

LLMs and Tools

Function Calling

Advanced Large Language Models

ELL8299 · AIL861 · ELL881



Dinesh Raghu
Senior Researcher & Manager, IBM Research

Recap

1. Training Verifiers to Solve Math Word Problems (GSM8K)
 - **Introduced tool augmentation** in LLMs during finetuning
2. TALM
 - Introduced a general framework
 - **Cannot generalize** and requires a lot of human annotations
3. PAL
 - Requires fewer human annotations
 - **Cannot generalize** to unseen tools during test
4. Toolformer
 - Generalizes to multiple tasks
 - Requires fewer human annotations
 - **Cannot generalize** to unseen tools during test



LLMs and Tools

Part 1: Incorporating Tools during Fine-tuning (Tool Augmentation)

Part 2: Teaching LLMs to Use External APIs (Function Calling)

Part 3: Automating Complex Tasks (AI Agents)



Motivation

Let's say you are tasked to build a chatbot for IIT Delhi students using LLMs.

The goal is to add a conversational interface that supports

1. institute rules/policy related queries

How would you build such a chatbot?



Motivation

Let's say you are tasked to build a chatbot for IIT Delhi students using LLMs.

The goal is to add a conversational interface that supports

1. institute rules/policy related queries
2. searching/adding/dropping courses

How would you build such a chatbot?



What is Function Calling?

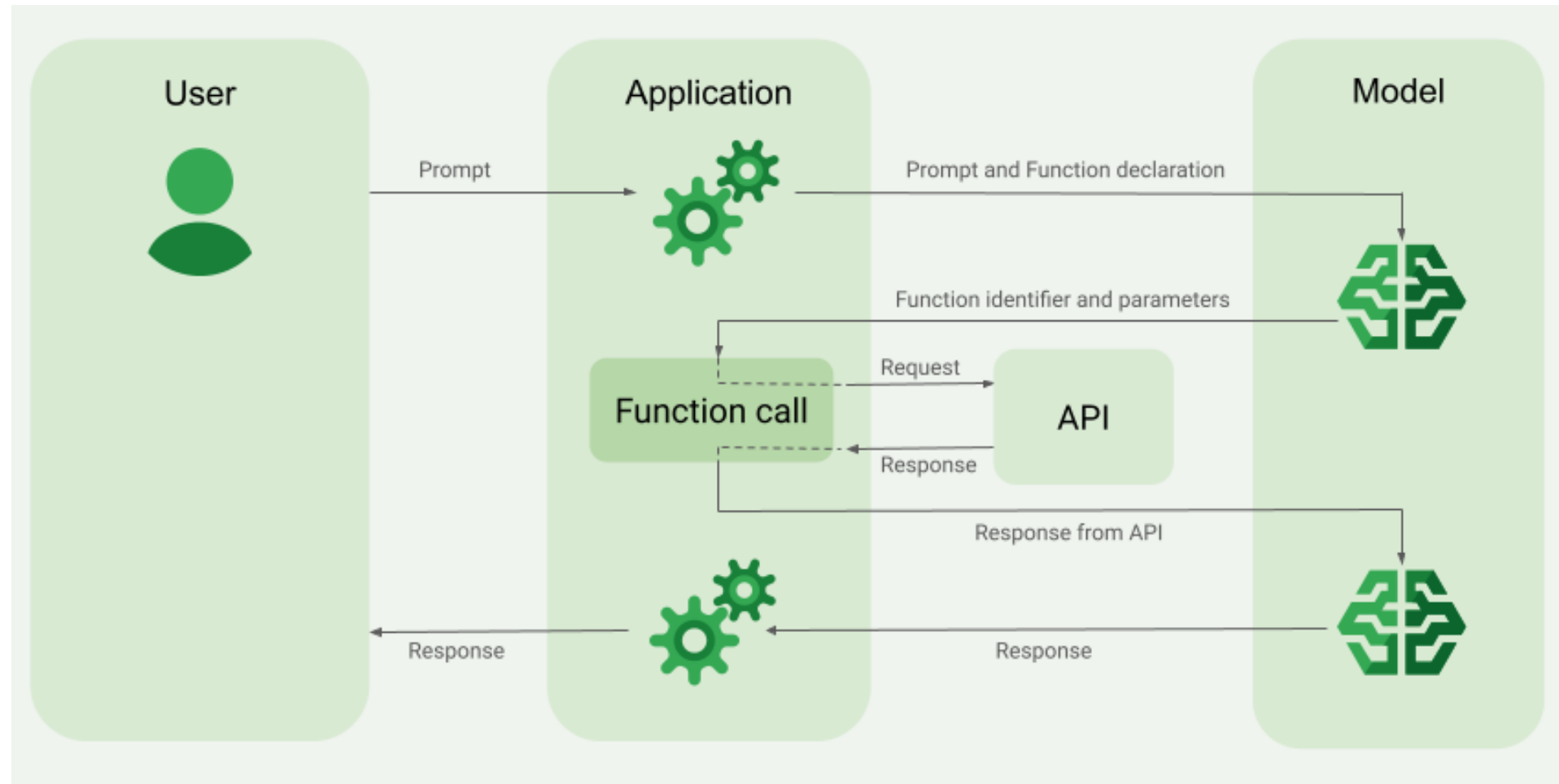


Image credit: <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling>



What is Function Calling?

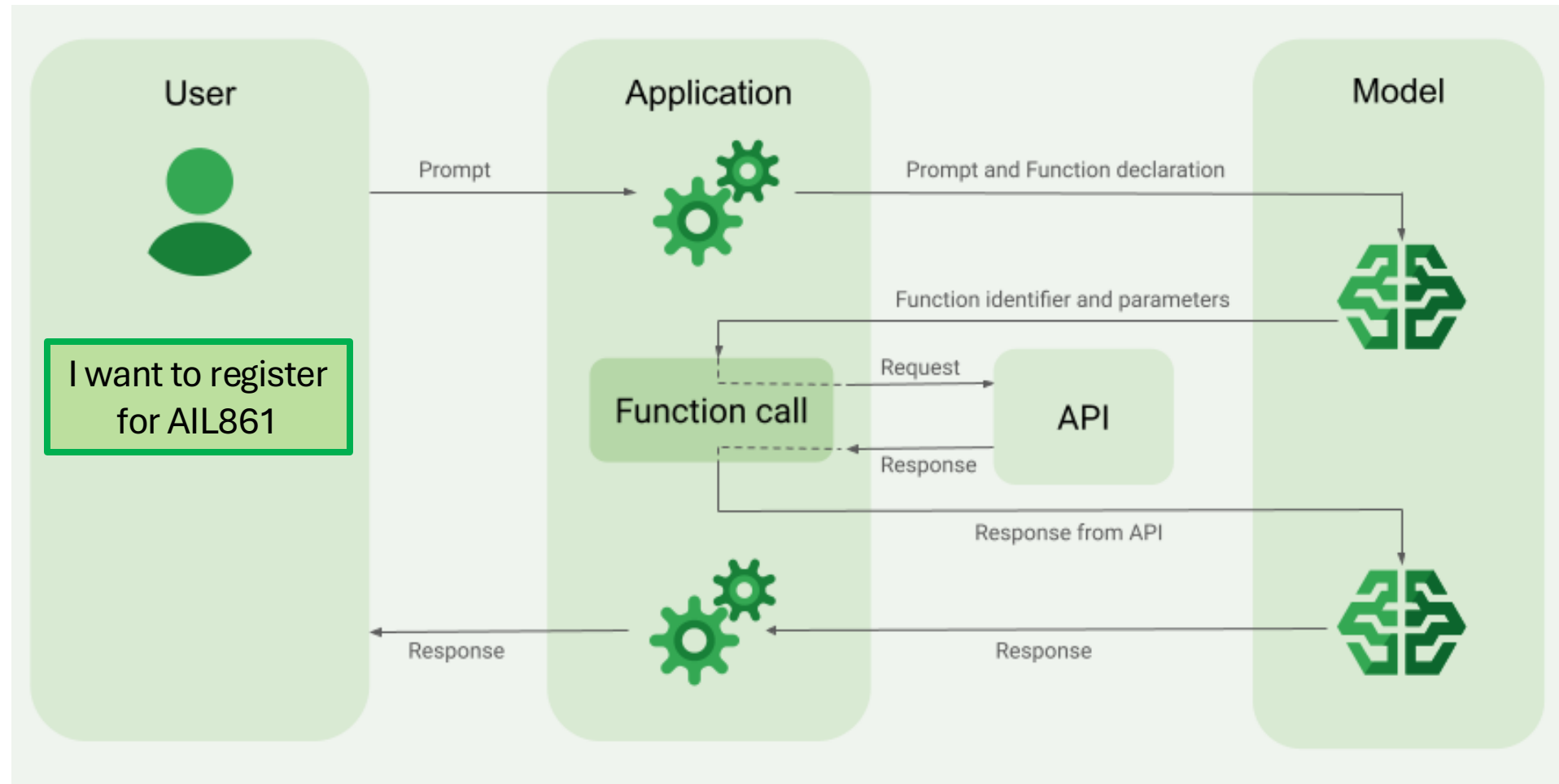


Image credit: <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling>



What is Function Calling?

Tools-Available: add_course, drop_course, search_course
User: I want to register for AIL861

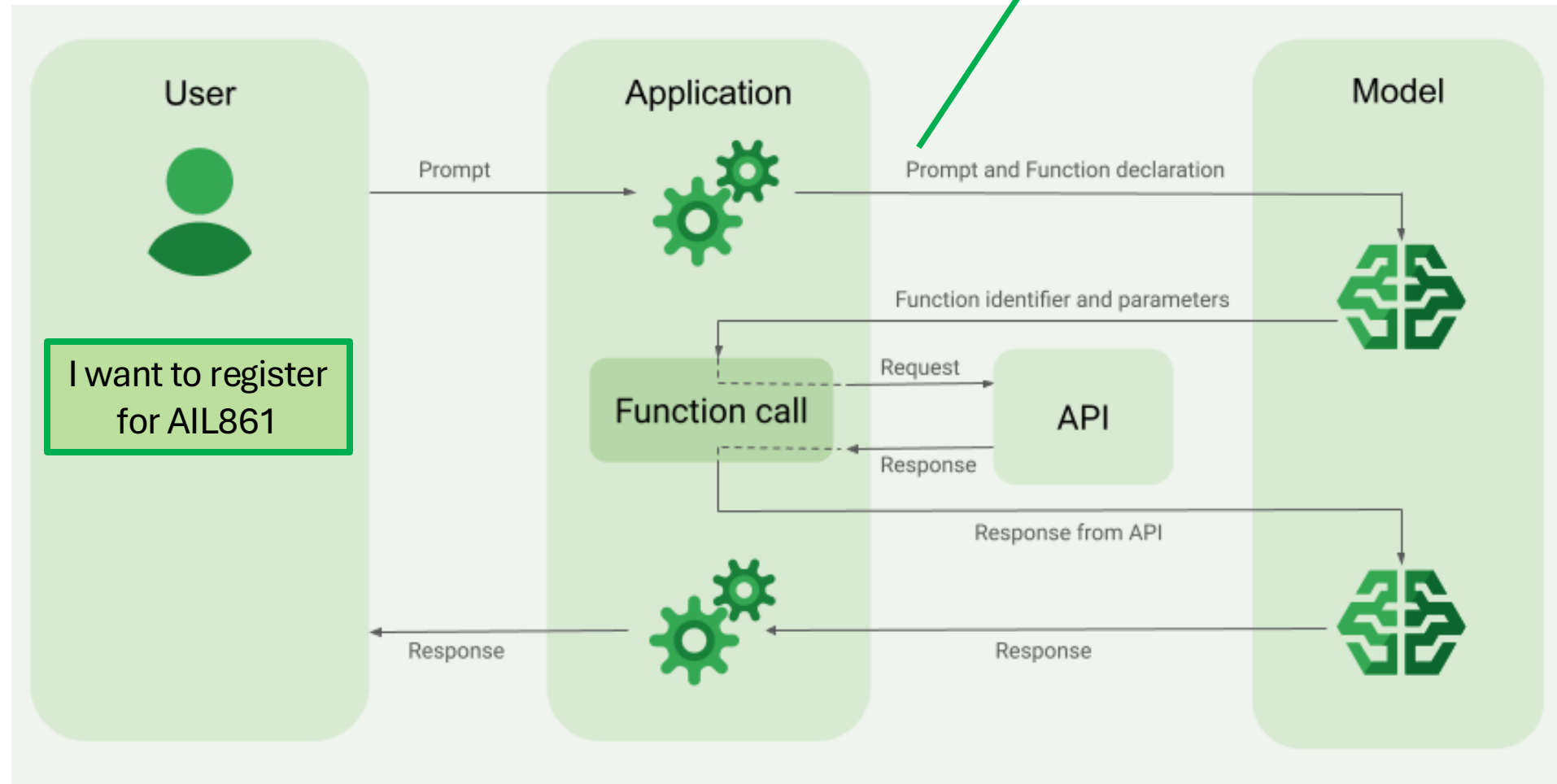


Image credit: <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling>



What is Function Calling?

```
|tool_call|  
add_course(student_id=XXXXX,  
course_id=AIL861)
```

