LLMs and Tools
Part-2: Function Calling

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Dinesh Raghu
Senior Researcher, IBM Research

LLMs and Tools

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

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

Part 3: Automating Complex, Multi-step Tasks (Agentic Workflows)

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 for the supports

1. institute rules/policy related queries
2. searching/adding/dropping courses
3. query academic calendar
4. pay fees

How would you build such a chatbot?

What is Function Calling?

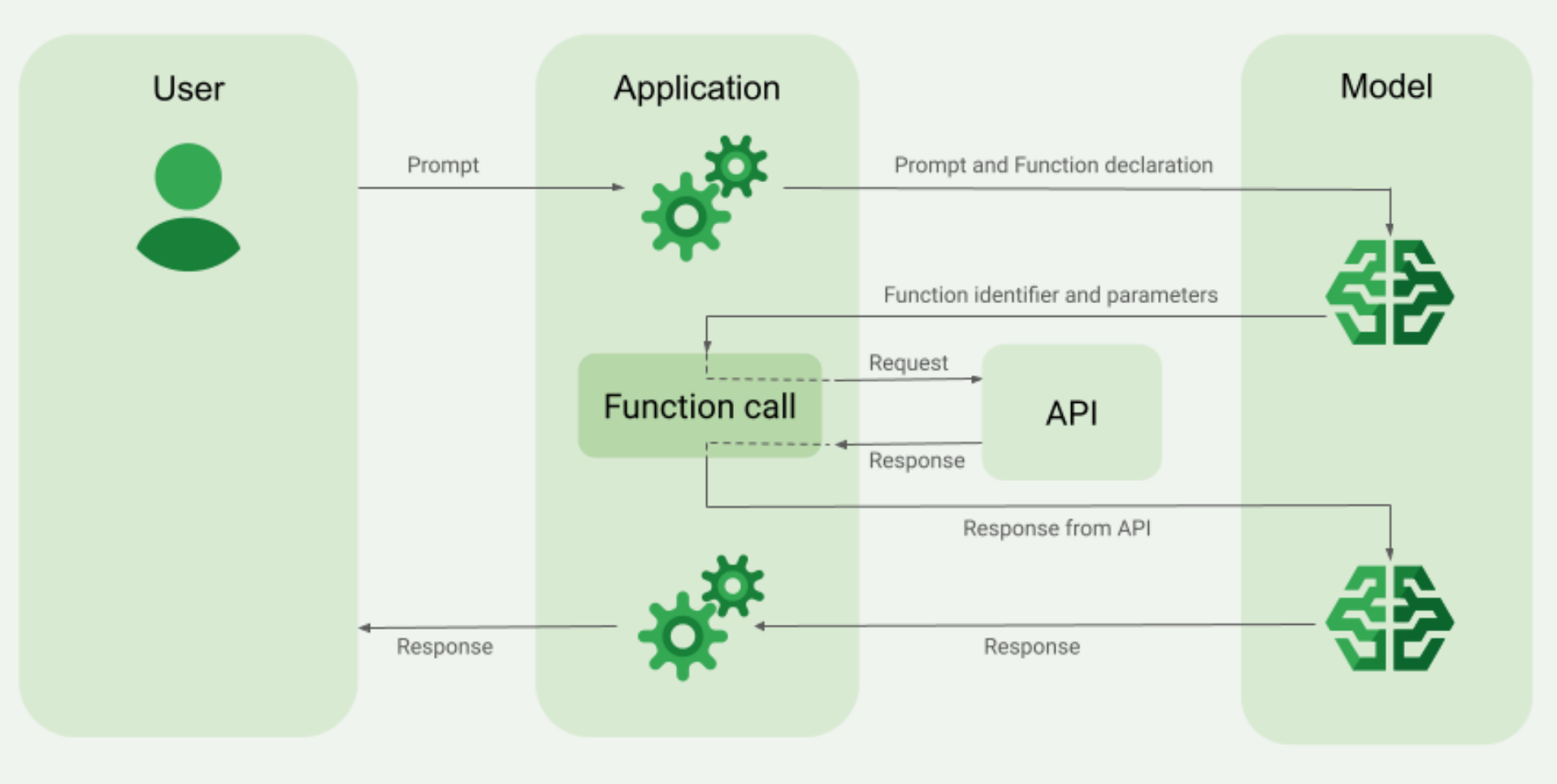
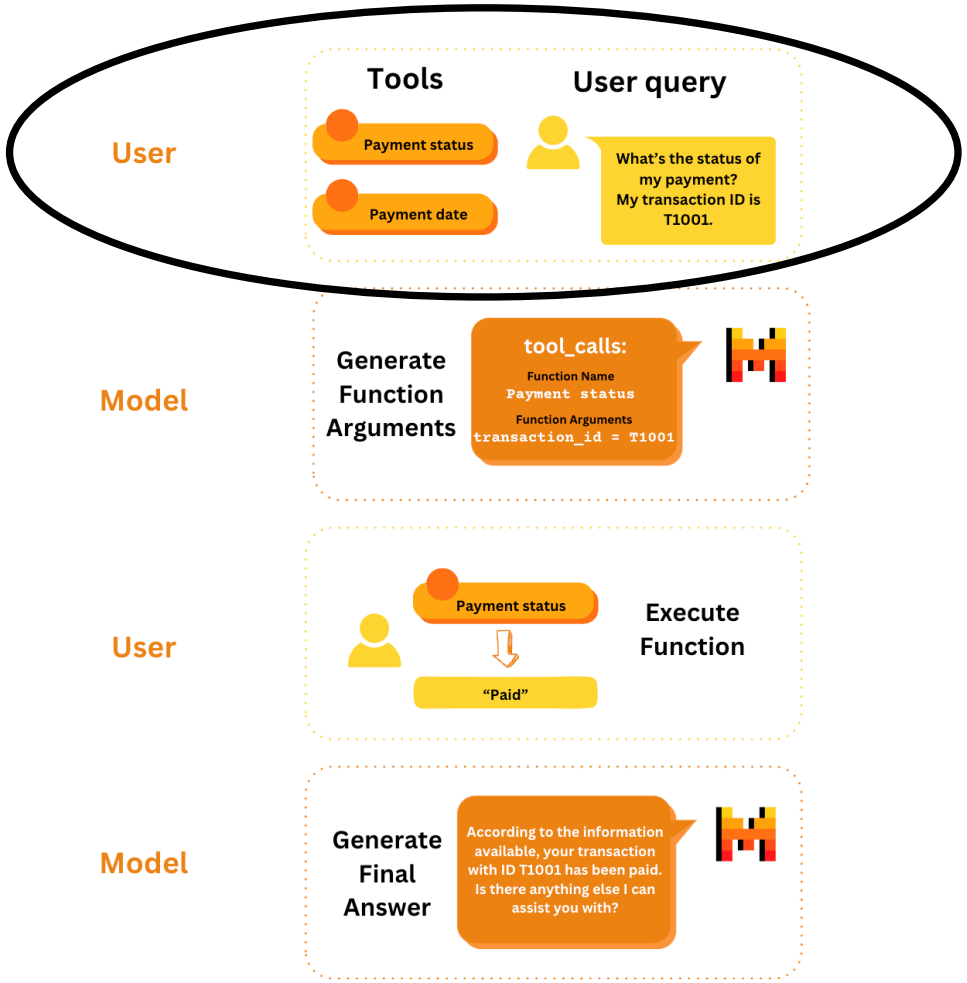


Image credit: https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling

What is Function Calling?



