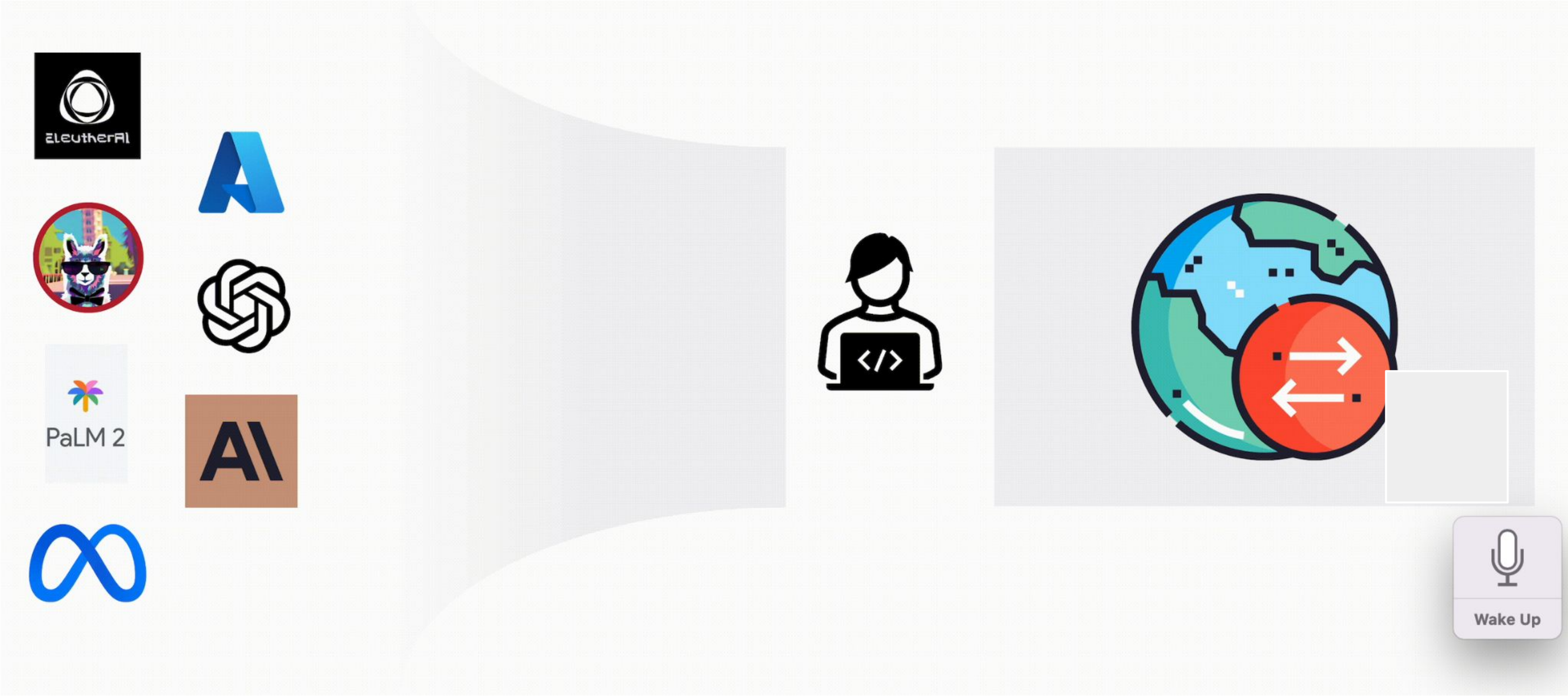
Gorilla: Large Language Model Connected with Massive APIs

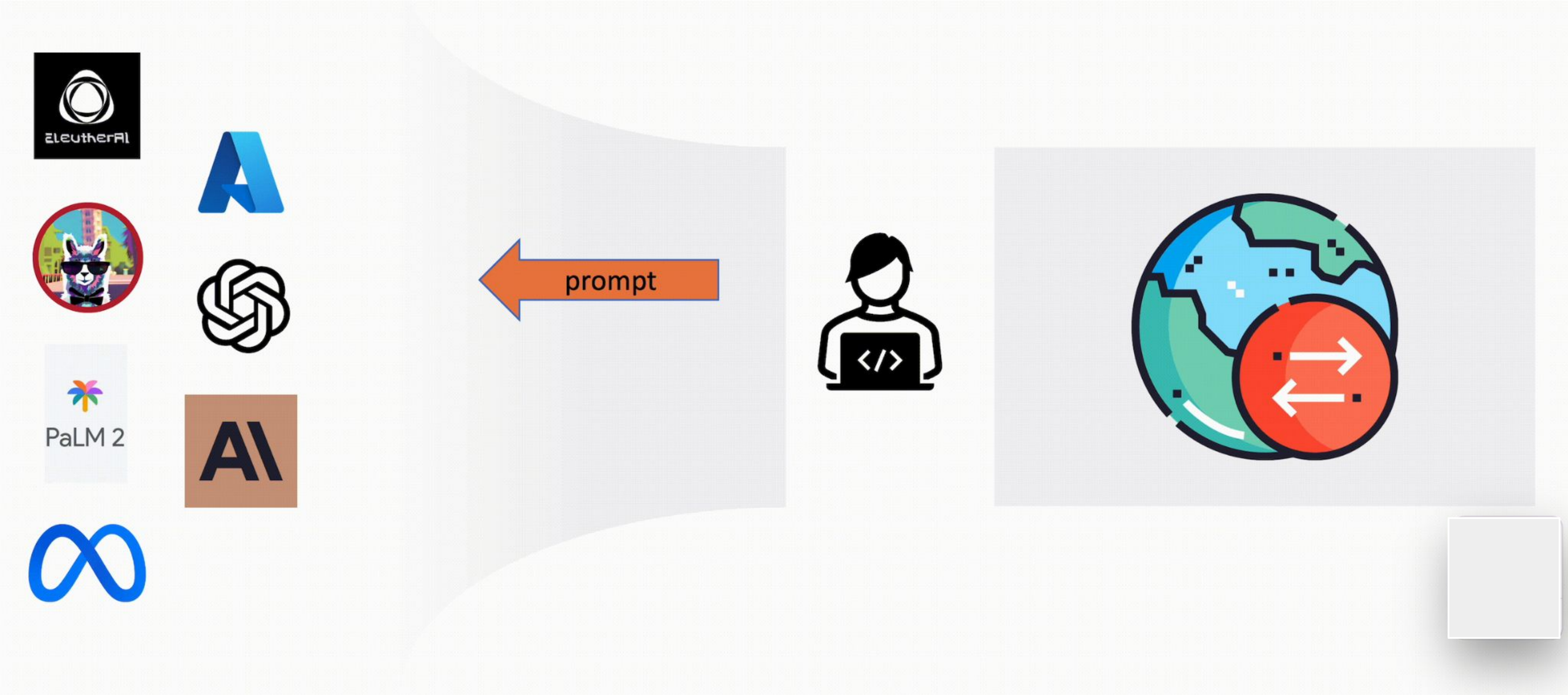
Shishir G. Patil
UC Berkeley, Microsoft Research