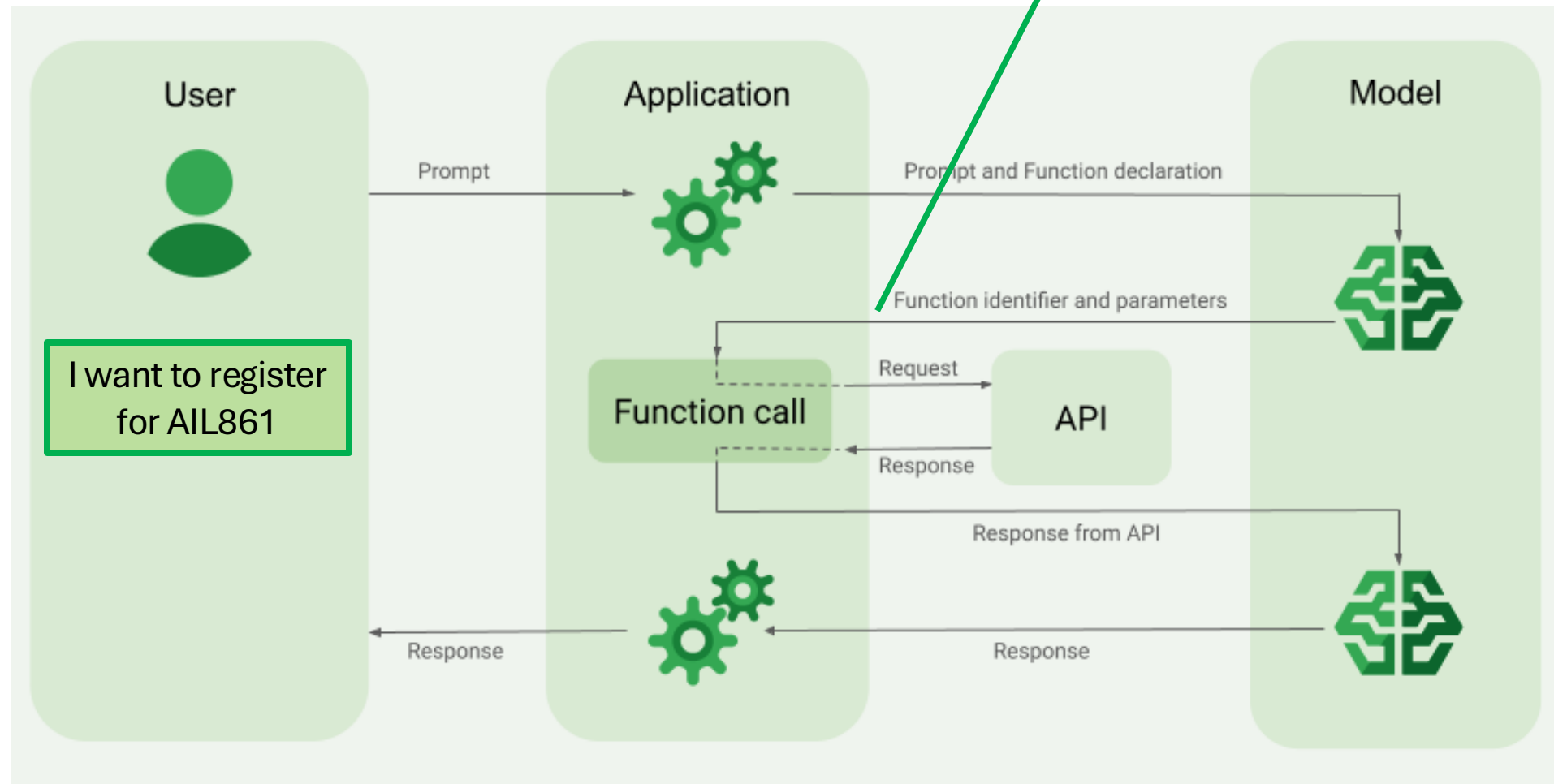
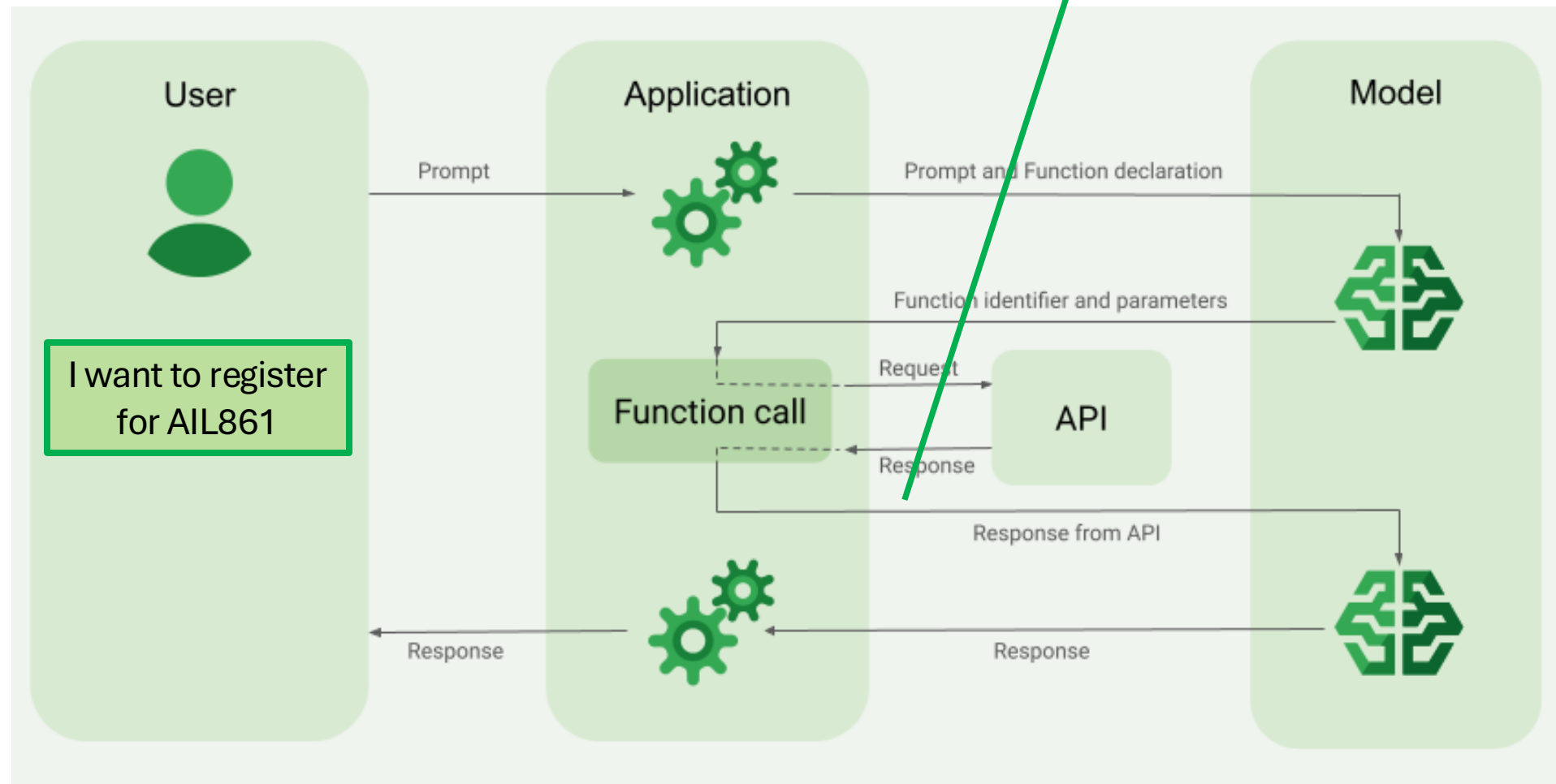


Image credit: <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling>



What is Function Calling?



```
{ "student_id"="XXXXXX",  
  "course_id"="AIL861",  
  "status"="registered")
```

Image credit: <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling>



What is Function Calling?

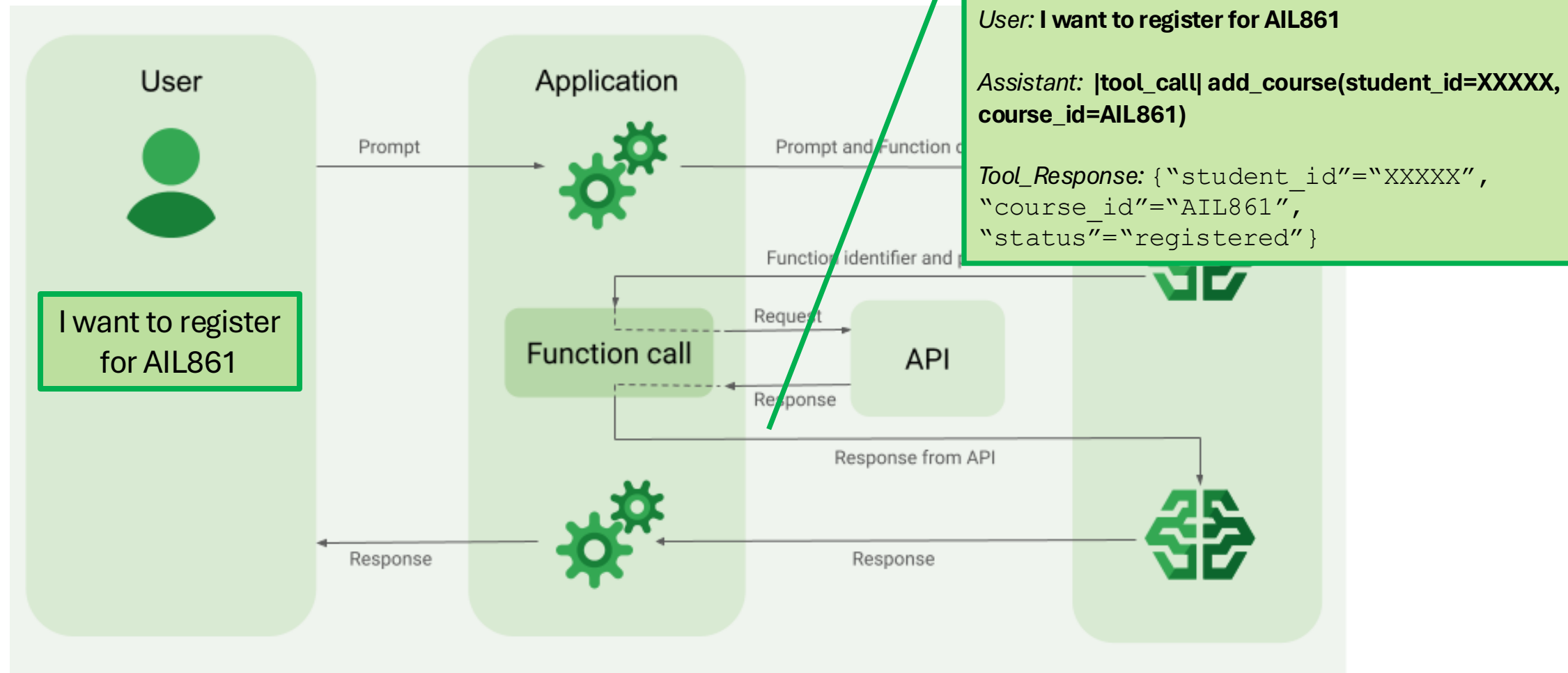
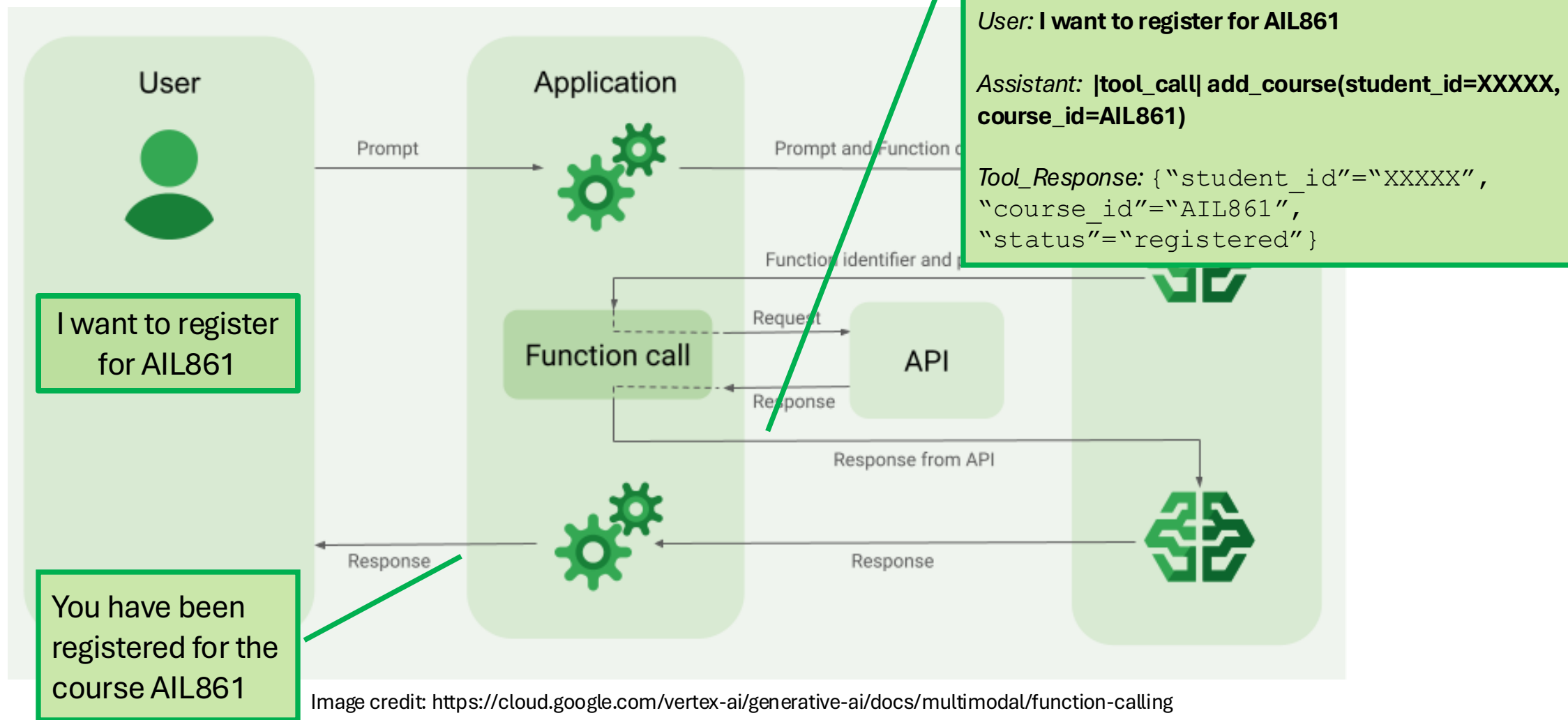


Image credit: <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling>



What is Function Calling?



How are tools specified?

```
{  
  "type": "function",  
  "function": {  
    "name": "retrieve_payment_status",  
    "description": "Get payment status of a transaction",  
    "parameters": {  
      "type": "object",  
      "properties": {  
        "transaction_id": {  
          "type": "string",  
          "description": "The transaction id.",  
        },  
      },  
      "required": ["transaction_id"],  
    },  
  },  
},
```

Image credit: https://docs.mistral.ai/capabilities/function_calling/



How are tools specified?