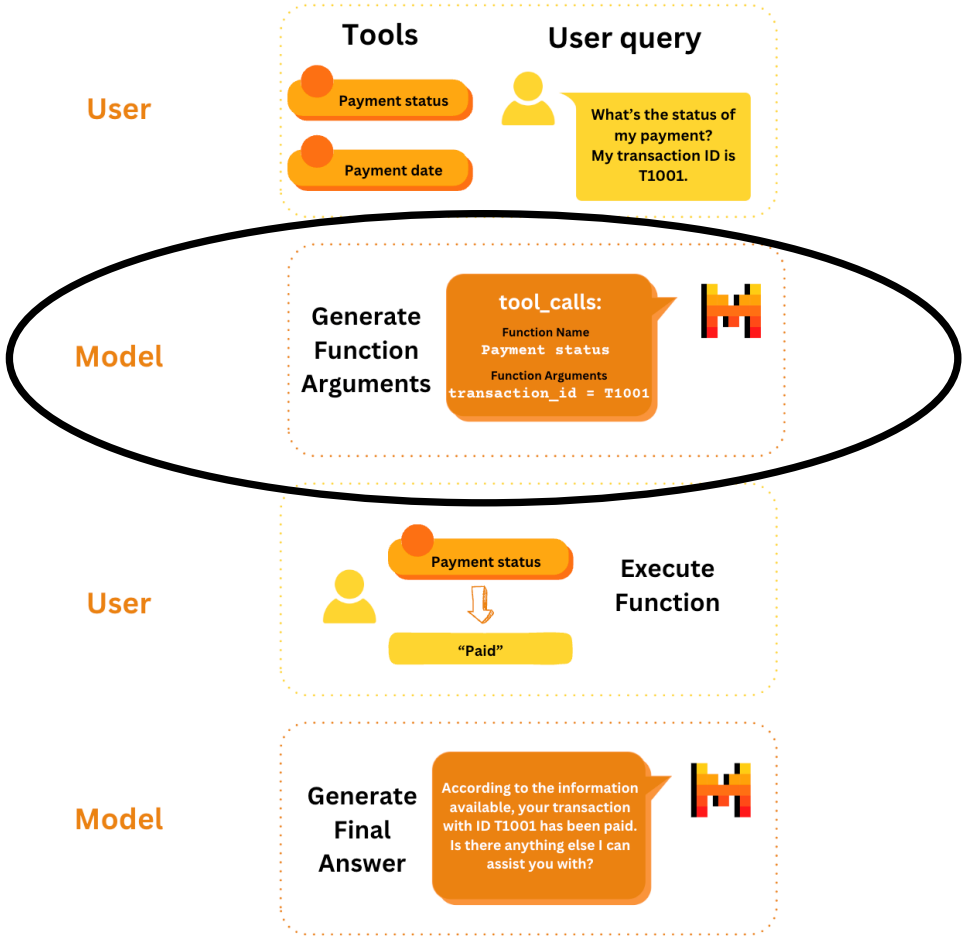
1. User specifies tools and enters a query

```python
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.'})
```

```
{
    "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/

What is Function Calling?

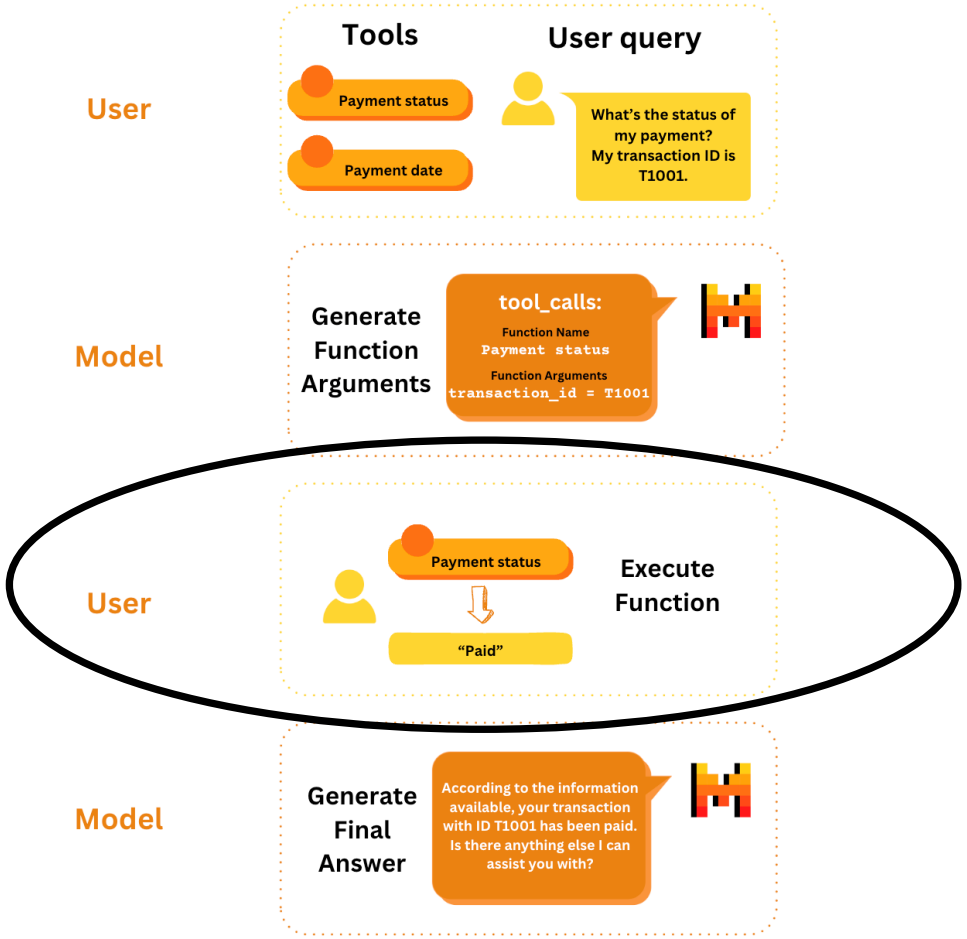


2. Model identifies the function and its arguments