Introduction

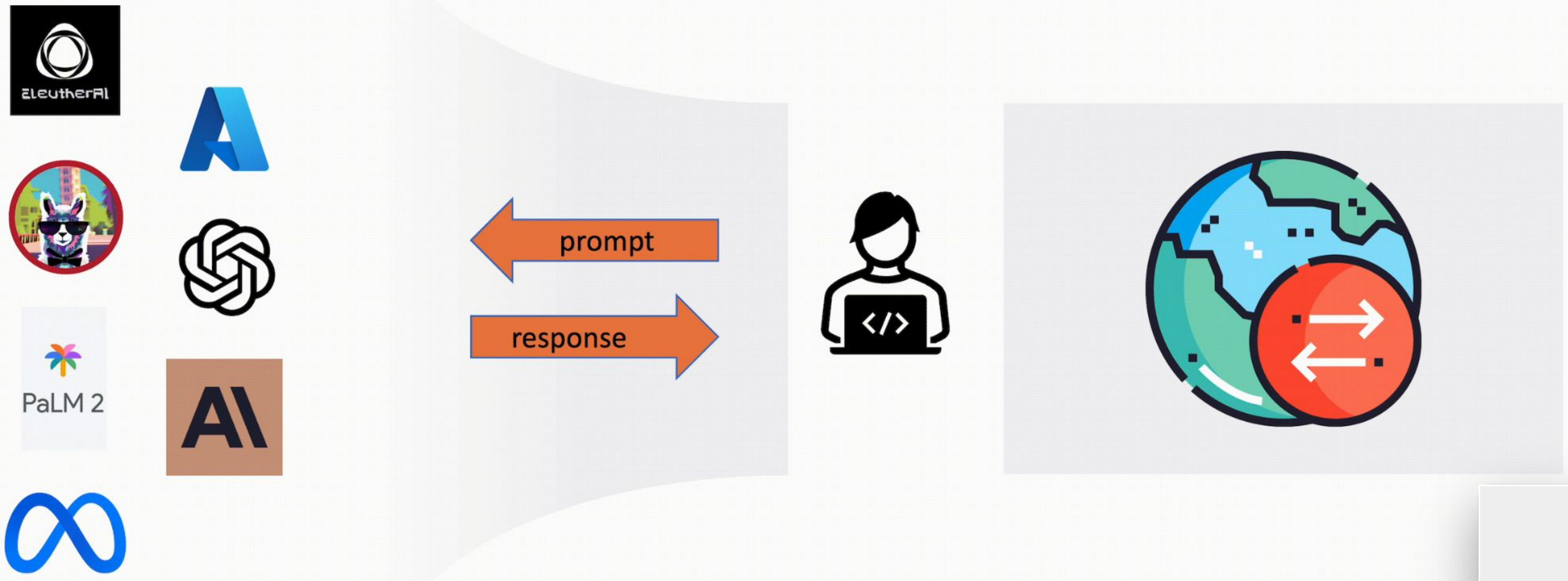
Gorilla LLM interacts with the world



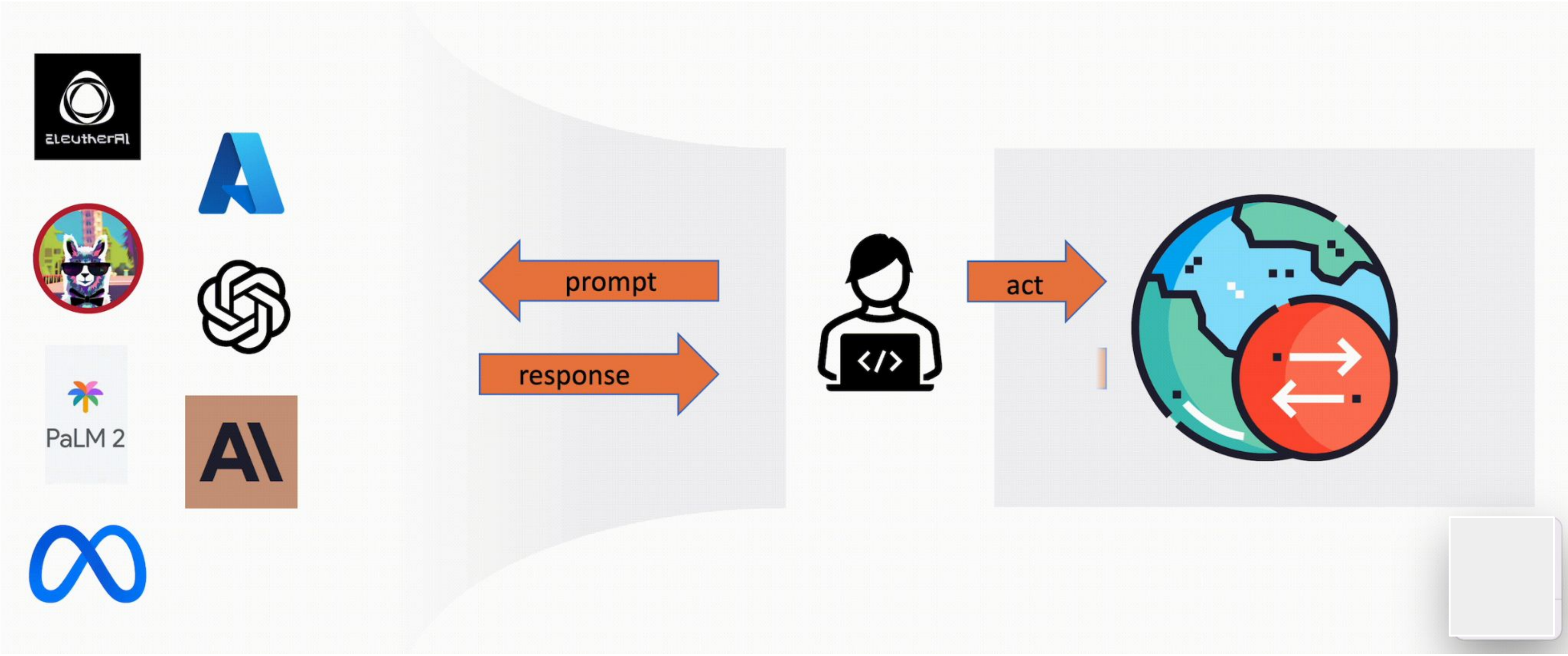
Gorilla LLM interacts with the world