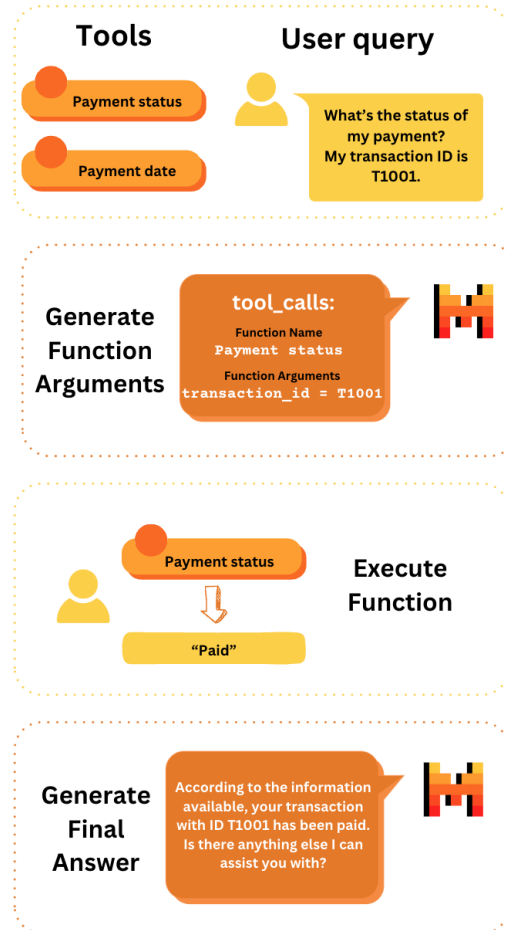
```
{
  "type": "function",
  "function": {
    "name": "retrieve_payment_status",
    "description": "Get payment status of a transaction",
    "parameters": {
      "type": "object",
      "properties": {
        "transaction_id": {
          "type": "string",
          "description": "The transaction id.",
        }
      },
      "required": ["transaction_id"],
    },
  },
},
```

```
def retrieve_payment_status(df: data, transaction_id: str) -> str:
    if transaction_id in df.transaction_id.values:
        return json.dumps({'status': df[df.transaction_id ==
transaction_id].payment_status.item()})
    return json.dumps({'error': 'transaction id not found.'})
```

Image credit: https://docs.mistral.ai/capabilities/function_calling/



What is Function Calling?



1. Agent developer defines a set of tools
2. User enters a query
3. Model identifies the function and its arguments
4. Agent executes the function to obtain tool results
5. Model uses the results to generate the final answer

Image credit: https://docs.mistral.ai/capabilities/function_calling/



How do we teach this to LLMs?



What data do we need?

Tools-Available: add_course, drop_course, search_course
User: I want to register for AIL861

|tool_call|
add_course(student_id=XXXXX,
course_id=AIL861)

Tools-Available: add_course, drop_course, search_course

User: I want to register for AIL861

Assistant: |tool_call| add_course(student_id=XXXXX, course_id=AIL861)

Tool_Response: {"student_id":"XXXXX",
"course_id":"AIL861",
"status":"registered"}

You have been registered for the course AIL861



Outline

1. Gorilla
2. ToolAlpaca
3. ToolLLM
4. APIGen



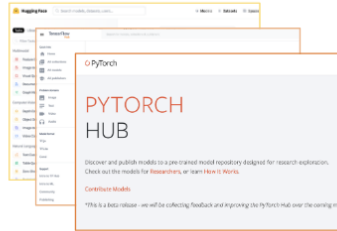
Gorilla: Large Language Model Connected with Massive APIs

1. Synthesized a function calling dataset named APIBench
 - Using model cards in HuggingFace Model Hub, Torch hub, and Tensorflow Hub
2. Finetuned LLaMA-7B model with APIBench to create the Gorilla model

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs

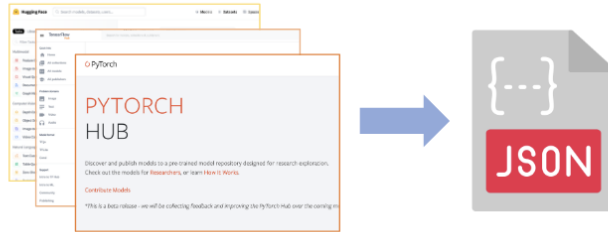


Dataset curation: 1,645 API calls. 94 from Torch Hub (exhaustive), 626 from TensorFlow Hub v2 (exhaustive) and 925 from HuggingFace (Top 20 in each domain).

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs



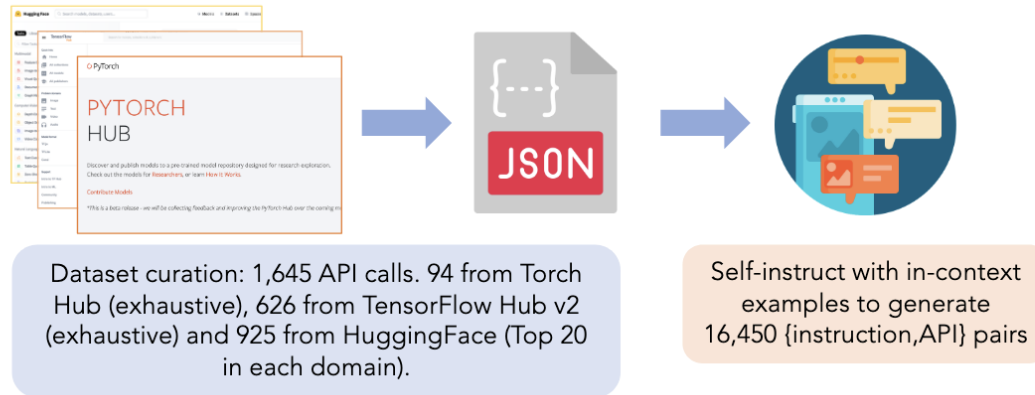
Dataset curation: 1,645 API calls. 94 from Torch Hub (exhaustive), 626 from TensorFlow Hub v2 (exhaustive) and 925 from HuggingFace (Top 20 in each domain).

<Reference API>: {domain: Object Detection, framework: PyTorch, functionality: Traffic Object Detection, Drivable Area Segmentation, Lane Detection, api_name: HybridNets, api_call: torch.hub.load(repo_or_dir='datvuthanh/hybridnets', model='hybridnets', pretrained=True), api_arguments: repo_or_dir, model, pretrained, python_environment_requirements: Python>=3.7, PyTorch>=1.10, example_code: 'import torch; model = torch.hub.load('datvuthanh/hybridnets', 'hybridnets', pretrained=True); img = torch.randn(1,3,640,384); features, regression, classification, anchors, segmentation = model(img), performance: dataset: [name: BDD100K, accuracy: Traffic Object Detection: Recall (%): 92.8, mAP@0.5 (%): 77.3, Drivable Area Segmentation: Drivable mIoU (%): 90.5, Lane Line Detection: Accuracy (%): 85.4, Lane Line IoU (%): 31.6], description: HybridNets is an end2end perception network for multi-tasks. Our work focused on traffic object detection, drivable area segmentation and lane detection. HybridNets can run real-time on embedded systems, and obtains SOTA Object Detection, Lane Detection on BDD100K Dataset. }

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



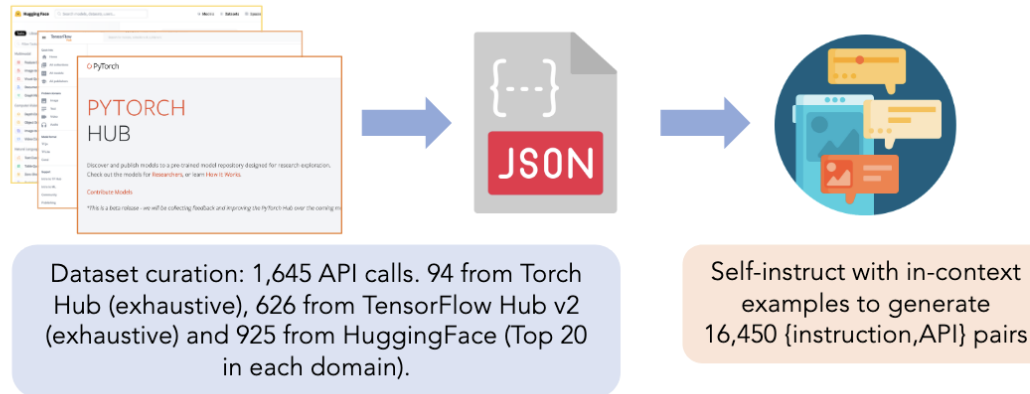
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs



User: Our customer is a zoo and we want to help them detect movement of different animals. Write a python program in 1 to 2 lines to call API in TensorFlowHub.

Assistant:

<api_call>: `detector = hub.load('https://tfhub.dev/google/openimages_v4/ssd_mobilenet_v2/1')`,

User: I am an engineer at Uber and I need to find a API that can classify pedestrians, cars, etc. from an image of the scene. Write a python program in 1 to 2 lines to call API in TorchHub.

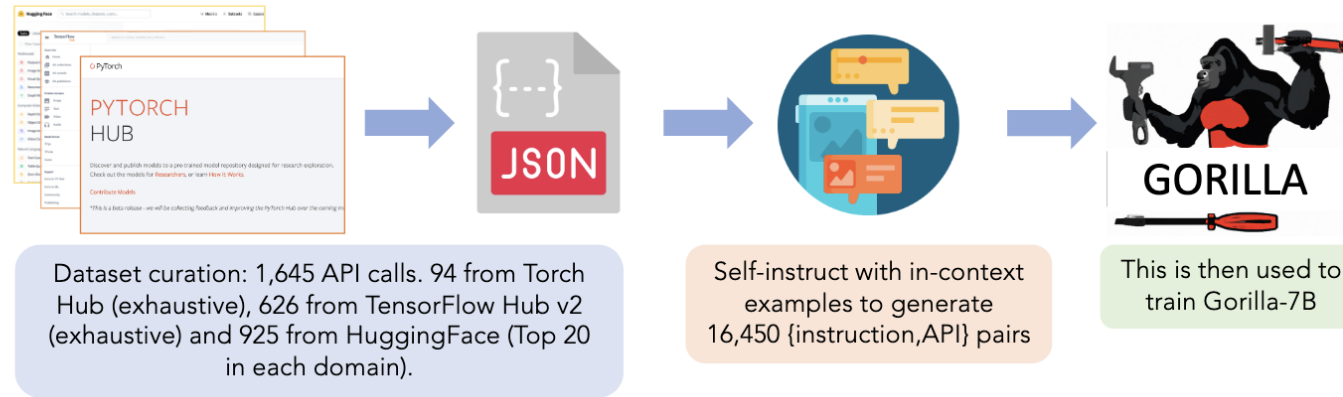
Assistant:

<api_call>: `model = torch.hub.load('datvuthanh/hybridnets', 'hybridnets', pretrained=True),`

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs



*"I want to see
some cats dancing
in celebration!"*

*Gorilla: Large Language Model Connected with Massive APIs, |



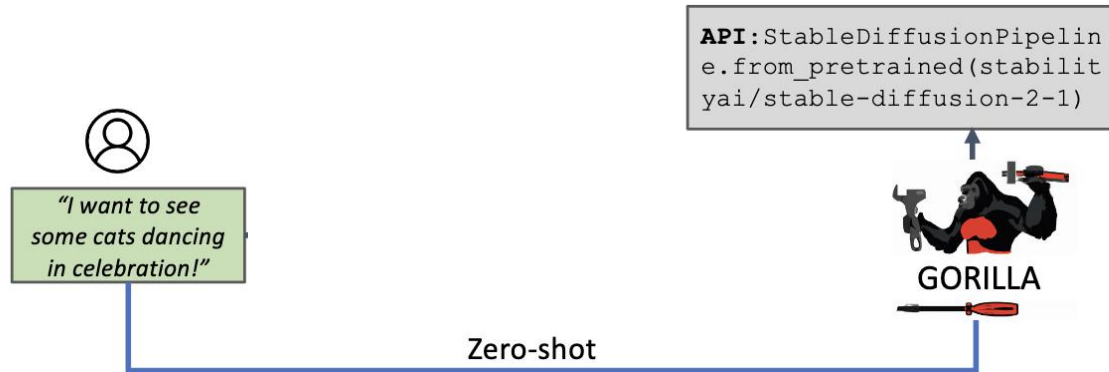
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



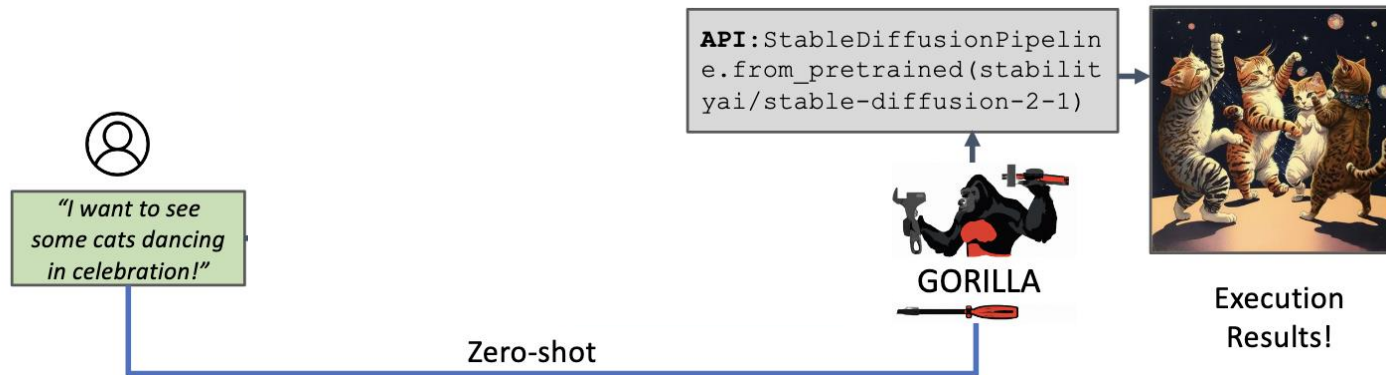
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs



Issues with this type of training:

- Cannot adapt to unseen APIs
- Can adapt to change in API specs

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs

User: I am an engineer at Uber and I need to find a API that can classify pedestrians, cars, etc. from an image of the scene. Write a python program in 1 to 2 lines to call API in TorchHub.