```
tool_calls=[ FunctionCall(name='payment_status',
arguments={"transaction_id": "T1001"}) ]
```

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

What is Function Calling?



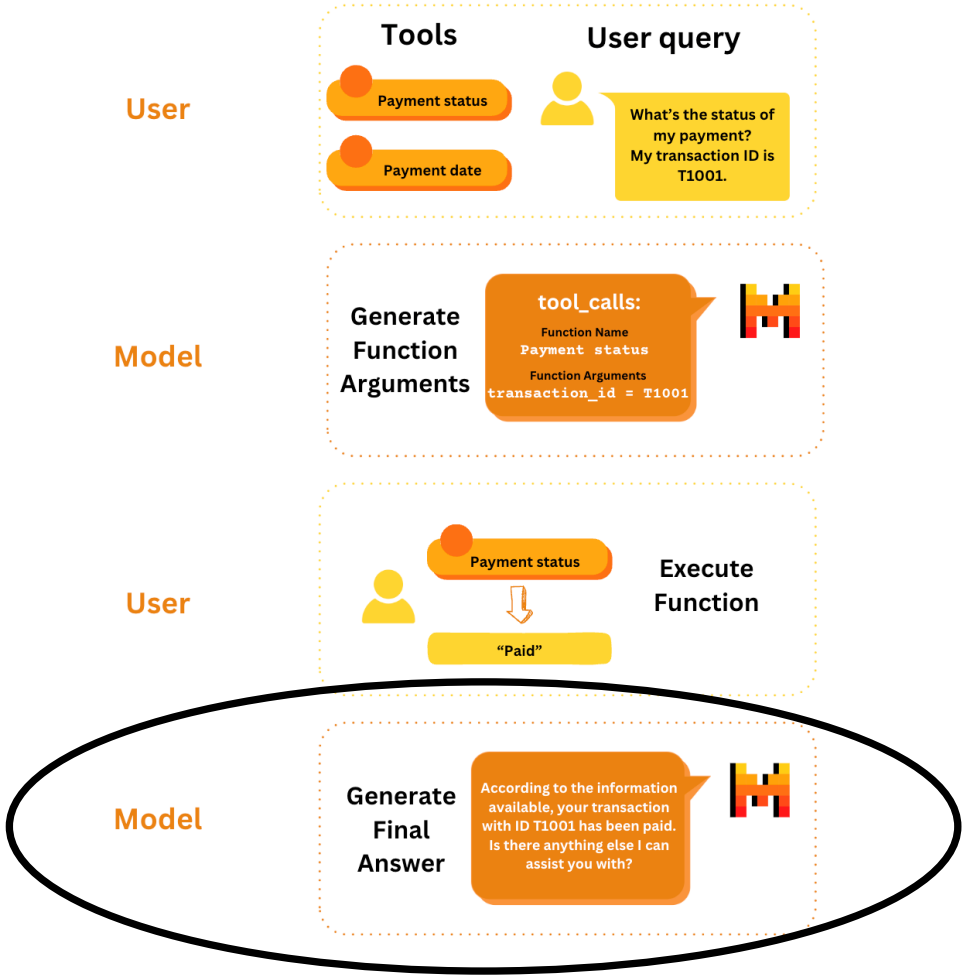
3. User executes the function to obtain tool results

Function Call Output:

```
{"status": "Paid"}
```

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

What is Function Calling?



4. Model uses the results to generate the final answer

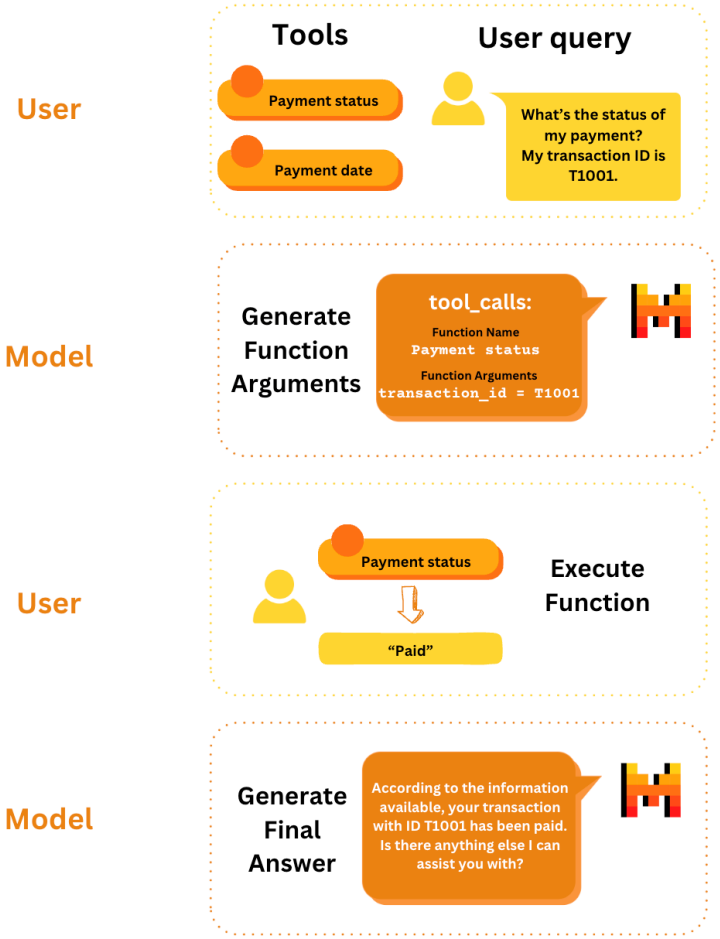
Function Call Output:

```
{"status": "Paid"}
```

The status of your transaction with ID T1001 is "Paid". Is there anything else I can assist you with?

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

What is Function Calling?



1. User specifies tools and enters a query
2. Model identifies the function and its arguments
3. User executes the function to obtain tool results
4. Model uses the results to generate the final answer

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

What data do we need?

What's the status of my transaction T1001?

Tools

```
tool_calls=[ FunctionCall(name='payment_status',
arguments={"transaction_id": "T1001"}) ]
```

What data do we need?

What's the status of my transaction T1001? | Tools