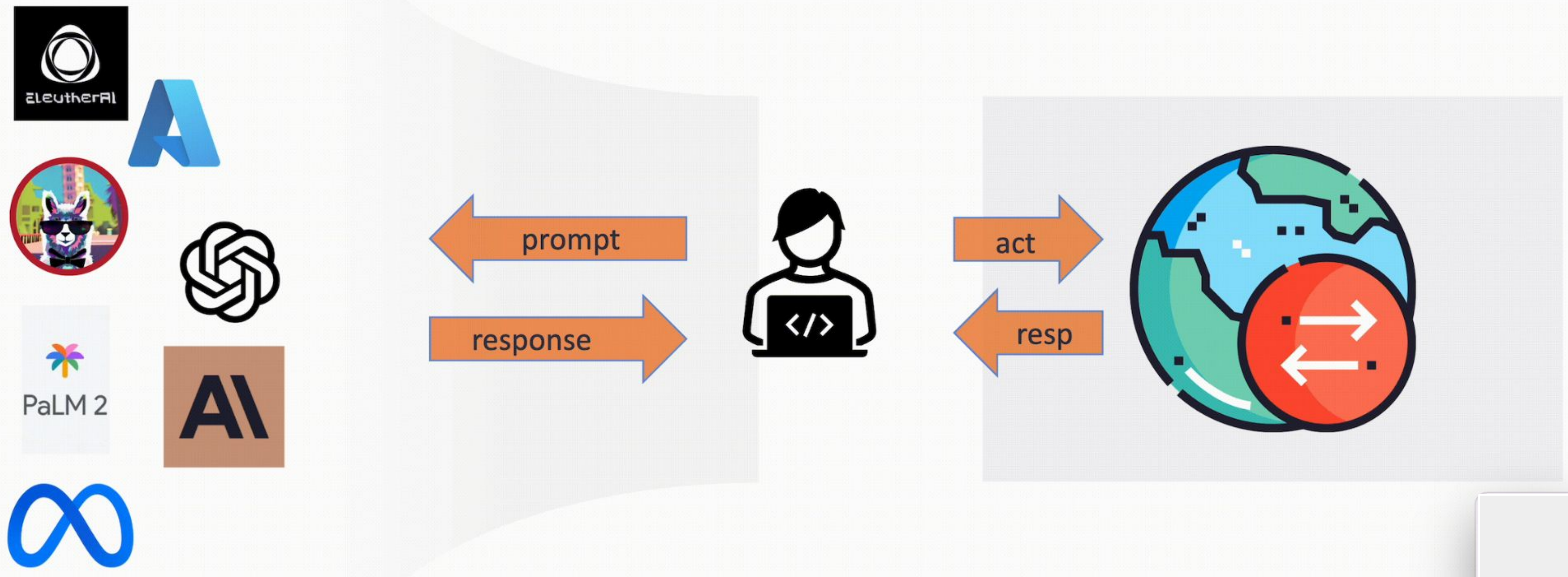
Gorilla LLM interacts with the world



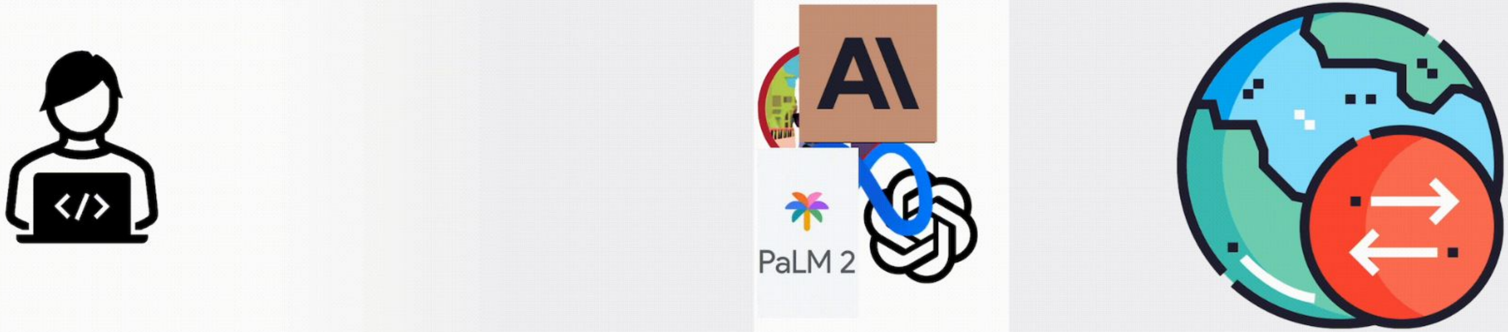
Gorilla LLM interacts with the world



