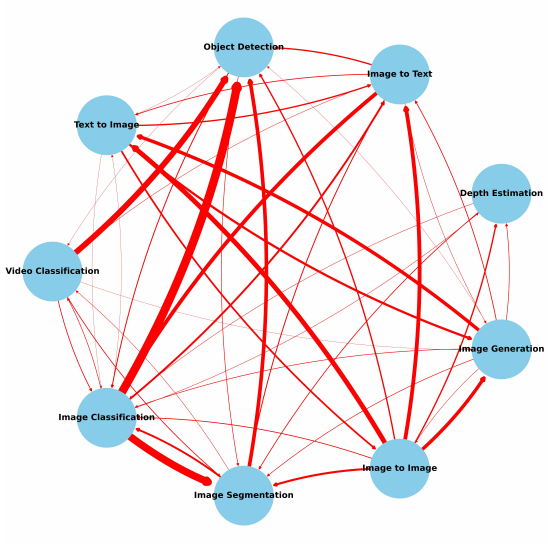

Figure 2: Domain misclassification network. Nodes in this graph represent distinct domains, with directed arrows illustrating instances where the model incorrectly applies a function from domain b intended for an instruction in domain a. The thickness of the arrows indicates the frequency of these errors, with thicker lines showing more common misclassifications.

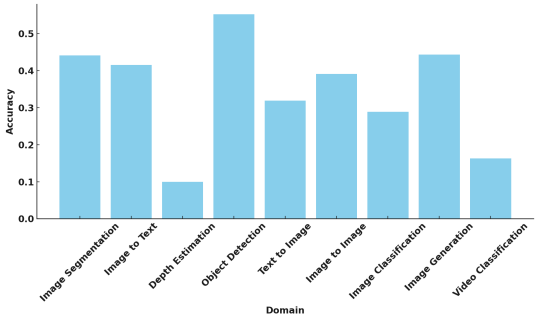

Figure 3: Function accuracy distribution for each domain.

6 Improvement Framework

Our analysis in Sections 4 and 5 reveals that LLMs primarily struggle with distinguishing domains and modalities, with argument generation as a significant bottleneck. To mitigate these challenges, we propose two intuitive yet effective solutions: domain description prompting and argument revision.

Domain description prompting involves adding a sentence to the model's system prompt to clearly define each domain. In addition, in visual analysis tasks, we specify whether the domain conducts global or local image analysis.

Building on research showing LLMs' effectiveness in evaluation and revision tasks (Liu et al., 2023; Zhang et al., 2023c), we employ a secondary LLM as an argument editor. This LLM checks and revises argument predictions to ensure they align with user instructions, reducing task complexity and the context length for the primary LLM.

Metric	GPT-3.5	GPT-3.5-dp-ac
Inv Acc	99.82	**99.87**
Dm Acc	71.78	**76.31**
Func Acc	51.73	**59.47**
Arg Acc	42.68	**48.82**
R1	25.05	**27.76**
R2	17.94	**18.45**
RL	24.64	**26.33**
Sim	46.61	**56.82**

Table 4: The result of adding detailed domain description prompting (-dp) and argument correction (-ac).

To avoid the noise arising from complex interactions between function calling feature and input context, we conducted our experiments using the GPT-3.5 model. Table 4 illustrates that our approach enhanced performance across all evaluation metrics. Notably, there was a significant improvement in domain accuracy, argument exact matching, and semantic evaluation. This significant improvement not only affirms the effectiveness of our approach but also strongly validates the accuracy of our analysis. Furthermore, we observed a notable enhancement in function accuracy, attributed to the incorporation of domain descriptions.

7 Conclusion

In this paper, we presented a comprehensive study on the application of LLMs to multimodal tasks with external API functions, using the newly introduced **MultiAPI** dataset. Our findings highlight the capabilities and limitations of LLMs in function calling. We revealed a significant discrepancy between the models' ability to recognize the need for function calls and their accuracy in selecting appropriate domains, functions, and arguments. This insight led us to propose a novel approach focusing on domain description prompting and argument revision, which demonstrated improved performance in addressing these challenges. Our work contributes to the field by introducing the first large-scale multimodal instruction-function benchmark dataset and providing a detailed analysis of LLMs in multimodal task execution. We hope our dataset and findings could assist the development of tool-augmented LLMs and more sophisticated models for complex real-world applications.

Limitations

Our model selection was confined to *gpt-3.5-turbo* and Llama2-13B due to computational and budget constraints. While our extensive experiments and improvement framework offer valuable insights, we acknowledge limitations. We only briefly touched upon areas like detailed sequential API invocation analysis and in-depth examination of the improvement framework's outcomes. Further comprehensive research in these areas is necessary and anticipated for future works.

References

Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. 2023. Foundational models defining a new era in vision: A survey and outlook. *arXiv preprint arXiv:2307.13721*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

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

Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. 2016. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International Conference on Cyber and IT Service Management*, pages 1–6.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Huang. 2023. A systematic study and comprehensive evaluation of ChatGPT on benchmark datasets. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 431–469, Toronto, Canada. Association for Computational Linguistics.

Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. 2023a. Multimodal foundation models: From specialists to general-purpose assistants.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Api-bank: A comprehensive benchmark for tool-augmented llms.

Zihao Li, Zhuoran Yang, and Mengdi Wang. 2023c. Reinforcement learning with human feedback: Learning dynamic choices via pessimism.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Chenyang Lyu, Minghao Wu, Longyue Wang, Xinting Huang, Bingshuai Liu, Zefeng Du, Shuming Shi, and Zhaopeng Tu. 2023. Macaw-llm: Multi-modal language modeling with image, audio, video, and text integration. *arXiv preprint arXiv:2306.09093*.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. 2023. Video-chatgpt: Towards detailed video understanding via large vision and language models.

OpenAI. 2023. Gpt-4 technical report.

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

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

Jielin Qiu, Jiacheng Zhu, William Han, Aditesh Kumar, Karthik Mittal, Claire Jin, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Bo Li, Ding Zhao, and Lijuan Wang. 2023. Multisum: A dataset for multimodal summarization and thumbnail generation of videos.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face.

Yifan Song, Weimin Xiong, Dawei Zhu, Cheng Li, Ke Wang, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world applications via restful apis. *arXiv preprint arXiv:2306.06624*.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2023. Next-gpt: Any-to-any multimodal llm.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2023a. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Tong Xu, Hao Wang, Dianbo Sui, Yunhang Shen, Ke Li, Xing Sun, and Enhong Chen. 2023b. Woodpecker: Hallucination correction for multimodal large language models. *arXiv preprint arXiv:2310.16045*.

Hang Zhang, Xin Li, and Lidong Bing. 2023a. Video-llama: An instruction-tuned audio-visual language model for video understanding.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023b. Extractive summarization via chatgpt for faithful summary generation. *arXiv preprint arXiv:2304.04193*.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023c. Summit: Iterative text summarization via chatgpt.

Jiawei Zhang. 2023. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2023d. How language model hallucinations can snowball.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools.

Appendix

A Domain and Required Arguments Mapping

Domains	Required Arguments	# Functions
Text to Image	`(prompt: str, output_path:str)`	11
Depth Estimation	`(image_path:str, output_path:str)`	10
Object Detection	`(image_path: str)`	30
Video Classification	`(video_path:str)`	23
Image Classification	`(image_path: str)`	48
Image to Text	`(image_path: str)`	28
Image Generation	`(output_path:str)`	33
Image Segmentation	`(image_path: str, prompt:str)`	29
Image to Image	`(control_image_path:str, output_image_path:str)`	23

Table 5: The domains of MultiAPI and their required arguments. # Functions represents the number of functions that each domain contains.

Table 5 displays the quantity of functions per domain, along with a comprehensive mapping between each domain and its requisite arguments. It's important to note that these required arguments constitute a subset of the parameters for each function in the respective domain, owing to the specific functionality. To maintain argument consistency within the domain, we have designated other arguments as optional arguments and assigned them default values.

B Dataset Construction Prompts

Role	Content
System	You are an expert python code rewriter. You are very good at calling the function with the correct arguments.
User	Given the function described as '{description}' with the signature `def {function_name(}{' ,'.join([{arg} for arg in {function_arguments}.keys()])}):`, the function code is {function_code} and the arguments description is {function_arguments}. Here's the task, this code '{code}' is doing the same thing as the function call. Please rewrite the code to the function call, you will need to find the right arguments for the function and call it. The image_path and video_path arguments will always be a single image, or video. Please output the function call with right arguments filled in, please use the format of (argument_name=argument_value) do not omit the default value even you don't need to change it. The parameter related to path can not be '' or empty. If the path is not mentioned, use './input.png' and './output.png' for images input and output and './input.mp4' and './output.mp4' for video input and output as default. The text related parameter should always be a string, if the text is not mentioned, use 'This is a test text' as default. Only output the function call.

Table 6: The system and user prompts used in Model Encapsulation and Ground Truth Transformation steps.

In Table 6, we list prompts used in **MultiAPI** benchmark dataset construction process. Specifically, we prompt gpt-3.5-turbo to transform the code which calling a specific HuggingFace model to a Python function. Note that according to OpenAI's document, the model could receive two categories of prompts: system prompt and user prompt, where the system prompt functions as the global instruction to initialize the model and the user prompt as the question proposed by users. In our experiment, we leverage both prompts to guide the model.