```
tool_calls=[ FunctionCall(name='payment_status',
arguments={"transaction_id": "T1001"}) ]
```

What's the status of my transaction T1001? | Tool Output

The status of your transaction with ID T1001 is "Paid". Is there anything else I can assist you with?

What data do we need?

What's the status of my transaction T1001? | Tools

```
tool_calls=[ FunctionCall(name='payment_status',
arguments={"transaction_id": "T1001"}) ]
```

What's the status of my transaction T1001? | Tool Output

The status of your transaction with ID T1001 is "Paid". Is there anything else I can assist you with?

What's the status of my transaction? | Tools

```
tool_calls=[ FunctionCall(name='payment_status',
arguments={"transaction_id": "?"}) ]
```

Hi there! I can help with that. Can you please provide your transaction ID?

Outline

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

Gorilla: Large Language Model Connected with Massive APIs

1. Synthesized a 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

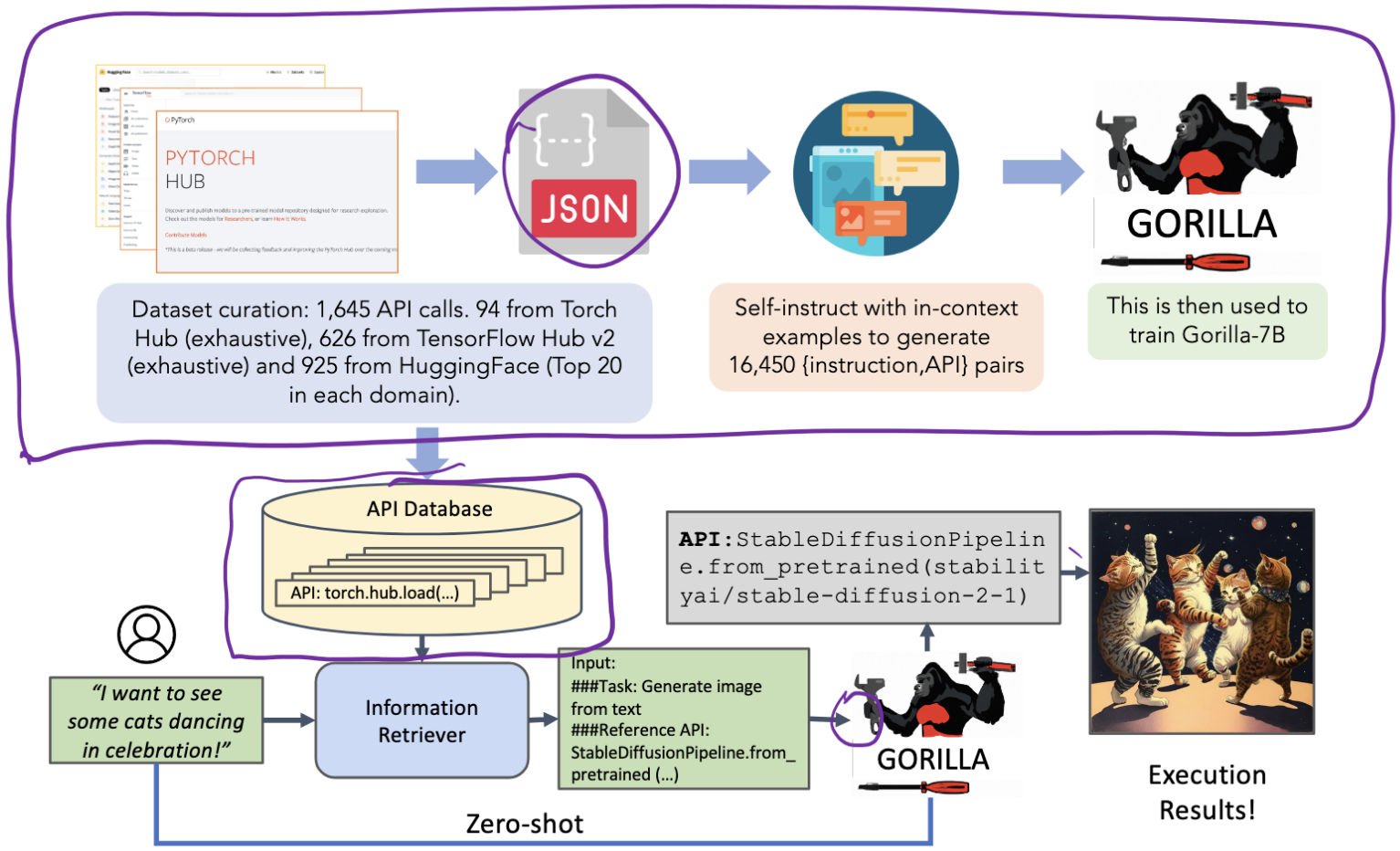
*Gorilla: Large Language Model Connected with Massive APIs, Patil et. al., Nov 2023

Gorilla: Large Language Model Connected with Massive APIs

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:
<domain>: image-object-detection,
<api_call>: detector = hub.load('https://tfhub.dev/google/openimages_v4/ssd_mobilenet_v2/1'),
<api_provider>: TensorFlow Hub,