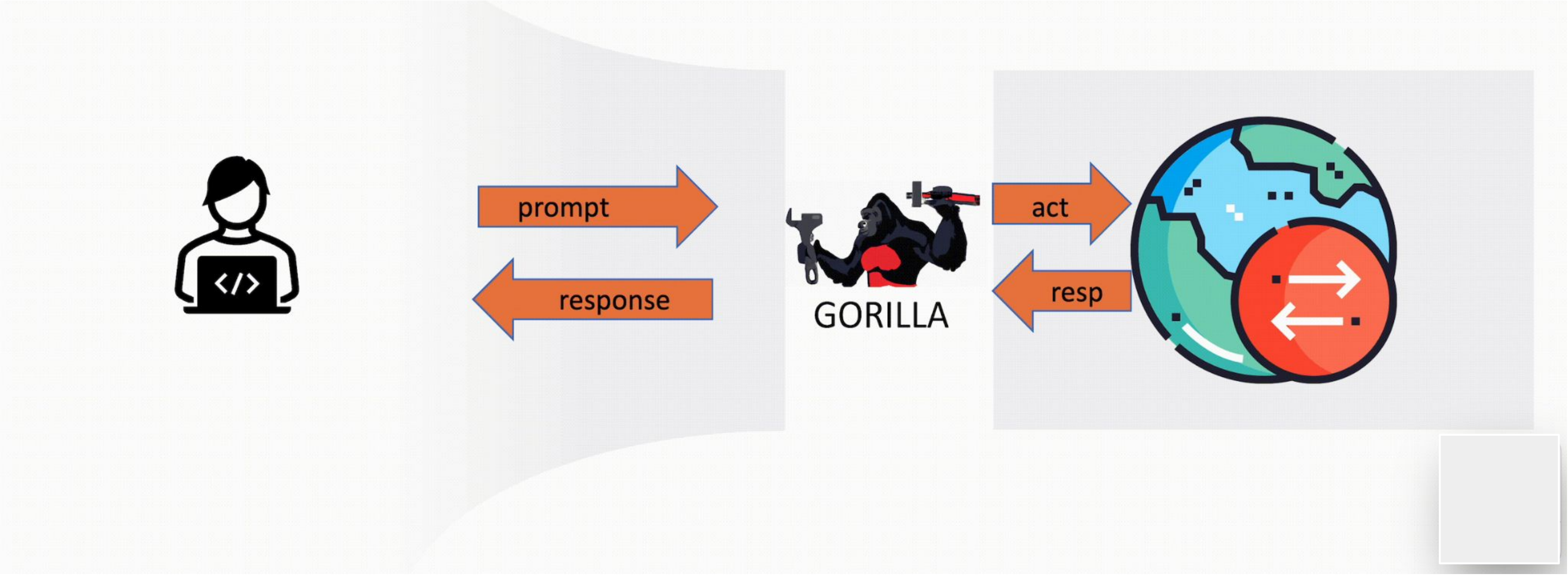
Gorilla LLM interacts with the world



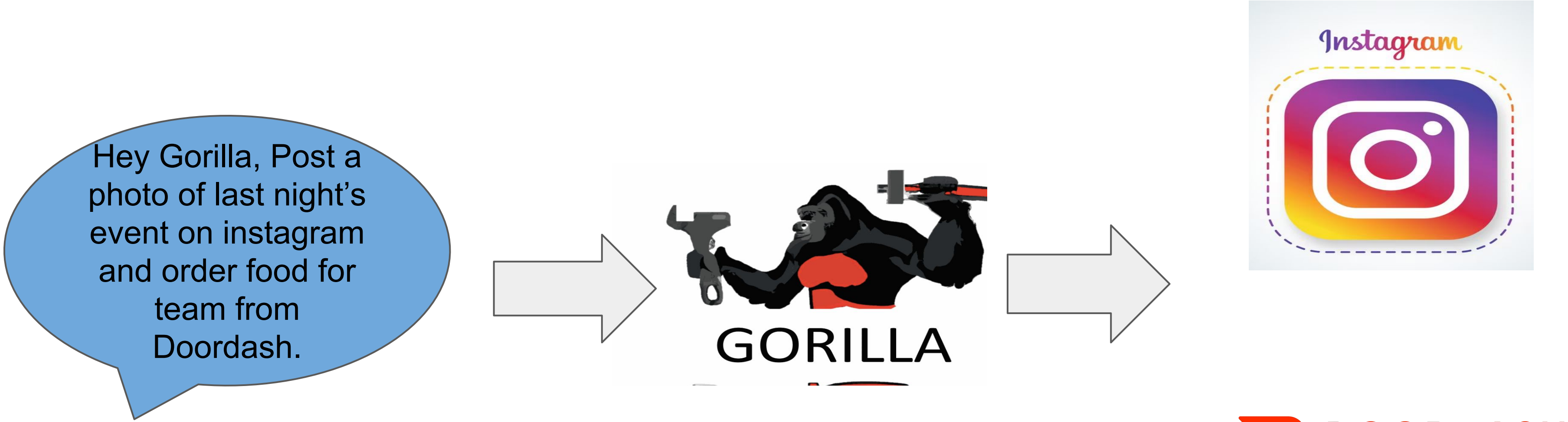
CAN WE FLIP THE INTERACTION?