System Prompt	User Prompt
Default: You are an expert multimodal assistant that solves multimodal tasks with the provided functions. **Function Call:** You are an expert multimodal assistant that solves multimodal tasks with the provided functions. For most of the time, you need to call the functions to solve the task and only one function is needed. <BEGIN_FUNCTION_LIST> {function_definitions} <END_FUNCTION_LIST> For most of the time, you need to call the functions to solve the task and only one function is needed.	**Default:** Here is the user's instruction: {instruction}, please help solve the task. Please only use the functions provided. **CoT:** Here is the user's instruction: {instruction}, please help solve the task.You can solve the problem follow these steps but please DO NOT answer these questions in your response this is just for your reference: 1. What is the domain of the task? The options are: {domains} 2. Do you need to call the functions? 3. Which function to call? 4. What are the arguments of the function? **Incontext:** Here is the user's instruction: {instruction}, please help solve the task. Please only use the functions provided. Here's some examples for your reference: {Exemplar Instruction-Function Pairs}

Table 7: System and user prompts for each experiment configurations.

C Experiment Prompts

We listed the system and user prompts we used for each configuration in Table 7.

D Case Study

In Table 8, we delineate the correct and incorrect function calls, with the first column illustrating instances of accurate calls and the second column showcasing erroneous ones. Each row presents a correct and incorrect function call example for the same function. The example in the top left shows a well-structured instruction, indicating the recommended instruction involving domain-specific keywords followed by function identification. This structure is exemplified by the explicit focus on the primary objective, as illustrated in the instruction: "Generate butterfly images." Conversely, the example presented in the top right serves as a counterexample, revealing the model's diminished accuracy in selecting the correct function when confronted with a vague term like "need" in the instruction, especially in the presence of numerous diverse domain functions, thus leading to ambiguity. In such cases, the model may misinterpret the instruction, leading to the erroneous employment of functions from unrelated domains and generating different function arguments.

The examples in the bottom left and bottom right show that even when the function description is not fully related to the intention of the instruction, the model demonstrates an understanding of the function's name, allowing it to align instructions with functions that share similar keywords. For instance, the association between "google_ddpm_celebahq_256" and "celebrity faces" illustrates this capability. In summary, to augment the multitasking proficiency of a Large Language Model (LLM), it is advisable to furnish a precise instruction followed by domain-specific keywords, the model description, and a recommended function name that succinctly captures the functionality of the designated task.

Correct Function Call	Incorrect Function Call
Instruction Description: We are running a butterfly-themed event and need to generate butterfly images for our marketing campaign. **Expect Model Description:** This model is a diffusion model for unconditional image generation of cute butterflies. **Domain:** Computer Vision Unconditional Image Generation **Expect function:** MFawad_sd_class_butterflies_32 **Expect Function Argument:** output_path='output.png' **Predict Function:** MFawad_sd_class_butterflies_32 **Predict Function Argument:** output_path: './output.png'	**Instruction Description:** One of our clients is launching a new board game with a butterfly theme. They need images of multiple butterfly illustrations for the game cards. **Expect Model Description:** This model is a diffusion model for unconditional image generation of cute butterflies. **Domain:** Computer Vision Unconditional Image Generation **Expect function:** MFawad_sd_class_butterflies_32 **Expect Function Argument:** output_path: './output.png' **Predict Model Description:** GreeneryScenery/SheepsControlV3 is a model for image-to-image tasks. It can be used to generate images based on the input image and optional text guidance. The model has some limitations, such as the conditioning image not affecting the output image much. Improvements can be made by training for more epochs, using better prompts, and preprocessing the data. **Domain:** Computer Vision Image-to-Image **Predict Function:** GreeneryScenery_SheepsControlV3 **Predict Function Argument:** (image_path:'./input.png',text_guidance:'Illustration of a butterfly')
Instruction Description: A synthetic company wants its model to automatically create a set of 256x256 images to be included in their website. They want to resemble celebrity faces but be out of their provided database. **Expect Model Description:** Denoising Diffusion Probabilistic Models (DDPM) for high quality image synthesis. Trained on the unconditional CIFAR10 dataset and 256x256 LSUN, obtaining state-of-the-art FID score of 3.17 and Inception score of 9.46. **Domain:** Computer Vision Unconditional Image Generation **Expect Function:** google_ddpm_celebahq_256 **Expect Function Argument:** output_path='./generated_celebrity_face.png' **Predict Function:** google_ddpm_celebahq_256 **Predict Function Argument:** output_path: './output.png'	**Instruction Description:** Develop a movie poster generator using the computer vision API mentioned above. Hint: generate a human-like face as the main poster background and save it as a file named "movie_poster.png". **Expect Model Description:** Denoising Diffusion Probabilistic Models (DDPM) for high quality image synthesis. Trained on the unconditional CIFAR10 dataset and 256x256 LSUN, obtaining state-of-the-art FID score of 3.17 and Inception score of 9.46. **Domain:** Computer Vision Unconditional Image Generation **Expect Function:** google_ddpm_celebahq_256 **Expect Function Argument:** output_path='movie_poster.png' **Predict Model Description:** Dream Shaper is a text-to-image model that generates artistic images based on the given input text. Read more about this model here: https://civitai.com/models/4384/dreamshaper **Domain:** Multimodal Text-to-Image **Predict Function:** Lykon_DreamShaper **Predict Function Argument:** prompt=": "humanlike face"

Table 8: Case Study