<explanation>: This API call loads the pre-trained Single Shot Detector (SSD) MobileNet V2 model from TensorFlow Hub, which has been trained on the Open Images v4 dataset. This model is used for object detection in images. Once loaded, you can pass an image (and an optional score threshold for detection confidence) to the detector to get back the detected objects along with their bounding boxes, class labels, and associated confidence scores.
<code>: import tensorflow as tf; import tensorflow_hub as hub; # Load the SSD MobileNet V2 model; detector = hub.load('https://tfhub.dev/google/openimages_v4/ssd_mobilenet_v2/1'); # Use the model to detect objects in an image (assuming 'image' is a TensorFlow tensor representing your image); result = detector(image, score_threshold=0.5)

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

Gorilla: Large Language Model Connected with Massive APIs

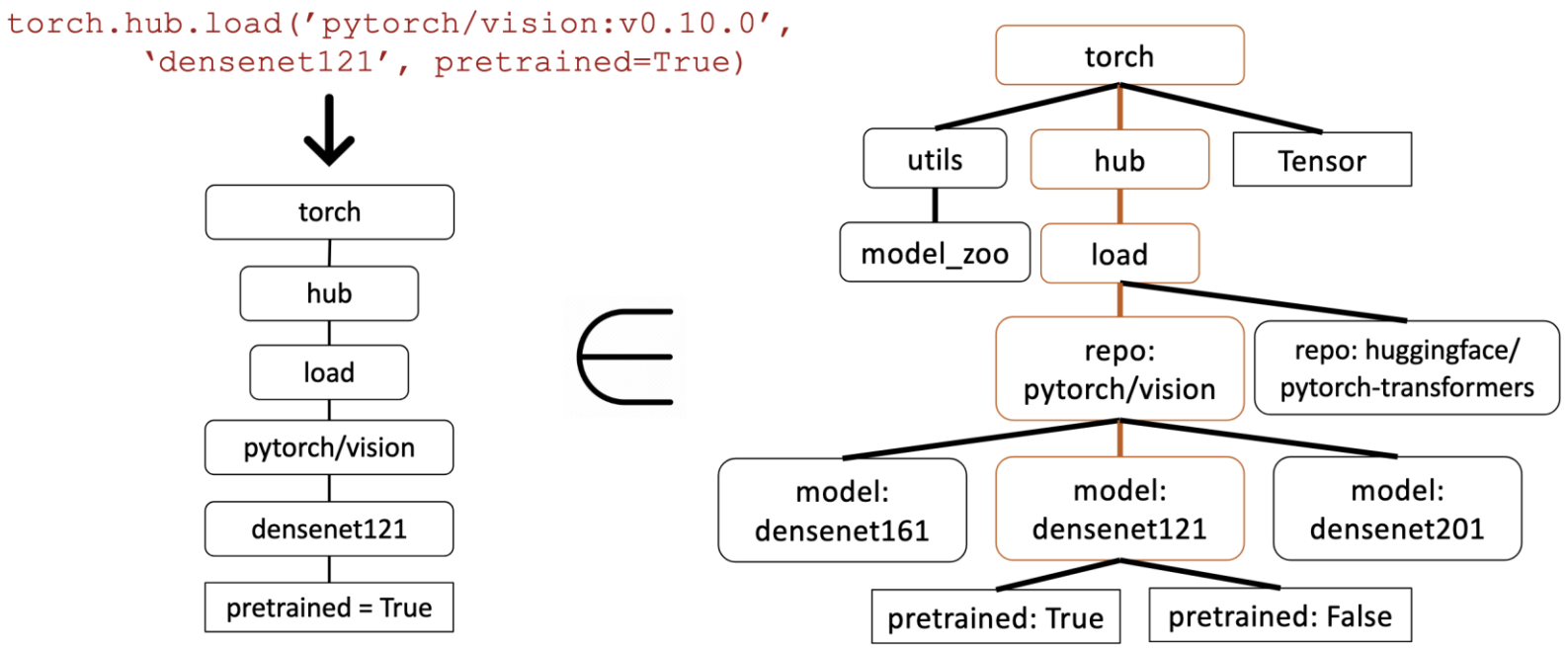


First to use retriever augmented training for APIs

- 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



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
<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)	<domain>: Audio-Translation <api_provider>: Pytorch <code>: import torchaudio translation = Torchaudio.pipelines. WAV2VEC2_ASR_PIPELINE("audio.wav")	<domain>: Speech-to-Text <api_provider>: TorchHub <code>: asr_model = torch.hub.load('snakers4/silero-models', 'silero_sst') result = asr_model.transcribe(audio_path)
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 (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