Gorilla LLM interacts with the world




Hey Gorilla, Schedule a meeting with my team next friday afternoon
GORILLA
31
Google Calender

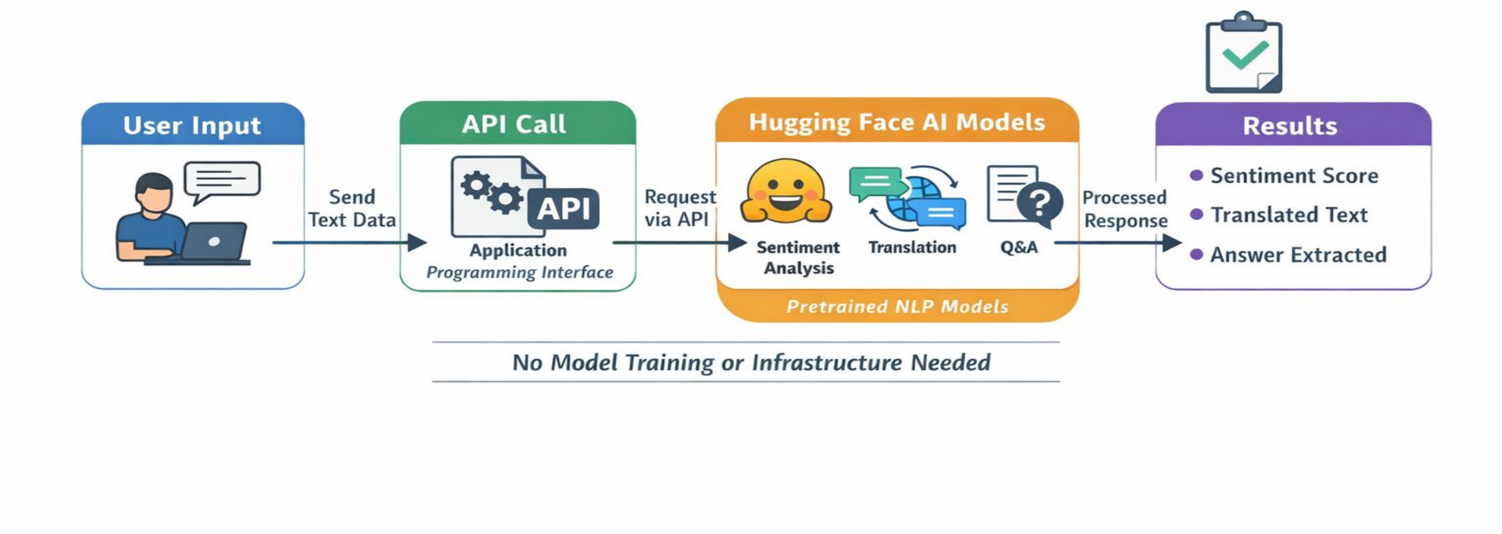

Hey Gorilla, Post a photo of last night's event on instagram and order food for team from Doordash.
GORILLA
Instagram
DOORDASH

INSPIRATION

- Humans are good **Discriminator**
- LLMs are good **Generators**.

A DEMO OF HOW API WORKS?

https://github.com/MeenakshiRajpurohit/Chat_bot_API/blob/main/Hugging_face_API_demo.ipynb%20-%20Colab.pdf



PROBLEM STATEMENT

- Frequent updation of API(not effectively using the tools)
- How to mix fine-tuning and retrievals?
- Hallucinations

EXAMPLE : FREQUENT API UPDATION

API Changes

2023/10/02 - bedrock - 5 new api methods

Changes Provisioned throughput feature with Amazon and third-party base models, and update validators for model identifier and taggable resource ARNs.

2023/10/02 - bedrock-runtime - 1 updated api methods

Changes Provisioned throughput feature with Amazon and third-party base models, and update validators for model identifier and taggable resource ARNs.

2023/10/02 - ec2 - 24 updated api methods

Changes Introducing Amazon EC2 R7iz instances with 3.9 GHz sustained all-core turbo frequency and deliver up to 20% better performance than previous generation z1d instances.

2023/10/02 - rds - 1 updated api methods

Changes Adds DefaultCertificateForNewLaunches field in the DescribeCertificates API response.

2023/09/28 - bedrock - 15 new api methods

Changes Model Invocation logging added to enable or disable logs in customer account. Model listing and description support added. Provisioned Throughput feature added. Custom model support added for creating custom models. Also includes list, and delete functions for custom model.

APIs evolve frequently! For example, there were 31 API modifications for AWS APIs just yesterday.

Gorilla: The vision

- **Key idea 1: Retriever Aware Training(RAT)**

- Key idea 2: Measuring Hallucination

Big Question: **How to mix Fine-Tuning and Retrieval?**

Hypothesis:

- Fine-tuning: augment the **behaviour** of model
- Retrieval: Introduce new **knowledge** to the model

Early Evidence(Gorilla): fine-tuning is effective at both behaviour and knowledge.