<Reference API>: {domain: Object Detection, framework: PyTorch, functionality: Traffic Object Detection, Drivable Area Segmentation, Lane Detection, api_name: HybridNets, api_call: torch.hub.load(repo_or_dir='datvuthanh/hybridnets', model='hybridnets', pretrained=True), api_arguments: repo_or_dir, model, pretrained, python_environment_requirements: Python>=3.7, PyTorch>=1.10, example_code: 'import torch; model = torch.hub.load('datvuthanh/hybridnets', 'hybridnets', pretrained=True); img = torch.randn(1,3,640,384); features, regression, classification, anchors, segmentation = model(img), performance: dataset: [name: BDD100K, accuracy: Traffic Object Detection: Recall (%): 92.8, mAP@0.5 (%): 77.3, Drivable Area Segmentation: Drivable mIoU (%): 90.5, Lane Line Detection: Accuracy (%): 85.4, Lane Line IoU (%): 31.6], description: HybridNets is an end2end perception network for multi-tasks. Our work focused on traffic object detection, drivable area segmentation and lane detection. HybridNets can run real-time on embedded systems, and obtains SOTA Object Detection, Lane Detection on BDD100K Dataset.}

Assistant:

<api_call>: model = torch.hub.load('datvuthanh/hybridnets', 'hybridnets', pretrained=True),

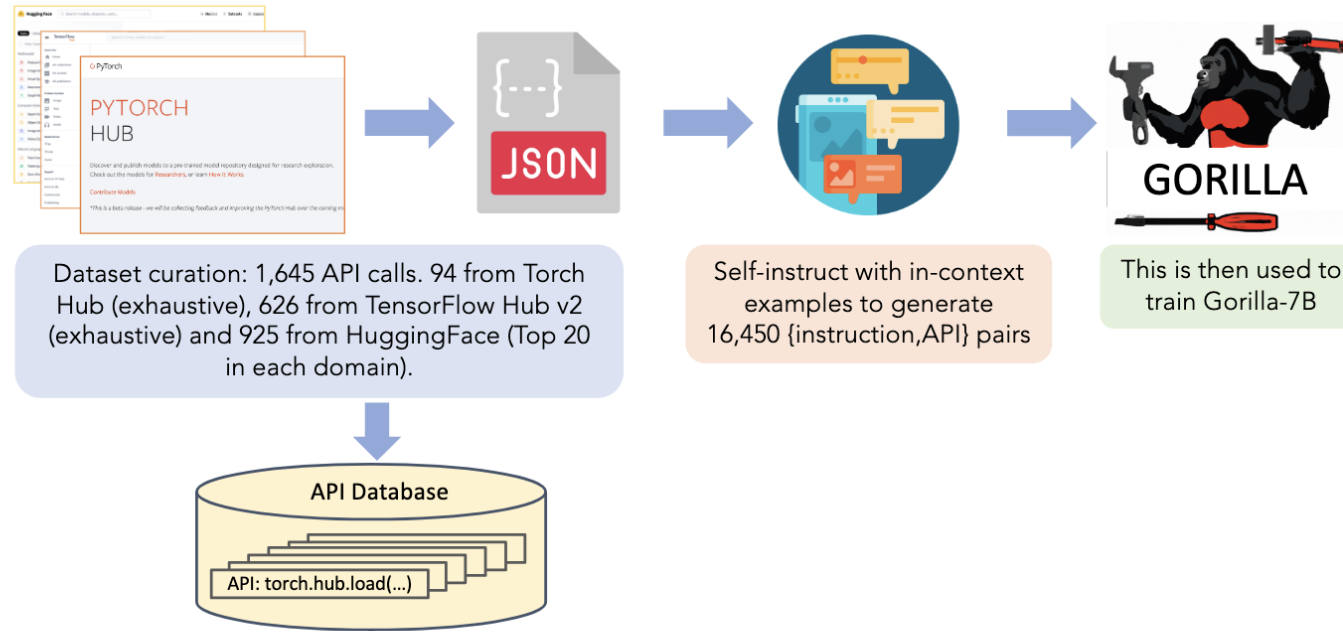
First to use retriever augmented training for APIs

- Leads to better generalization
- Can adapt to change in API specs

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



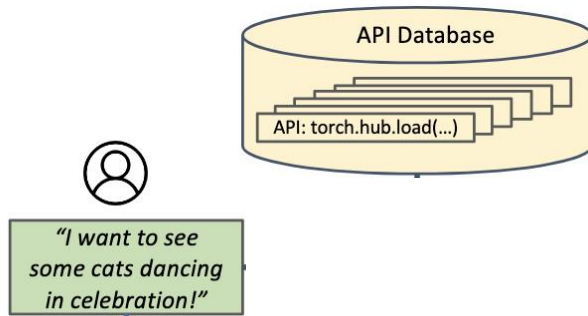
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



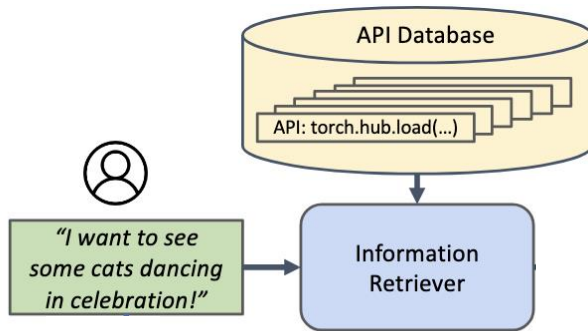
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



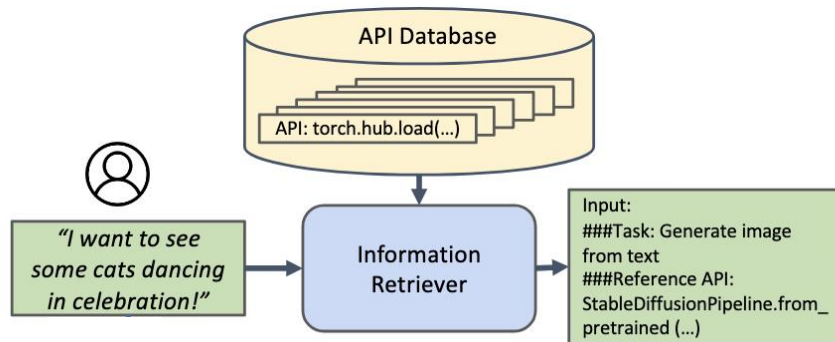
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



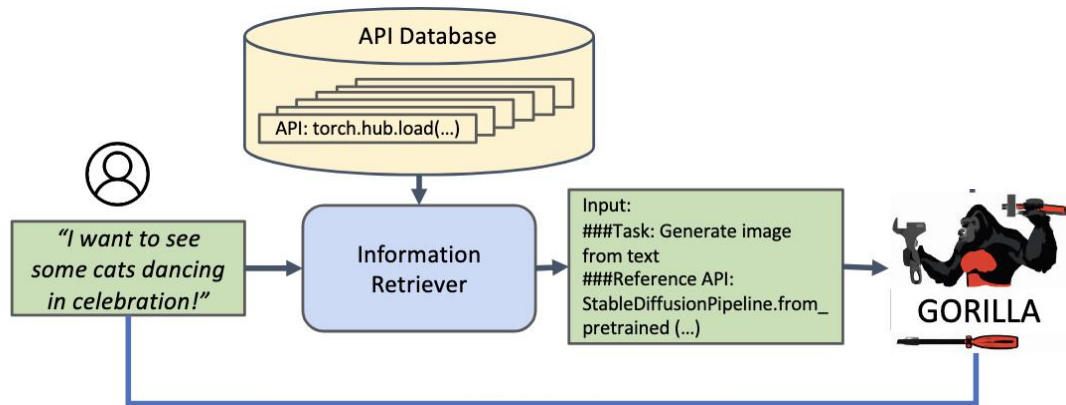
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



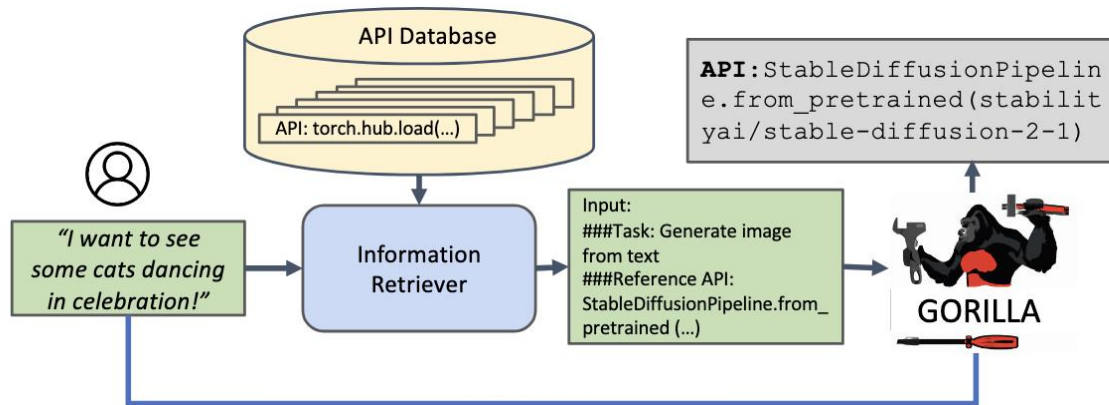
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



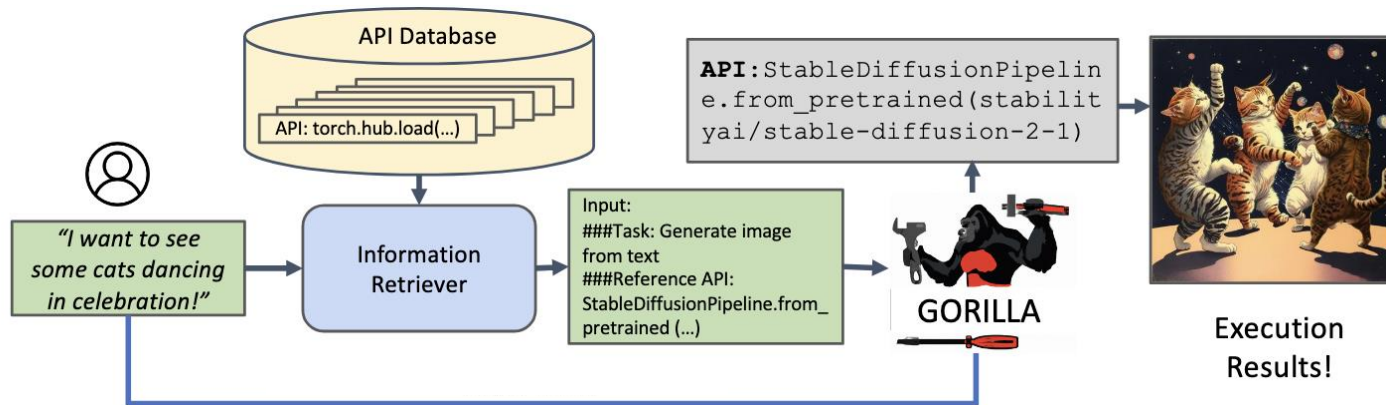
Gorilla: Large Language Model Connected with Massive APIs



*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs



First to use retriever augmented training for APIs

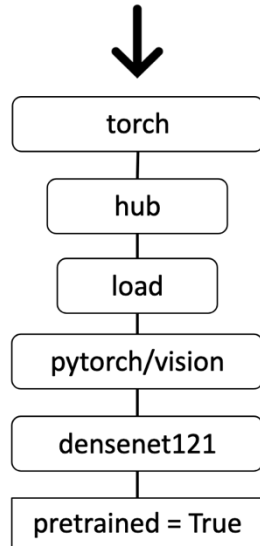
- Leads to better generalization
- Can adapt to change in API specs

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023

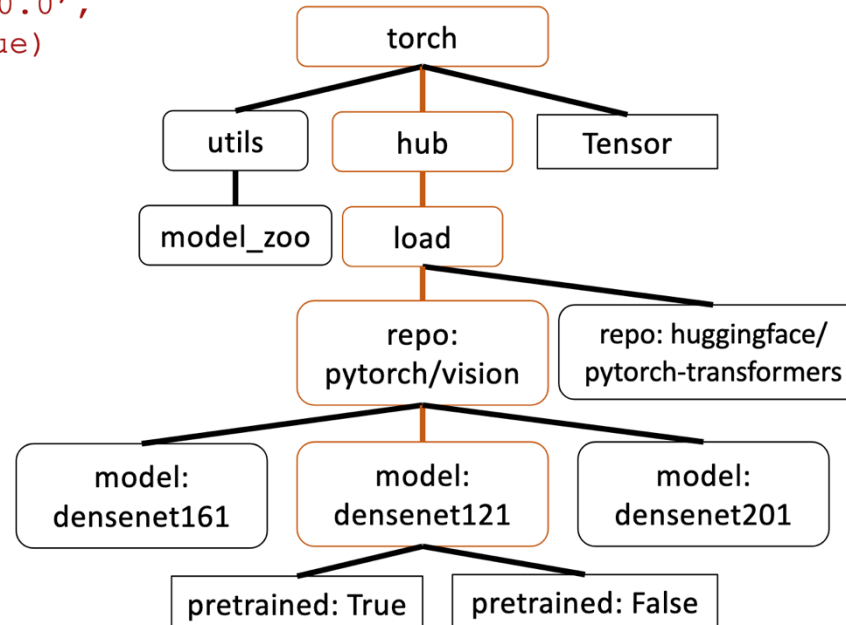


Gorilla: Large Language Model Connected with Massive APIs

```
torch.hub.load('pytorch/vision:v0.10.0',  
              'densenet121', pretrained=True)
```



∈



AST Sub-Tree Matching For **Evaluation**

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs

GPT-4	Claude	Gorilla
<pre><domain>: Speech-to-Text <api_provider>: TorchHub <code>: asr_model = torch.hub.load('snakers4/silero-models', 'asr', source='local') result = asr_model.transcribe(audio_path)</pre>	<pre><domain>: Audio-Translation <api_provider>: Pytorch <code>: import torchaudio translation = Torchaudio.pipelines. WAV2VEC2_ASR_PIPELINE("audio.wav")</pre>	<pre><domain>: Speech-to-Text <api_provider>: TorchHub <code>: asr_model = torch.hub.load('snakers4/silero-models', 'silero_sst') result = asr_model.transcribe(audio_path)</pre>
 Hallucinate!	 Wrong library!	 Good to go!

Prompt: Help me find an API to convert the spoken language in a recorded audio to text using Torch Hub.