BFCL: Berkeley Function-Calling Leaderboard

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

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

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:

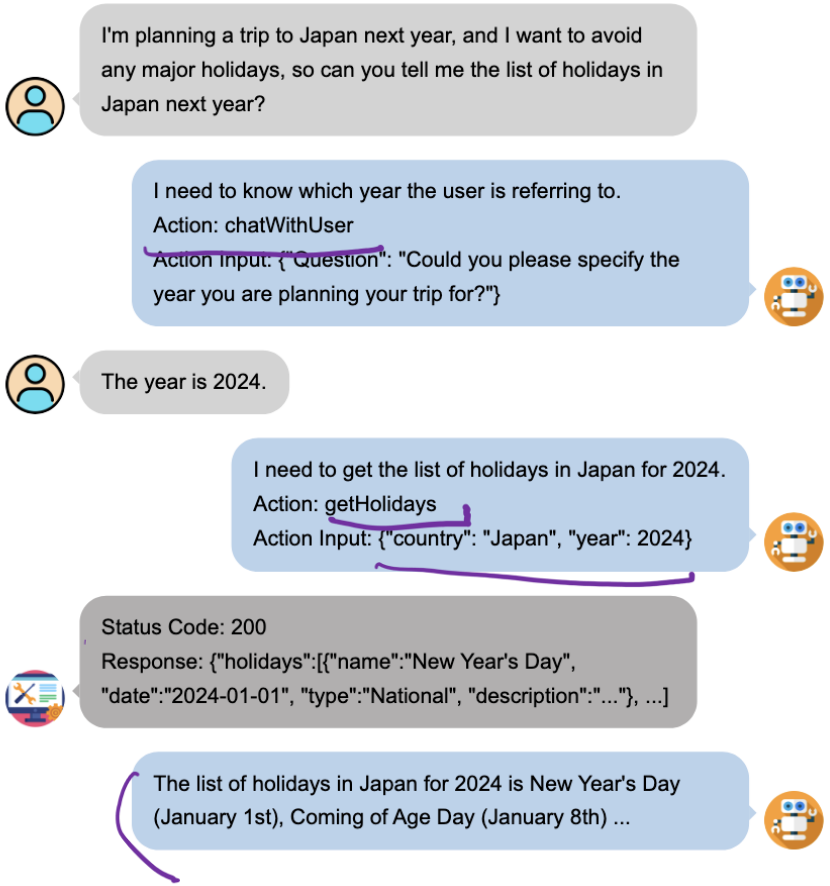
- GET /holidays/{country} - GET /holidays/{holidayId}/details - GET /holidays/search

*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.



*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

*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

2. ToolAlpaca Dataset
    - High Diversity (400 real world inspired APIs)
    - Multi turn dialogs with question generation and response generation

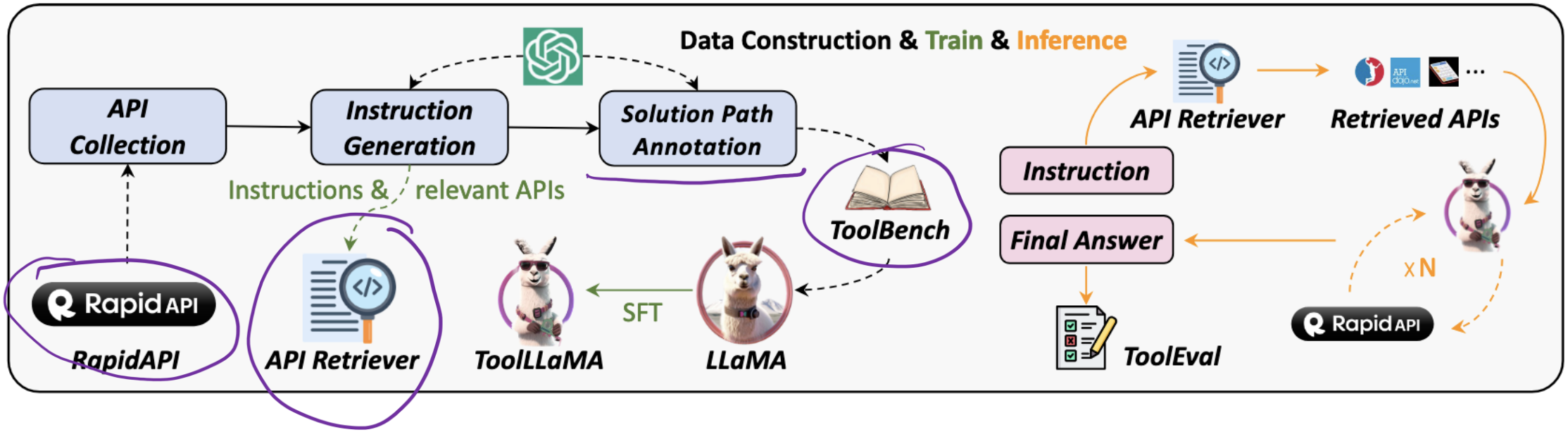
ToolLLM*

1. Synthesized a dataset named `ToolBench`
   - Scraped 16, 464 real-word REST APIs from RapidAPI Hub

2. Finetuned LLaMA-7B model with `ToolBench` to create the `ToolLLaMA` model

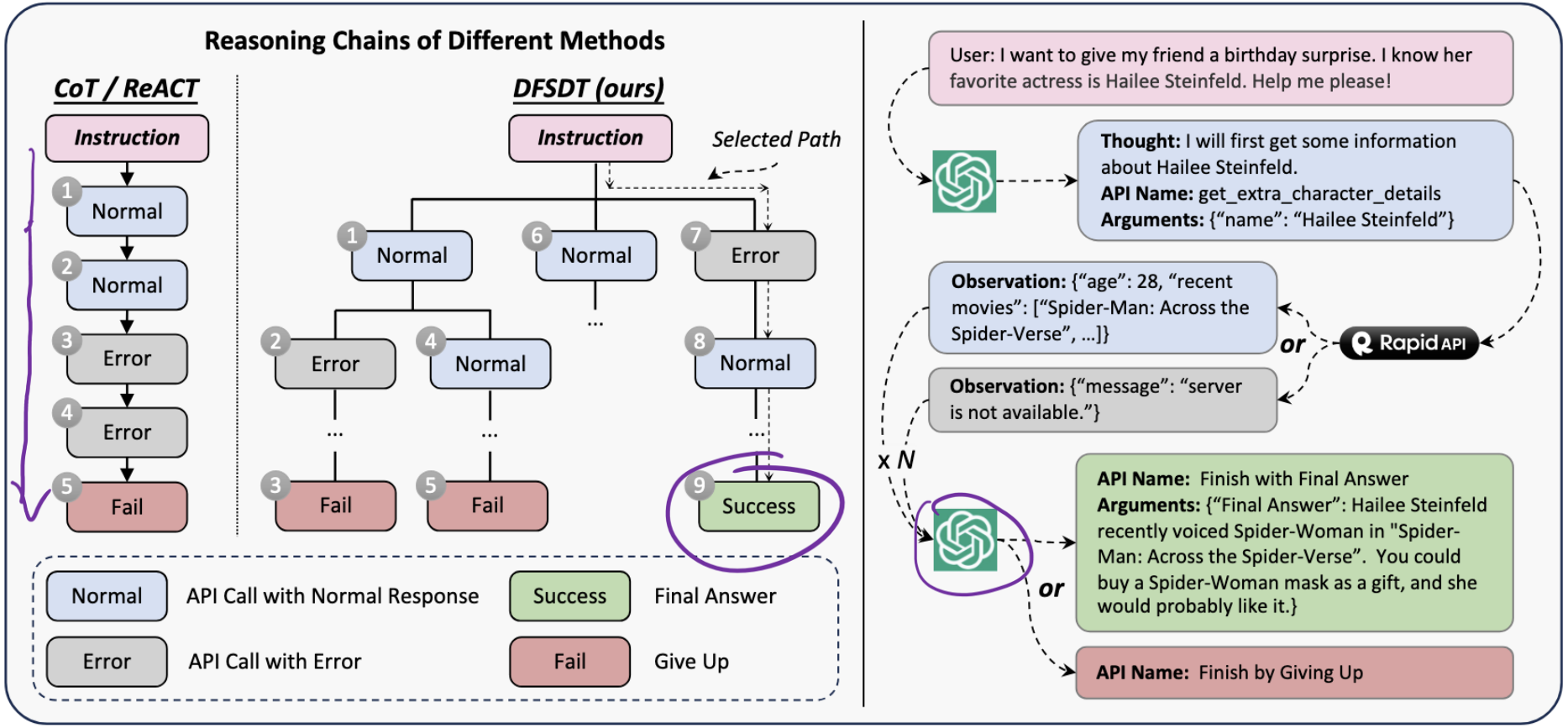
*ToolLLM: Facilitating Large Language Models to Master 16000+ Real World APIs, Qin et. al., Oct 2023

ToolLLM*



*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	10.60	**16.77**	15.70	**51.16**	6.48	40.59
Gorilla-ZS + BM25	46.90	10.51	17.20	44.62	20.58	34.31
Gorilla-RS + BM25	**6.42**	15.71	**5.91**	50.00	**2.77**	**41.90**
ToolLLaMA + Oracle	8.66	88.80	14.12	85.88	7.44	88.62
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

2. ToolAlpaca Dataset
   - High Diversity (400 real world inspired APIs)