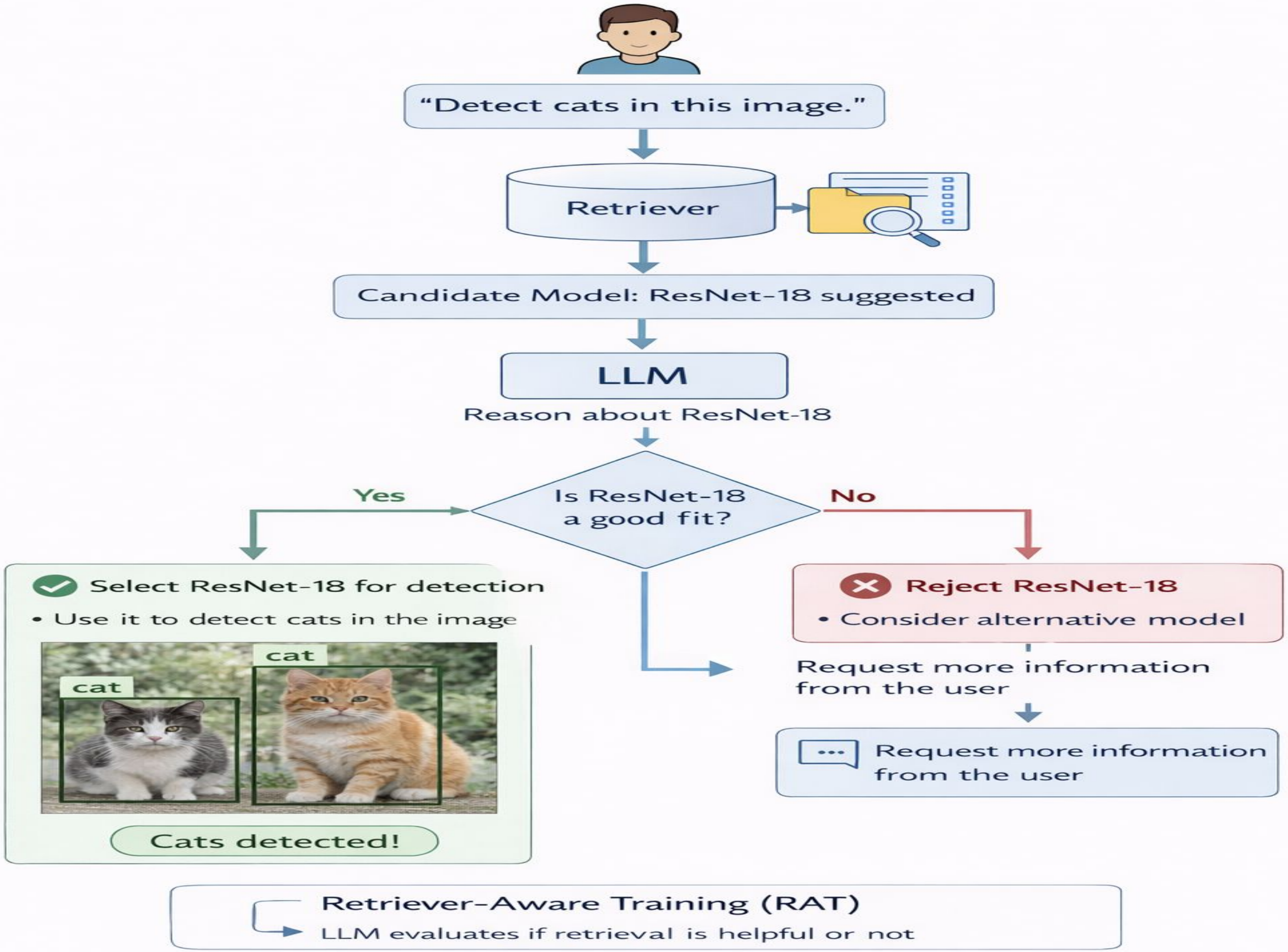
- ..but **retrievers are needed for data freshness and are inaccurate.**

Big Idea: Retrieval Aware Training(RAT)

Fine-tune the model to **use** or **ignore** retrieved context.

- Introduce correct and incorrect **retrieval** results during **instruction fine-tuning**

- Ensure model is **robust to low-quality retrieval.**



"Detect cats in this image."
Retriever
Candidate Model: ResNet-18 suggested
LLM
Reason about ResNet-18
Is ResNet-18 a good fit?
Yes
No
Select ResNet-18 for detection
• Use it to detect cats in the image
cat
cat
Cats detected!
Reject ResNet-18
• Consider alternative model
Request more information from the user
Request more information from the user
Retriever-Aware Training (RAT)
LLM evaluates if retrieval is helpful or not