Error Types in Function Calling

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs

LLM (retriever)	TorchHub			HuggingFace			TensorFlow Hub		
	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓
LLAMA (0-shot)	0	100	0	0.00	97.57	2.43	0	100	0
GPT-3.5 (0-shot)	48.38	18.81	32.79	16.81	35.73	47.46	41.75	47.88	10.36
GPT-4 (0-shot)	38.70	36.55	24.7	19.80	37.16	43.03	18.20	78.65	3.13
Claude (0-shot)	18.81	65.59	15.59	6.19	77.65	16.15	9.19	88.46	2.33
Gorilla (0-shot)	59.13	6.98	33.87	71.68	10.95	17.36	83.79	5.40	10.80

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs

LLM (retriever)	TorchHub			HuggingFace			TensorFlow Hub		
	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓
LLAMA (0-shot)	0	100	0	0.00	97.57	2.43	0	100	0
GPT-3.5 (0-shot)	48.38	18.81	32.79	16.81	35.73	47.46	41.75	47.88	10.36
GPT-4 (0-shot)	38.70	36.55	24.7	19.80	37.16	43.03	18.20	78.65	3.13
Claude (0-shot)	18.81	65.59	15.59	6.19	77.65	16.15	9.19	88.46	2.33
Gorilla (0-shot)	59.13	6.98	33.87	71.68	10.95	17.36	83.79	5.40	10.80
LLAMA (GPT-Index)	14.51	75.8	9.67	10.18	75.66	14.20	15.62	77.66	6.71
GPT-3.5 (GPT-Index)	60.21	1.61	38.17	29.08	7.85	44.80	65.59	3.79	30.50
GPT-4 (GPT-Index)	59.13	1.07	39.78	44.58	11.18	44.25	43.94	31.53	24.52
Claude (GPT-Index)	60.21	3.76	36.02	41.37	18.81	39.82	55.62	16.20	28.17
Gorilla (GPT-Index)	61.82	0	38.17	47.46	8.19	44.36	64.96	2.33	32.70

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023



Gorilla: Large Language Model Connected with Massive APIs

LLM (retriever)	TorchHub			HuggingFace			TensorFlow Hub		
	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓
LLAMA (0-shot)	0	100	0	0.00	97.57	2.43	0	100	0
GPT-3.5 (0-shot)	48.38	18.81	32.79	16.81	35.73	47.46	41.75	47.88	10.36
GPT-4 (0-shot)	38.70	36.55	24.7	19.80	37.16	43.03	18.20	78.65	3.13
Claude (0-shot)	18.81	65.59	15.59	6.19	77.65	16.15	9.19	88.46	2.33
Gorilla (0-shot)	59.13	6.98	33.87	71.68	10.95	17.36	83.79	5.40	10.80
LLAMA (GPT-Index)	14.51	75.8	9.67	10.18	75.66	14.20	15.62	77.66	6.71
GPT-3.5 (GPT-Index)	60.21	1.61	38.17	29.08	7.85	44.80	65.59	3.79	30.50
GPT-4 (GPT-Index)	59.13	1.07	39.78	44.58	11.18	44.25	43.94	31.53	24.52
Claude (GPT-Index)	60.21	3.76	36.02	41.37	18.81	39.82	55.62	16.20	28.17
Gorilla (GPT-Index)	61.82	0	38.17	47.46	8.19	44.36	64.96	2.33	32.70
LLAMA (Oracle)	16.12	79.03	4.83	17.70	77.10	5.20	12.55	87.00	0.43
GPT-3.5 (Oracle)	66.31	1.60	32.08	89.71	6.64	3.65	95.03	0.29	4.67
GPT-4 (Oracle)	66.12	0.53	33.33	85.07	10.62	4.31	55.91	37.95	6.13
Claude (Oracle)	63.44	3.76	32.79	77.21	19.58	3.21	74.74	21.60	3.64
Gorilla (Oracle)	67.20	0	32.79	91.26	7.08	1.66	94.16	1.89	3.94

*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023





Berkeley Function-Calling Leaderboard

BFCL: From Tool Use to Agentic Evaluation of Large Language Models

The Berkeley Function Calling Leaderboard (BFCL) V4 evaluates the LLM's ability to call functions (aka tools) accurately. This leaderboard consists of real-world data and will be updated periodically. For more information on the evaluation dataset and methodology, please refer to our blogs: [BFCL-v1](#) introducing AST as an evaluation metric, [BFCL-v2](#) introducing enterprise and OSS-contributed functions, [BFCL-v3](#) introducing multi-turn interactions, and [BFCL-v4](#) introducing holistic agentic evaluation. Checkout [code](#) and [data](#).

Last Updated: 2025-08-26 [\[Change Log\]](#)

Search

				Agentic		Multi Turn	Single Turn		Hallucination Measurement		Format Sensitivity			
				Web Search ▶	Memory ▶	Multi turn ▶	Non-live (AST) ▶	Live (AST) ▶					Latency (s) ▶	
Rank 📊	Overall Acc	Model	Cost (\$)	Overall Acc	Overall Acc	Overall Acc	Overall Acc	Overall Acc	Relevance	Irrelevance	Max Delta	SD	Mean	Organization
1	70.85	GLM-4.5 (FC)	2.9	79	50.75	65.62	86.6	81.72	75	83.79	N/A	N/A	2.73	Zhipu AI
2	70.36	Claude-Opus-4-1-20250805 (FC)	207.12	77	62.15	57.88	88.38	81.5	62.5	81.82	N/A	N/A	4.33	Anthropic
3	70.29	Claude-Sonnet-4-20250514 (FC)	41.49	84	59.35	54.75	88.38	81.05	62.5	82.47	N/A	N/A	4.08	Anthropic
4	67.87	GLM-4.5-Air (FC)	4.22	73.5	47.53	62.5	87.15	79.42	81.25	82.58	N/A	N/A	3.89	Zhipu AI
5	61.6	Grok-4-0709 (Prompt)	333.24	72	54.41	43.25	81.27	69.73	75	82.41	9.0	1.99	19.23	xAI
6	61.01	Grok-4-0709 (FC)	329.44	72.5	65.38	36.12	85.21	74.39	87.5	66.39	N/A	N/A	10.78	xAI
7	59.22	GPT-5-2025-08-07 (FC)	159.16	84.5	57.63	28.5	72.92	58.25	62.5	91.31	N/A	N/A	10.85	OpenAI
8	58.76	o3-2025-04-16 (Preview)	235.89	43.5	46.45	56.12	81.42	73.43	87.5	84.45	5.0	1.22	5.68	OpenAI

image credits: screenshot of <https://gorilla.cs.berkeley.edu/leaderboard.html>



Summary

1. APIBench (Gorilla)
 - Low diversity in APIs
 - Single turn dialogs



ToolAlpaca*

1. Synthesized a dataset 3000 examples
 - Using 400 real-world inspired tools spanning 50 distinct categories
2. Finetuned Vicuna-7B (and 13B) model to create ToolAlpaca-7B (and 13B)

*ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases, Tang et. al., Sep 2023



ToolAlpaca*



Public Holidays

Introduction: Data on national, regional, and religious holidays via API

- 1400 tool names and introductions
- collected from github.com/public-apis/public-apis

*ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases, Tang et. al., Sep 2023



ToolAlpaca*



Public Holidays

Introduction: Data on national, regional, and religious holidays via API

Description: The Public Holidays API is a user-friendly interface that provides comprehensive information on national, regional, and religious holidays around the world.

The API's key features are:

- 1) Get a list of holidays for a particular country with dates, descriptions, and types.
- 2) Retrieve detailed information on a specific holiday, including its history, purpose, and traditions.
- 3) Obtain information on public holidays for a specific year, month, or day.

Function Documentation:

getHolidays: Get a list of holidays for a particular country with dates, descriptions, and types.

Parameters: {"country": "Required. String. The country for which holidays are to be retrieved.", "year": "Integer. The year for which holidays are to be retrieved."}

Output: A list of holidays with their dates, descriptions, and types for the specified country, year, month, and day.

searchHoliday: Search for holidays based on keywords, country, and date range.

getHolidayDetails: Retrieve detailed information on a specific holiday, including its history, purpose, and traditions.

OpenAPI Specification:

- [GET /holidays/{country}](#) - [GET /holidays/{holidayId}/details](#) - [GET /holidays/search](#)

*ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases, Tang et. al., Sep 2023

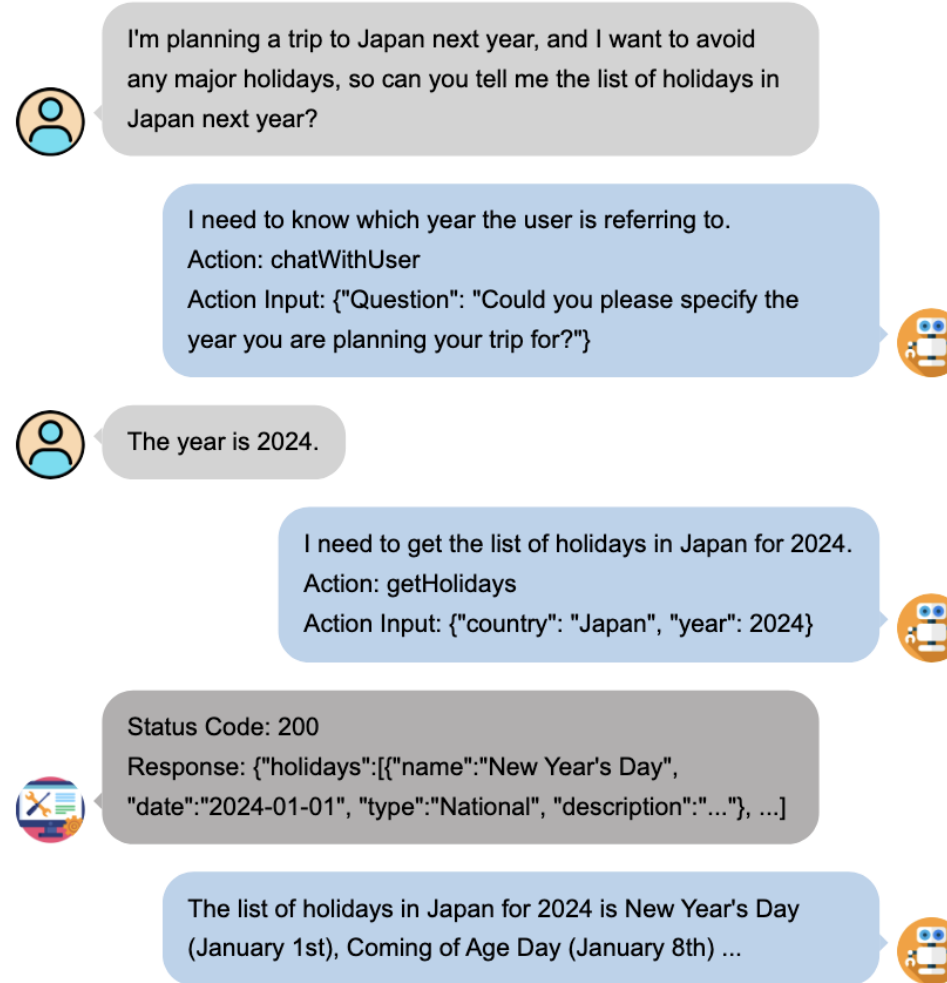


ToolAlpaca*

Public Holidays

getHolidays: Get a list of holidays for a particular country with dates, descriptions, and types.
Parameters: {"country": "Required. String. The country for which holidays are to be retrieved.", "year": "Integer. The year for which holidays are to be retrieved."}
Output: A list of holidays with their dates, descriptions, and types for the specified country, year, month, and day.
searchHoliday: Search for holidays based on keywords, country, and date range.
getHolidayDetails: Retrieve detailed information on a specific holiday, including its history, purpose, and traditions.

- Given a tool, uses multi agent setup to generate a dialog
- Agents involved:
 - User agent
 - Assistant Agent
 - Tool Executor Agent



*[ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases, Tang et. al., Sep 2023](#)



ToolAlpaca*

Model	Simulated Tools				Real-world APIs		
	Procedure	Response	Overall	Human	Procedure	Response	Overall
GPT-3.5	77.0	85.0	75.0	79.0	75.4	80.7	72.8
Vicuna-7B	19.0	21.0	17.0	16.0	7.9	11.4	7.9
ToolAlpaca-7B	63.0	69.0	60.0	73.0	63.2	57.9	55.3
Vicuna-13B	17.0	31.0	16.0	25.0	13.2	16.7	12.3
ToolAlpaca-13B	70.0	73.0	70.0	75.0	66.7	67.5	61.4

Evaluation results on unseen simulated tools and real-world APIs

1. **Procedure:** selecting suitable actions, and utilizing correct parameters
2. **Response:** does the final response satisfy the user's instruction
3. **Overall:** both procedure and response

*ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases, Tang et. al., Sep 2023



Summary

1. APIBench (Gorilla)
 - Low diversity in APIs
 - Single turn dialogs grounded on single tool
2. ToolAlpaca Dataset
 - Better Diversity (400 real world inspired APIs)
 - Multi turn dialogs grounded on single tool



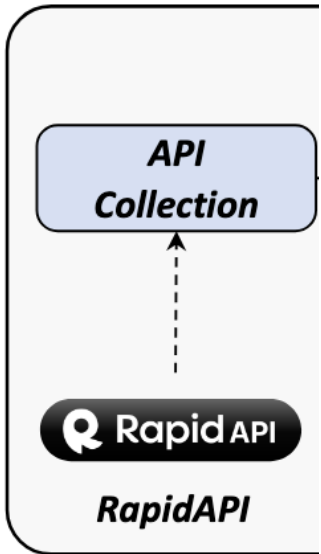
ToolLLM*

1. Synthesized a dataset named ToolBench
 - Scraped 16,464 real-world REST APIs from RapidAPI Hub
2. Finetuned LLaMA-7B model with ToolBench to create the ToolLLaMA model

*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



ToolLLM*



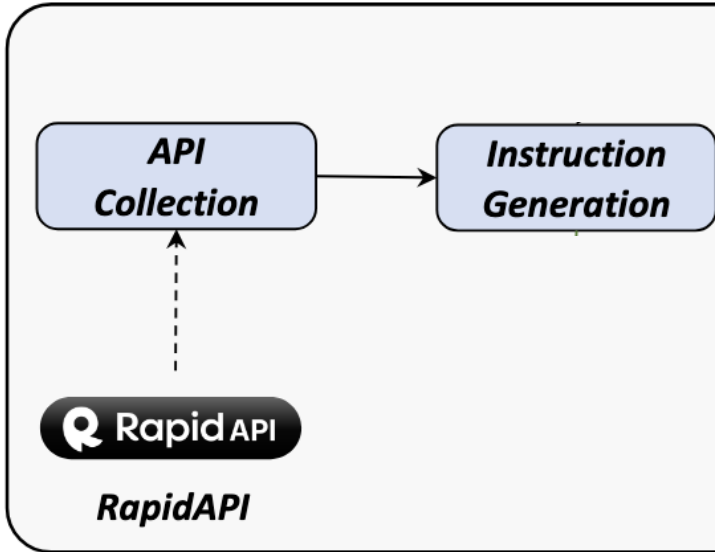
API Collection:

- gathered 53,190 APIs from RapidAPI
- retained 16,464 APIs that are functional

*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



ToolLLM*



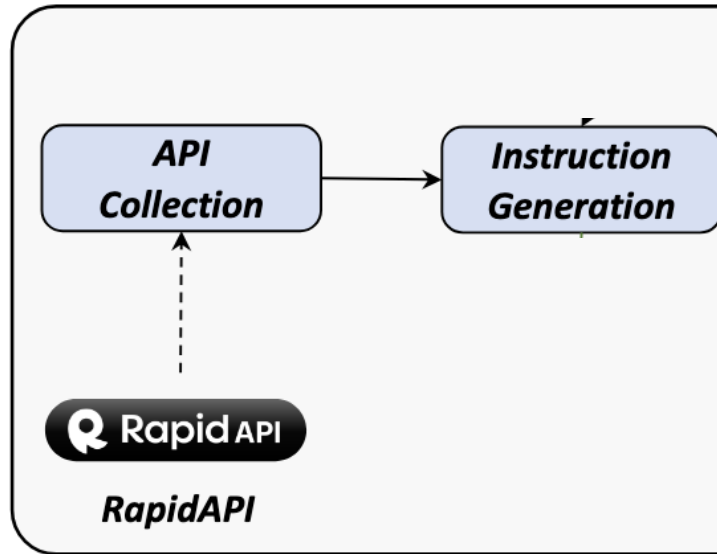
Instruction Generation:

- Diversity -> 3451 tools
- Multiple Tools in a dialog
 - Select a subset of N tools from the collection
 - Synthesize a user query using the N tools
- Generates ~200,000 instructions

*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



ToolLLM*



Tools: ['Entrepreneur Mindset Collection', 'Random Words', 'thedigitalnewsfeederapi', 'Chemical Elements']



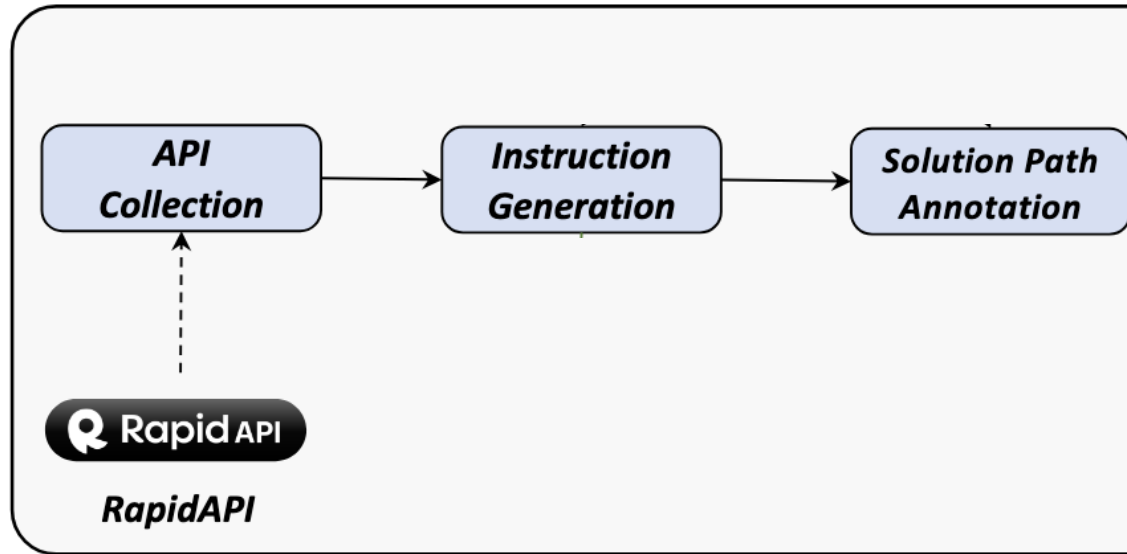
User Query: For my best friend's surprise birthday party, I require inspiration for party games and decorations. Kindly suggest some random words that can serve as themes for the party. Furthermore, I'm interested in gathering news articles about the latest party trends to ensure a modern celebration

Related APIs: ['Random Words', 'thedigitalnewsfeederapi']

*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



ToolLLM*



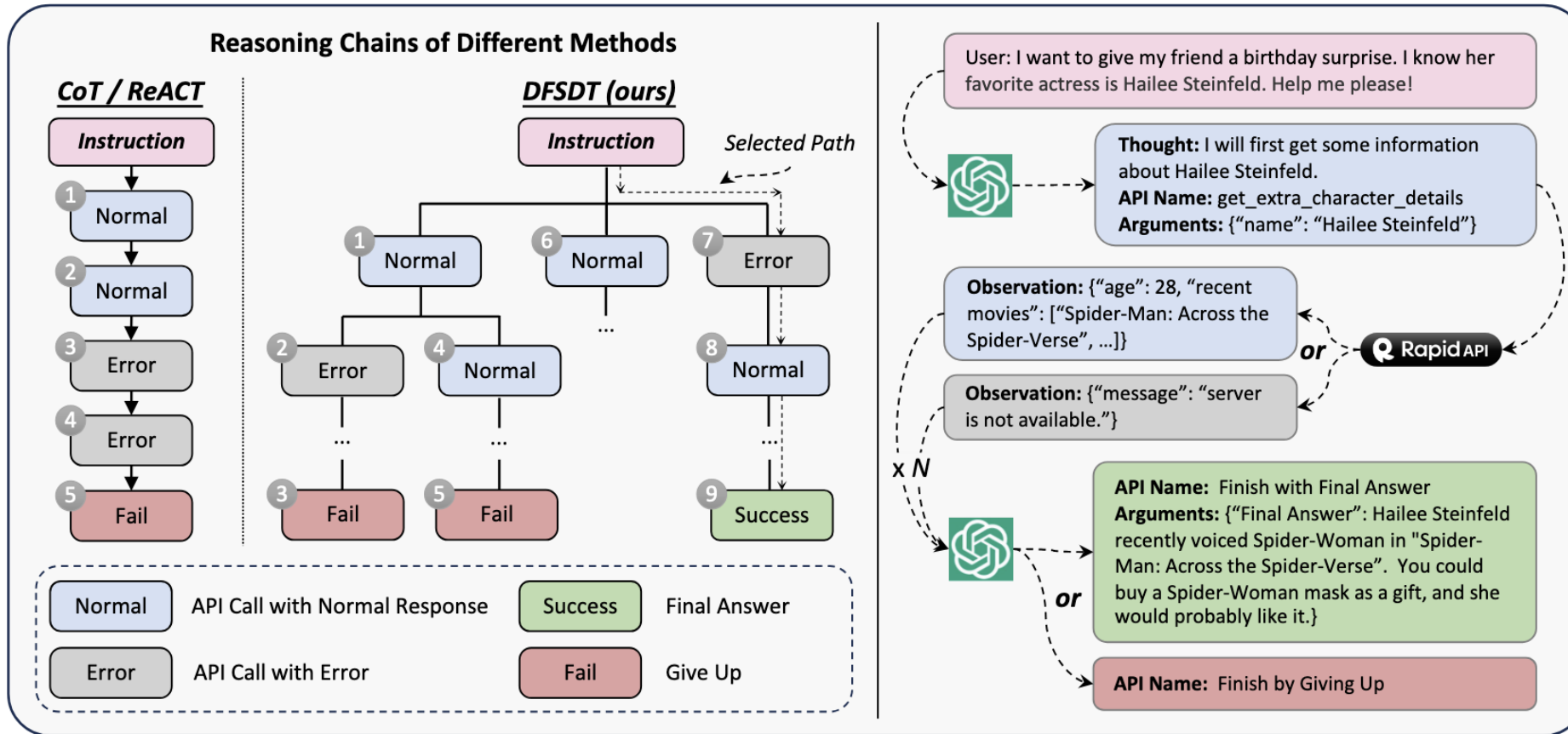
Solution Path Annotation

- Generate dialogs starting from user query and using relevant APIs
- Uses a novel depth-first search-based decision tree (DFSDT)
- generates 126,486 (instruction, solution path) pairs