   - Multi turn dialogs with question generation and response generation

3. ToolBench (ToolLLM)
   - High diversity (16K real world APIs from RapidAPI Hub)
   - Multi turn dialogs
   - Has single tool setup and multi tool setup

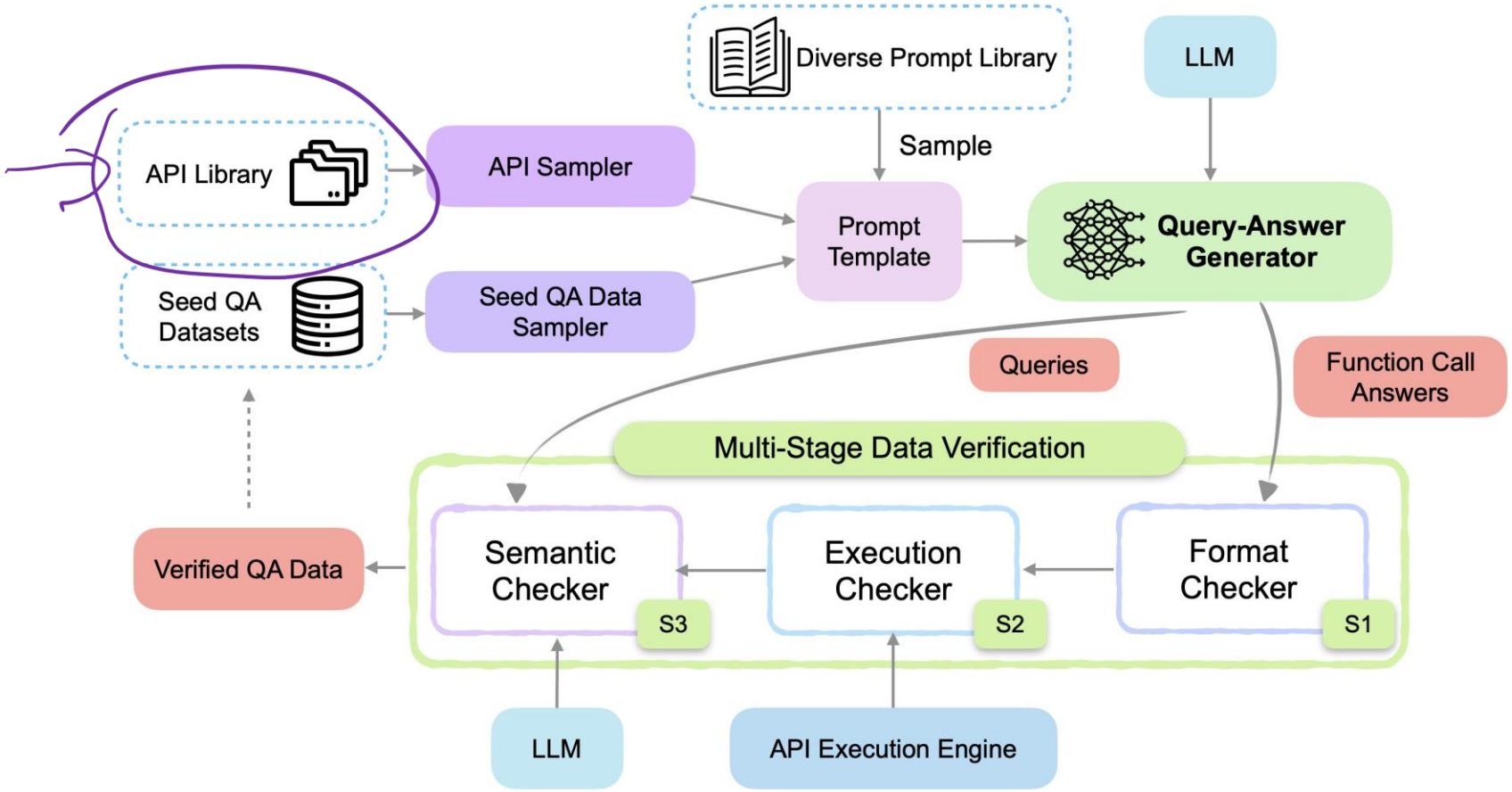
APIGen*

1. Synthesized a dataset named `xlam-function-calling-60k`
   - Used only 3673 real-word REST 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 `xLAM-1B(FC)` (and `7B`) model

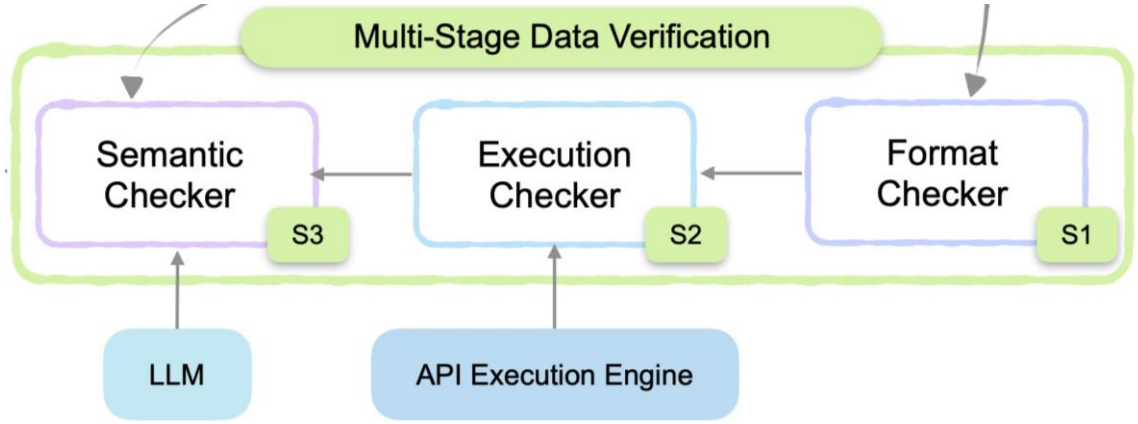
* APIGen: Automated PIpeline for Generating Verifiable and Diverse Function-Calling Dataset, Liu et. al., Jun 2024

APIGen*



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

APIGen*



APIs

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

Function Call

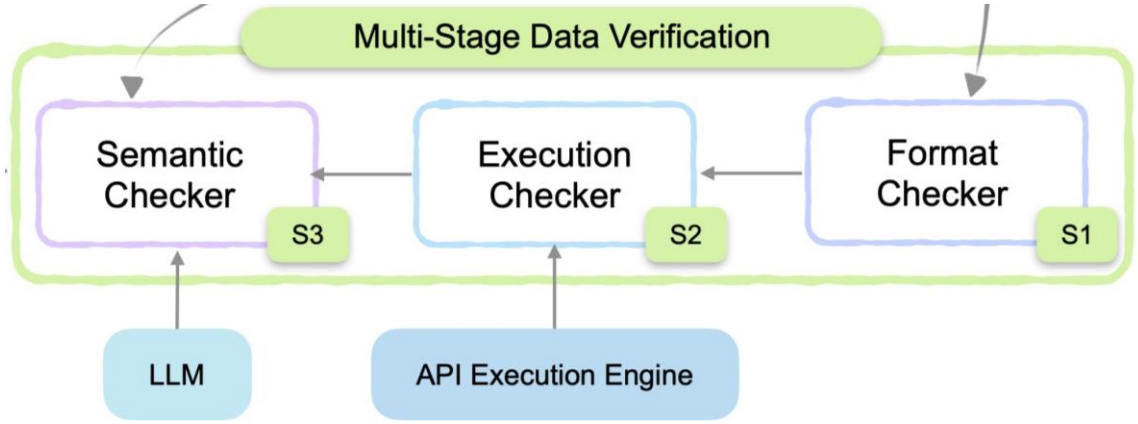
```
[
 {
   "name": "api_name",
   "arguments": {
     "arg_name1": "value1",
     "arg_name2": "value2",
     ... (more arguments)
   }
 }
 ... (more API calls)
]
```

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*

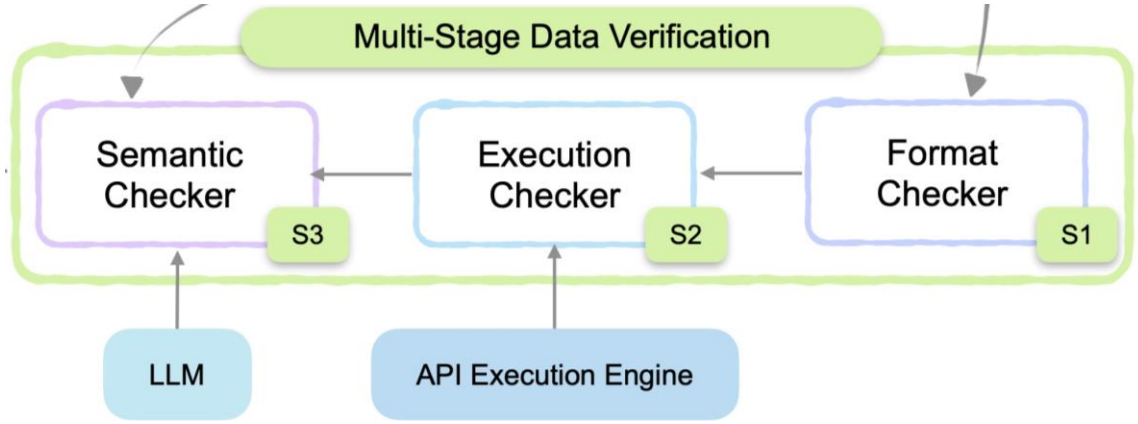


Execution Checker

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

APIGen*

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

2. ToolAlpaca Dataset
   - High Diversity (400 real world inspired APIs)
   - Multi turn dialogs with question generation and response generation

3. ToolBench (ToolLLM)
   - High diversity (16K real world APIs from RapidAPI Hub)
   - Multi turn dialogs