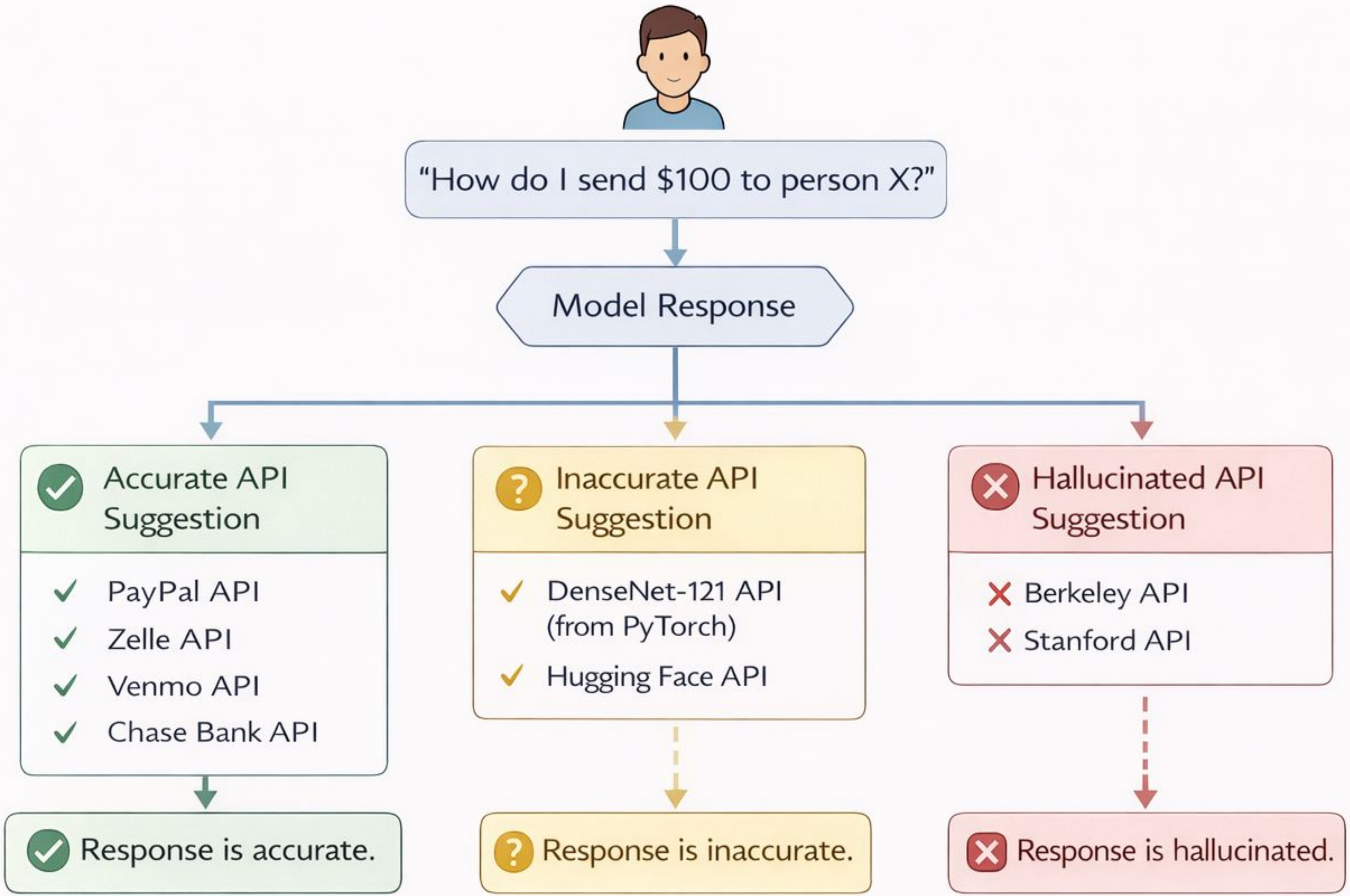
DEMO

https://github.com/MeenakshiRajpurohit/Chat_bot_API/blob/main/Gorilla_LLM_Demo.ipynb

Gorilla: The vision

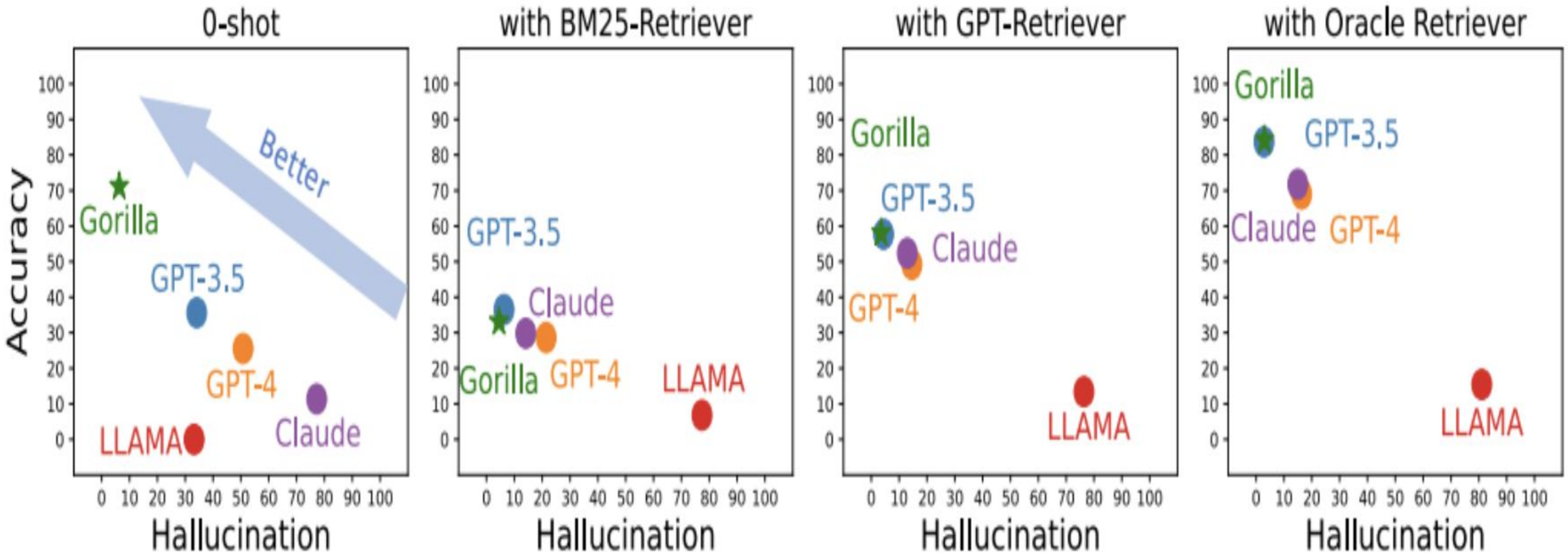
- Key idea 1: Retriever Aware Training(RAT)