*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



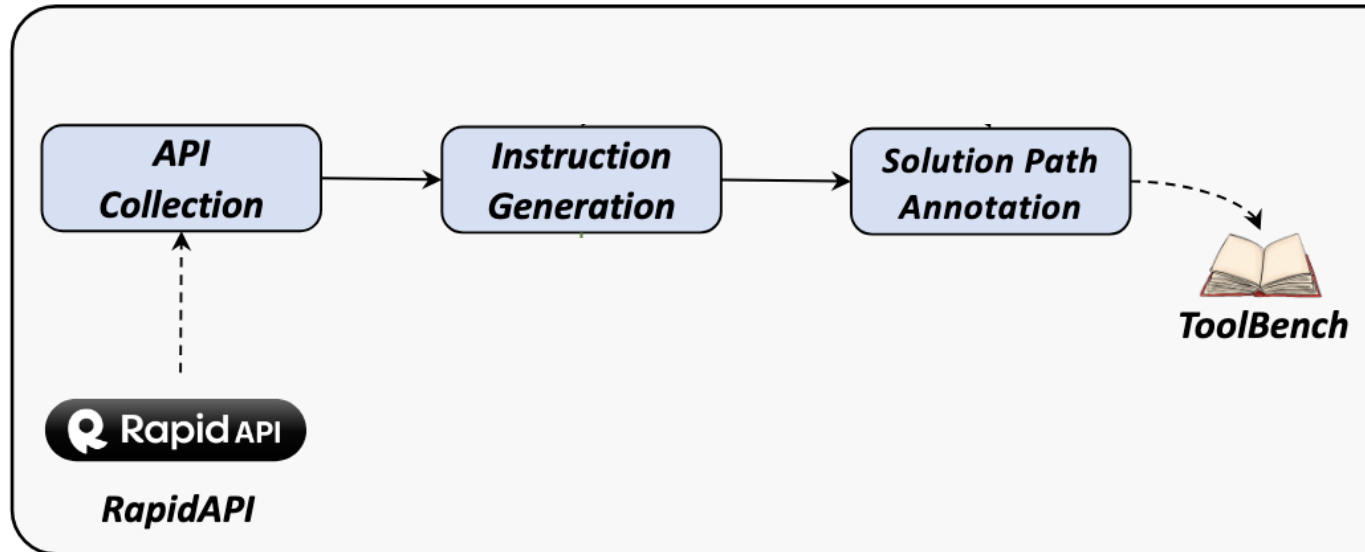
ToolLLM*



*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



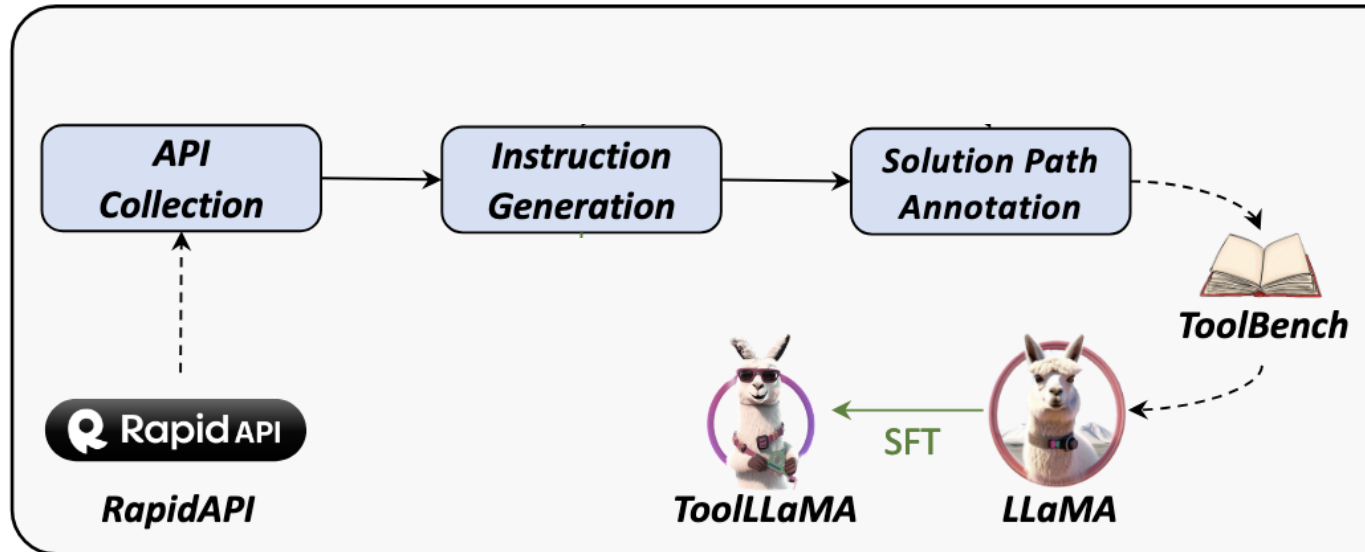
ToolLLM*



*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



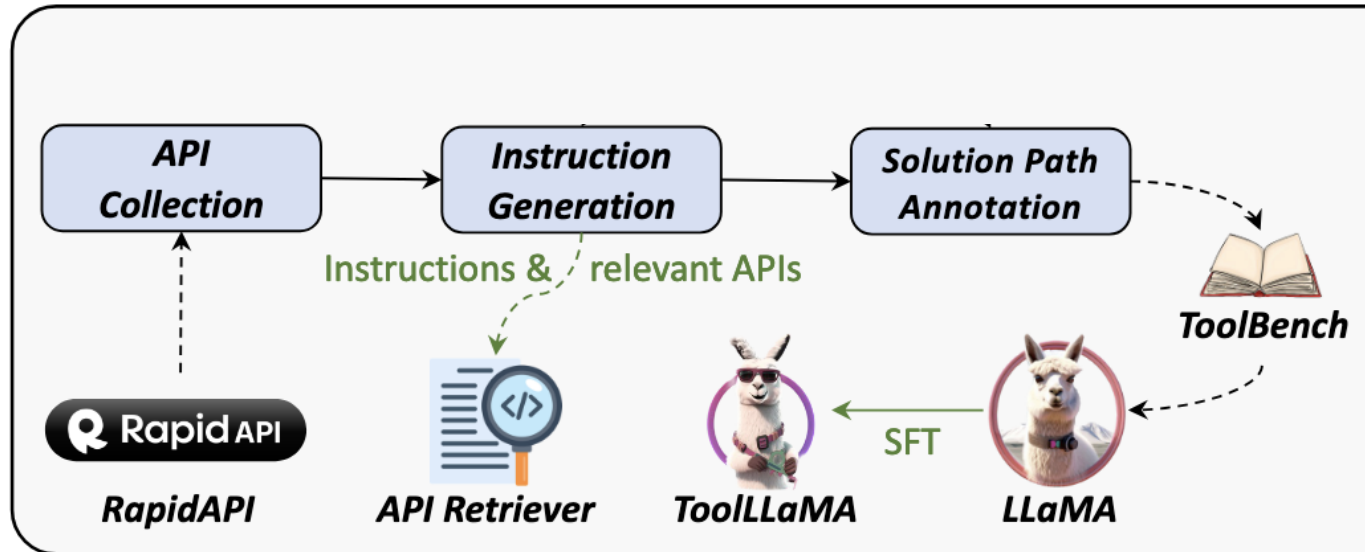
ToolLLM*



*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



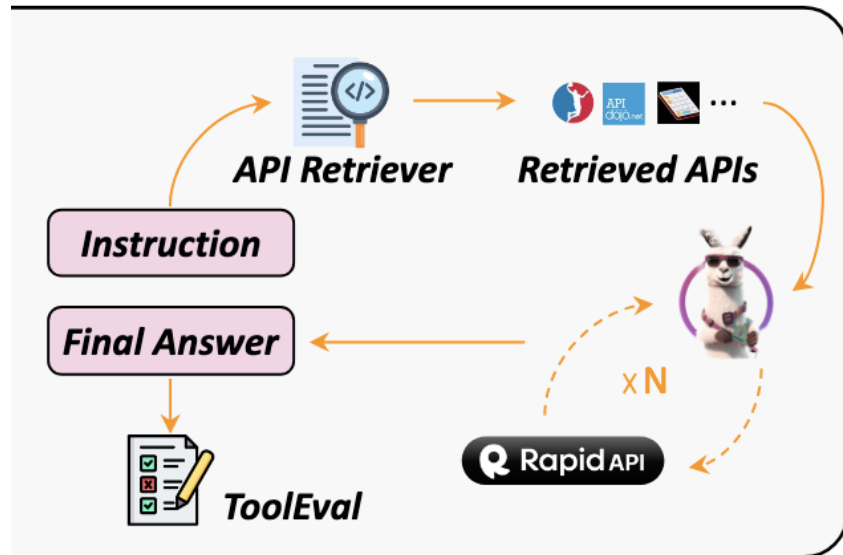
ToolLLM*



*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



ToolLLM*



*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



ToolLLM*

Method	HuggingFace		TorchHub		TensorHub	
	Hallu. (↓)	AST (↑)	Hallu. (↓)	AST (↑)	Hallu. (↓)	AST (↑)
ToolLLaMA + Our Retriever	<u>10.60</u>	16.77	<u>15.70</u>	51.16	<u>6.48</u>	<u>40.59</u>
Gorilla-ZS + BM25	46.90	10.51	17.20	44.62	20.58	34.31
Gorilla-RS + BM25	6.42	<u>15.71</u>	5.91	<u>50.00</u>	2.77	41.90
ToolLLaMA + Oracle	<u>8.66</u>	<u>88.80</u>	<u>14.12</u>	<u>85.88</u>	<u>7.44</u>	<u>88.62</u>
Gorilla-ZS + Oracle	52.88	44.36	39.25	59.14	12.99	83.21
Gorilla-RS + Oracle	6.97	89.27	6.99	93.01	2.04	94.16

OOD generalization experiments on APIBench

*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023



Summary

1. APIBench (Gorilla)
 - Low diversity in APIs
 - Single turn dialogs grounded on single tool
2. ToolAlpaca Dataset
 - Moderate Diversity (400 real world inspired APIs)
 - Multi turn dialogs grounded on single tool
3. ToolBench (ToolLLM)
 - High diversity (16K real world APIs from RapidAPI Hub)
 - Multi turn dialogs grounded on multiple tools



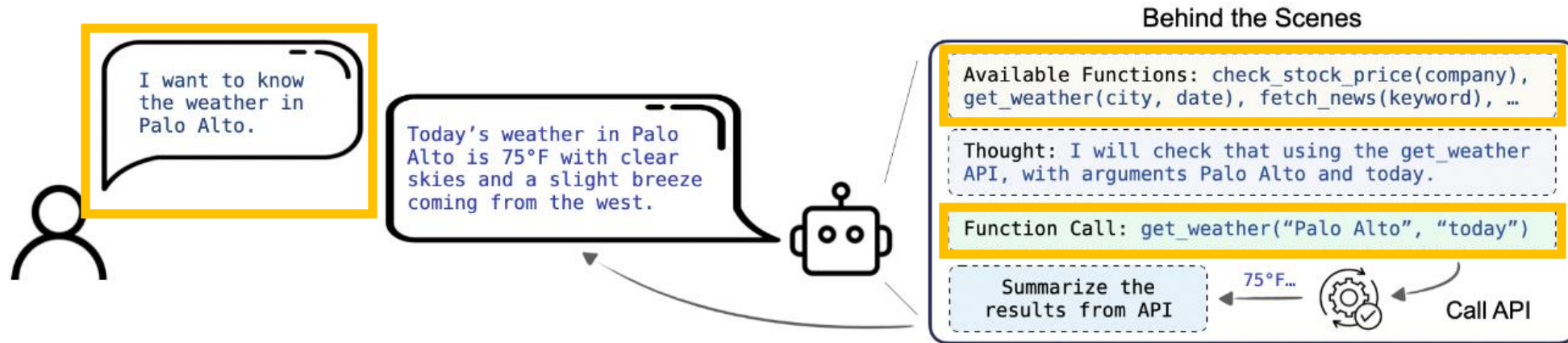
APIGen*

1. Synthesized a dataset named `xlam-function-calling-60k`
 - Used 3673 APIs from RapidAPI Hub (scraped for ToolBench)
2. Finetuned DeepSeek-Coder-1.3B-instruct (and 7B) model with `xlam-function-calling-60k` to create the `s_xLAM-1B (FC)` (and 7B) model

* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



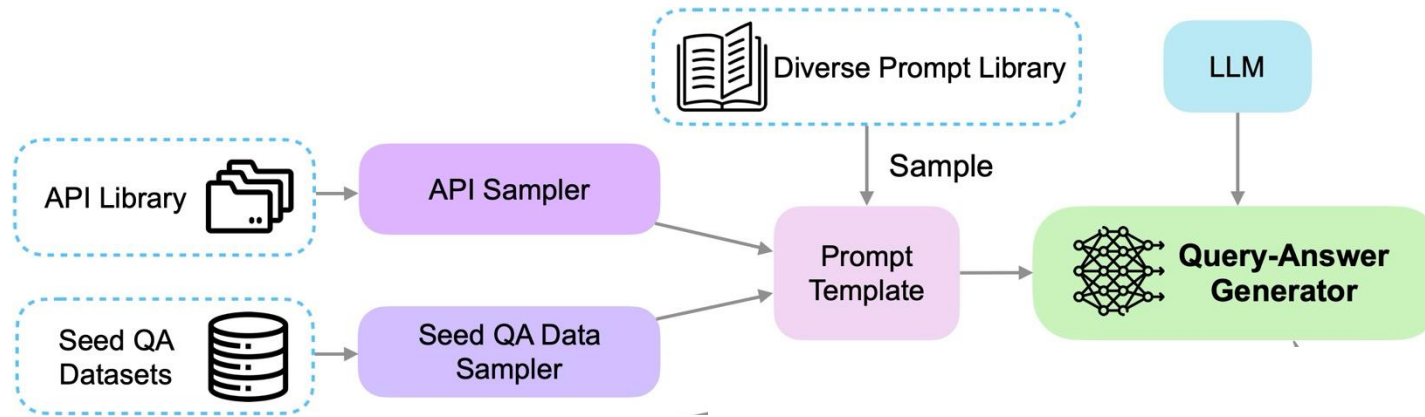
APIGen*



* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



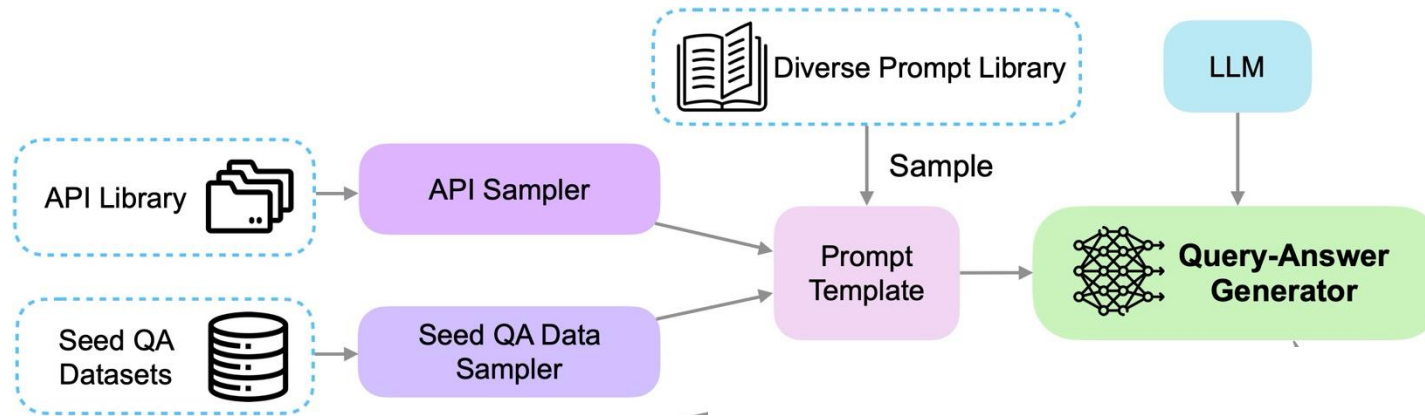
APIGen*



* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



APIGen*



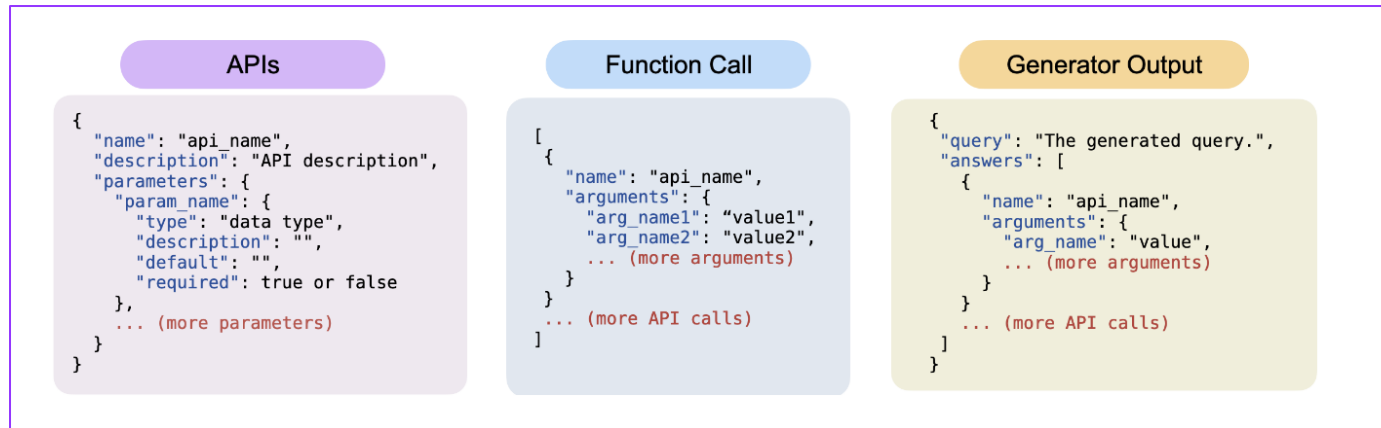
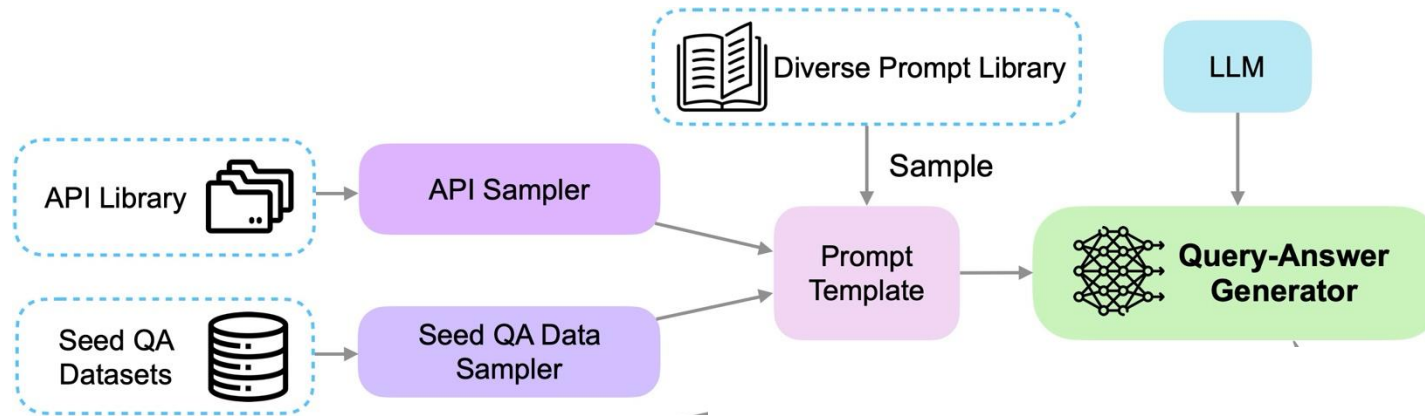
Types of Diversity:

1. Query Style Diversity - Single, Multiple, Parallel, and Parallel-Multiple
2. Question verbosity
3. Ambiguous or misspelled user request

* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



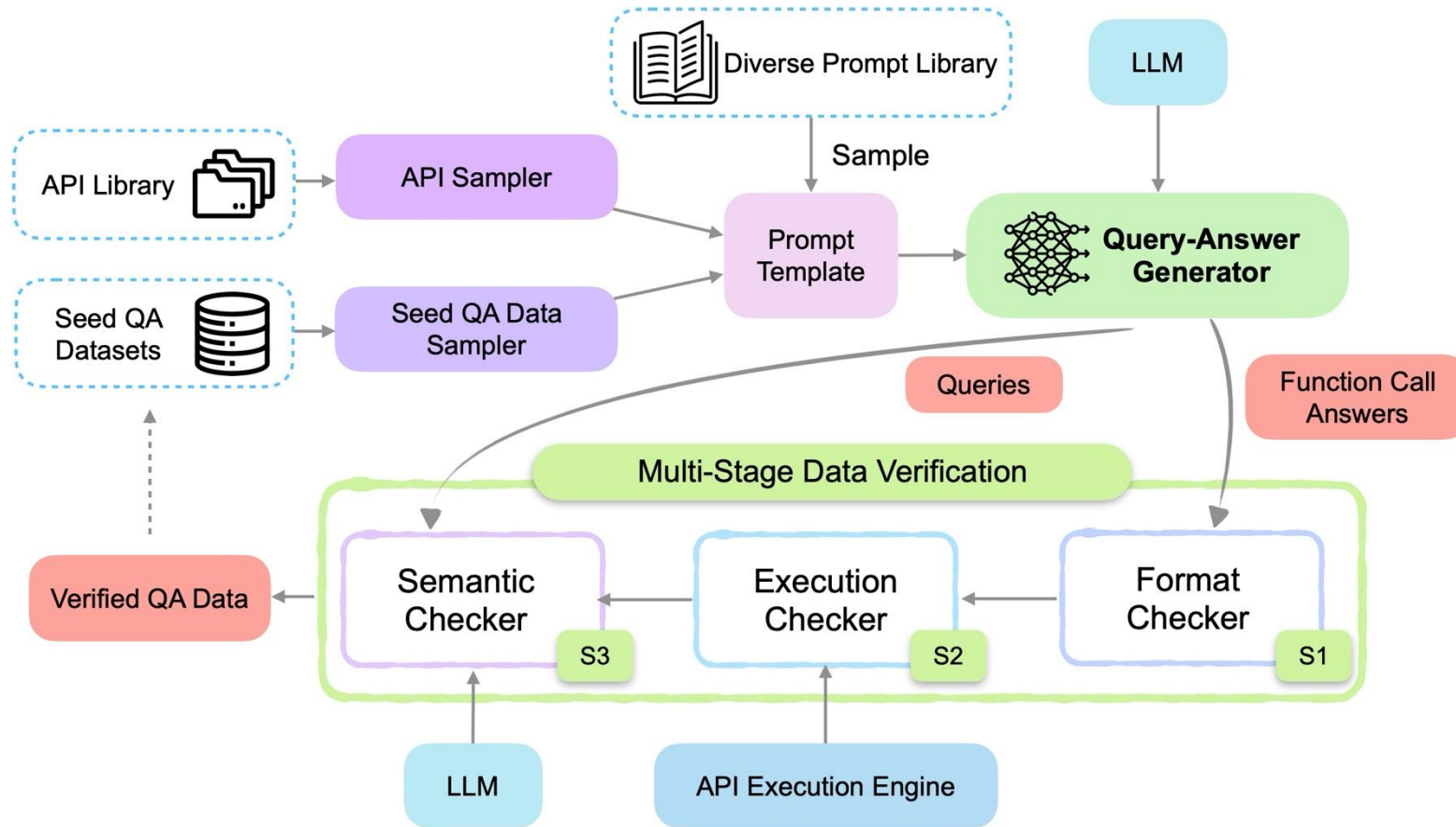
APIGen*



* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



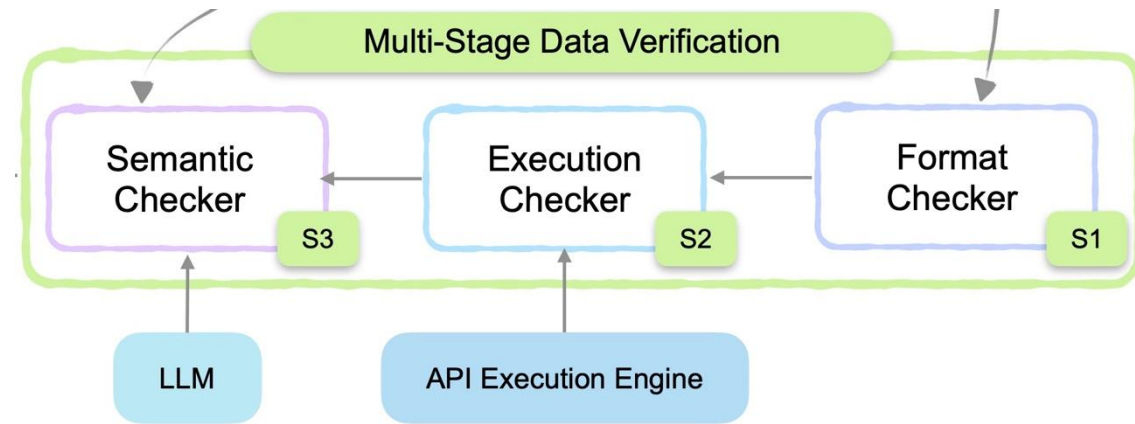
APIGen*



* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



APIGen*



APIs

```
{
  "name": "api_name",
  "description": "API description",
  "parameters": {
    "param_name": {
      "type": "data type",
      "description": "",
      "default": "",
      "required": true or false
    },
    ... (more parameters)
  }
}
```

Function Call

```
[
  {
    "name": "api_name",
    "arguments": {
      "arg_name1": "value1",
      "arg_name2": "value2",
      ... (more arguments)
    }
  },
  ... (more API calls)
]
```

Generator Output

```
{
  "query": "The generated query.",
  "answers": [
    {
      "name": "api_name",
      "arguments": {
        "arg_name": "value",
        ... (more arguments)
      }
    },
    ... (more API calls)
  ]
}
```

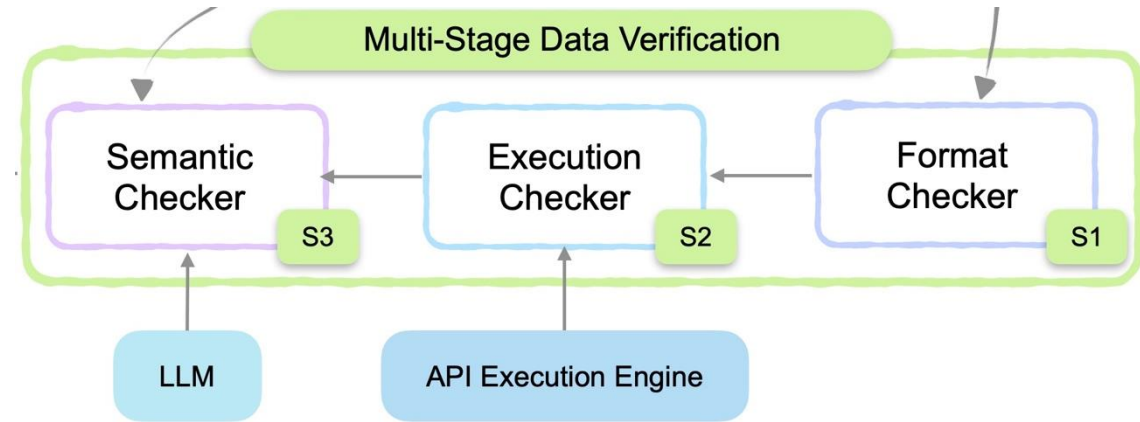
* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



APIGen*

Execution Checker

1. Functions are executed using the appropriate backend
 - Python functions are directly imported and executed in a separate subprocess
 - REST APIs are called to obtain results and status codes).
2. Unsuccessful executions are filtered out



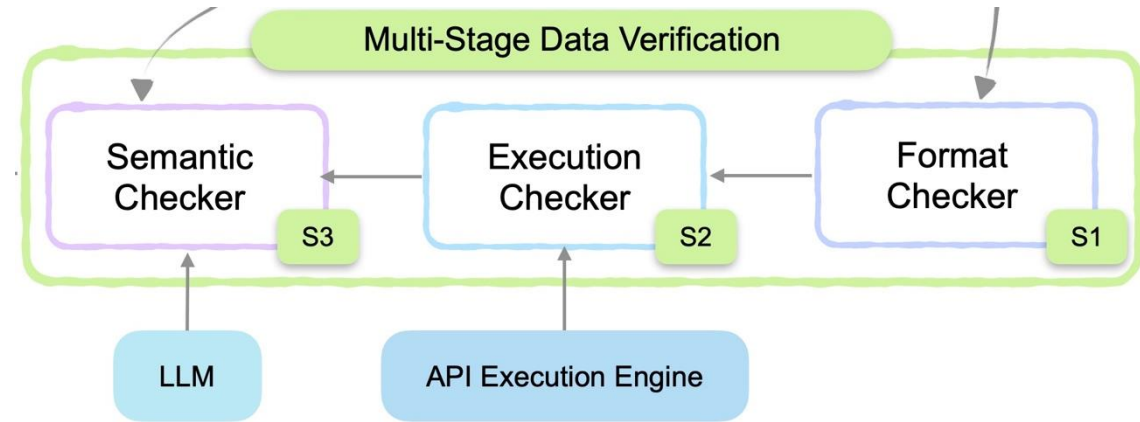
* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



APIGen*

Semantic Checker

- Does the final answer semantically align with the query's objective?
- Query-answer pairs that execute successfully can produce meaningless results due to
 1. infeasible queries
 2. incorrect arguments



* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



APIGen*

Model	Verified Data	Fail Format	Fail Execution	Fail Semantic	Pass Rate
DeepSeek-Coder-33B-Inst	13,769	4,311	15,496	6,424	34.42%
Mixtral-8x7B-Inst	15,385	3,311	12,341	7,963	38.46%
Mixtral-8x22B-Inst	26,384	1,680	5,073	6,863	65.96%
DeepSeek-V2-Chat (236B)	33,659	817	3,359	2,165	84.15%

Filtering statistics for the generated datasets using different base LLMs.

* APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024



BFCL Leaderboard

The Berkeley Function Calling Leaderboard V3 (also called Berkeley Tool Calling Leaderboard V3) evaluates the LLM's ability to call functions (aka tools) accurately. This leaderboard consists of real-world data and will be updated periodically. For more information on the evaluation dataset and methodology, please refer to our blogs: [BFCL-v1](#) introducing AST as an evaluation metric, [BFCL-v2](#) introducing enterprise and OSS-contributed functions, and [BFCL-v3](#) introducing multi-turn interactions. Checkout [code and data](#).

Last Updated: 2024-09-20 [\[Change Log\]](#)

					Single Turn			Multi Turn	
				Latency (s) ▶	Non-live (AST) ▶	Non-live (Exec) ▶	Live (AST) ▶	Multi turn ▶	
Rank ▲	Overall Acc	Model	Cost (\$)	Mean	AST Summary	Exec Summary	Overall Acc	Overall Acc	
1	59.49	GPT-4-turbo-2024-04-09 (FC)	34.98	2.87	82.65	83.8	73.39	21.62	
2	59.29	GPT-4o-2024-08-06 (FC)	8.37	1.33	85.52	82.96	71.79	21.25	
3	59.13	xLAM-8x22b-r (FC)	N/A	2.64	89.75	89.32	72.81	15.62	
4	58.45	GPT-4o-mini-2024-07-18 (FC)	0.56	1.67	82.83	81.8	67.53	25.75	
5	57.94	xLAM-8x7b-r (FC)	N/A	1.21	88.44	85.89	71.97	15.75	

image credits: screenshot of <https://gorilla.cs.berkeley.edu/leaderboard.html>



Summary

1. APIBench (Gorilla)
 - Low diversity in APIs
 - Single turn dialogs grounded on single tool
2. ToolAlpaca Dataset
 - Moderate Diversity (400 real-world inspired APIs)
 - Multi turn dialogs grounded on single tool
3. ToolBench (ToolLLM)
 - High diversity (16K real-world APIs from RapidAPI Hub)
 - Multi turn dialogs grounded on multiple tools
4. xlam-function-calling-60k (APIGen)
 - High diversity (3K high-quality real-world APIs from RapidAPI Hub)
 - Single turn dialogs grounded on single tool
 - Focus on diverse queries



Granite-Function-Calling-Model*



Issues with existing open models:

1. **Openness:** The best performing models are proprietary and the ones that have open licenses (e.g., Gorilla) are trained using data generated from OpenAI models that have restrictive usage license.
2. **Generalizability:** Even though the datasets are generated using diverse sets of APIs (e.g., RapidAPIs), [Basu et al. \(2024\)](#) has shown that models trained on these datasets have difficulty generalizing to out-of-domain datasets
3. **Granular tasks:** Function calling encompasses multiple granular sub-tasks such as function-name detection, slot filling, and detecting the ordered sequence of functions needed to be called. Existing models trained to perform function calling lack the ability to handle these granular tasks independently

*Granite-Function Calling Model: Introducing Function Calling Abilities via Multi-task Learning of Granular Tasks, Abdelaziz et. Al., Jun 2024



Granite-Function-Calling-Model*



1. **Openness:** synthesize training data using models with provide open license
2. **Generalizability:** repurposed existing datasets with permissible license and added new synthesized datasets to get better generalization
3. **Granular tasks:** enabled support for granular tasks

*Granite-Function Calling Model: Introducing Function Calling Abilities via Multi-task Learning of Granular Tasks, Abdelaziz et. Al., Jun 2024



Summary

1. APIBench (Gorilla)
 - Low diversity in APIs
 - Single turn dialogs grounded on single tool
2. ToolAlpaca Dataset
 - Moderate Diversity (400 real-world inspired APIs)
 - Multi turn dialogs grounded on single tool
3. ToolBench (ToolLLM)
 - High diversity (16K real-world APIs from RapidAPI Hub)
 - Multi turn dialogs grounded on multiple tools
4. xlam-function-calling-60k (APIGen)
 - High diversity (3K high-quality real-world APIs from RapidAPI Hub)
 - Single turn dialogs grounded on single tool
 - Focus on diversity

5. Granite-Function-Calling-Model



- High diversity
- Multi turn dialogs grounded on multiple tools
- **Focus on open-sourced data and model with permissible license**