   - Has single tool setup and multi tool setup

4. xlam-function-calling-60k (APIGen)
   - High diversity (3K high quality real world APIs from RapidAPI Hub)
   - high quality multi turn dialogs – thanks to the 3 stage filtering
   - Has single tool setup and multi tool setup along with its parallel variants

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

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

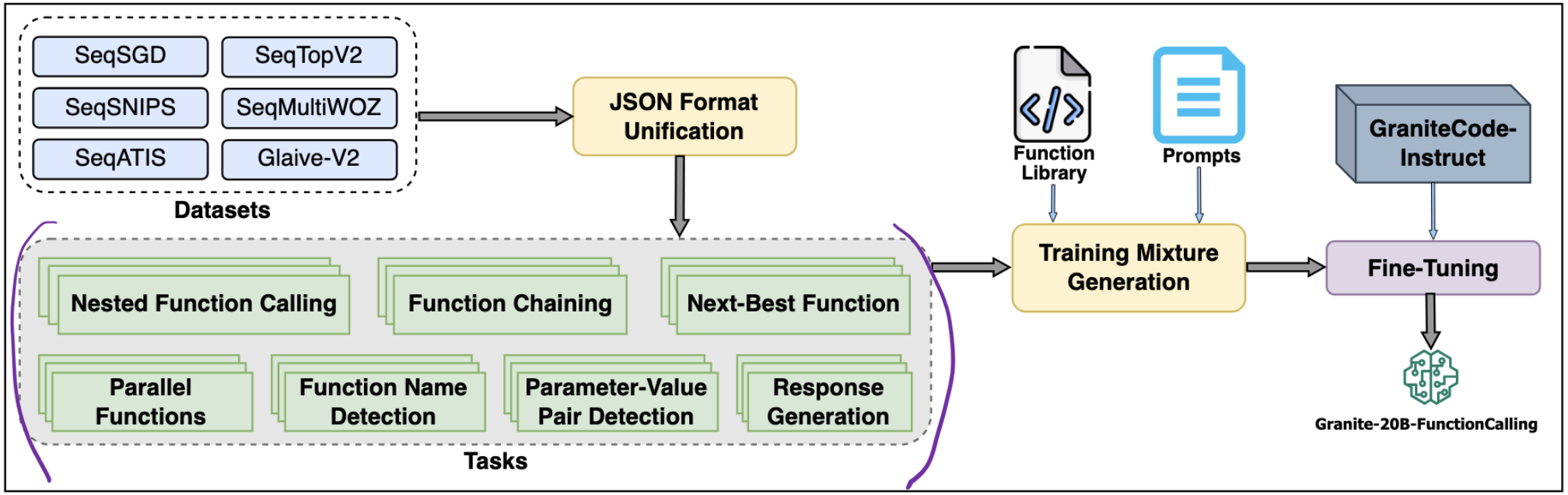
*Granite-Function Calling Model: Introducing Function Calling Abilities via Multi-task Learning of Granular Tasks, Abdelaziz et. Al., Jun 2024

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

Granite-Function-Calling-Model*



*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

2. ToolAlpaca Dataset
   - High Diversity (400 real world inspired APIs)
   - Multi turn dialogs with question generation and response generation

3. ToolBench (ToolLLM)
   - High diversity (16K real world APIs from RapidAPI Hub)
   - Multi turn dialogs
   - Has single tool setup and multi tool setup

4. xlam-function-calling-60k (APIGen)
   - High diversity (3K high quality real world APIs from RapidAPI Hub)
   - high quality multi turn dialogs – thanks to the 3 stage filtering
   - Has single tool setup and multi tool setup along with its parallel variants

5. Granite-Function-Calling-Model
   - High diversity
   - Has single tool, & multiple tool setup
   - Has multi turn dialogs
   - Open-sourced data and model with permissible license