- **Key idea 2: Measuring Hallucination**



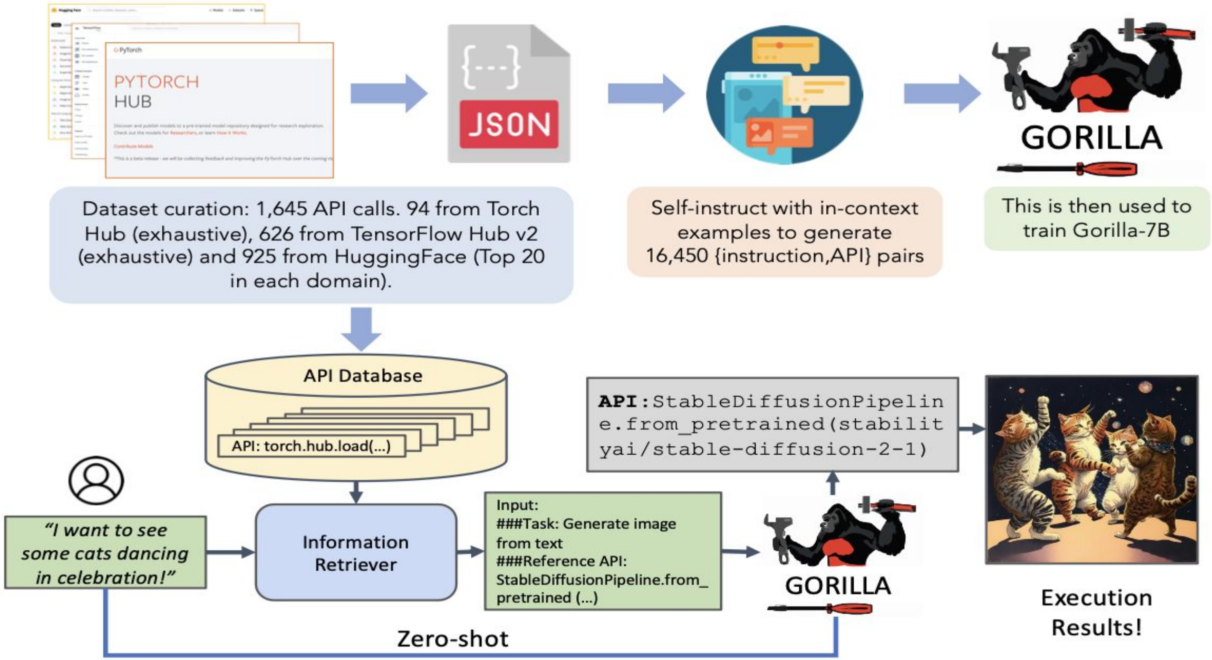
“How do I send $100 to person X?”
Model Response
Accurate API Suggestion
PayPal API
Zelle API
Venmo API
Chase Bank API
Response is accurate.
Inaccurate API Suggestion
DenseNet-121 API (from PyTorch)
Hugging Face API
Response is inaccurate.
Hallucinated API Suggestion
Berkeley API
Stanford API
Response is hallucinated.

Accuracy vs Hallucination



Methodology

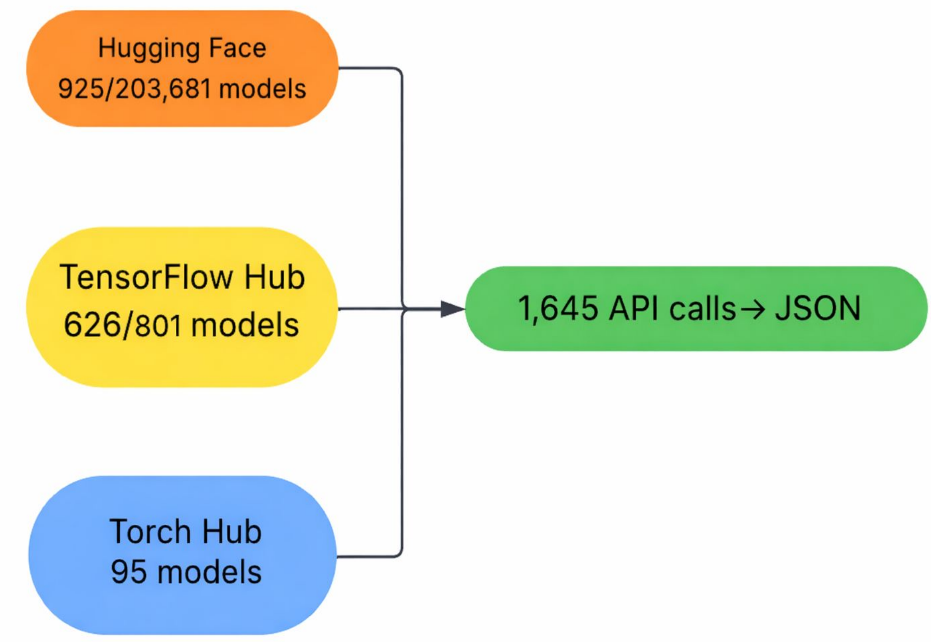
Gorilla: A system for enabling LLMs to interact with APIs.



Dataset Curation

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).

Data collected → Hugging Face Model, PyTorch and TensorFlow Hubs



- Filtering Models - poor documentation,have no information in their model card, etc
- Top 20 models from each domain - 7 domains in multimodal data, 8 in CV, 12 in NLP, 5 in Audio, 2 in tabular data, and 2 in reinforcement learning.
- domain, framework, functionality, api_name, api_call, api_arguments, environment_requirements, example_code, performance and description

API Call with Constraints

Example User Prompt: “Invoke an image classification model that **uses less than 10M parameters**, but maintains an ImageNet **accuracy of at least 70%.**”

→ common constraints are parameter size and a lower bound on accuracy

→ Why is it challenging for LLMs:

1. Parse the natural language correctly for constraints
2. Identify the constraints in prompt
3. Filter the API's according to constraints

→ During *training the dataset is added with instruction that has constraints* - **Gorilla Learns!**

→ How Constraints are expressed in natural language and also how constraints affect the API selection.

Instruction Generation

Self-instruct with in-context examples to generate 16,450 {instruction,API} pairs

→ **Generating 16,450 Instruction - API pairs**

→ **Only 18 examples hand-generated**

1. Select one hub
2. Create 6 human written Instruction - API pair examples for that hub
3. Randomly pick 3 examples from 6 human written examples
4. Give the picked 3 (in- context) + API documentation to GPT-4
5. GPT-4 generates 1 API - instruction pair
6. Repeated step 3 to 5 → 10 times

How Gorilla is *built*?

GPT-4 self-instruct output

Instruction: <instruction>
API: <API call>

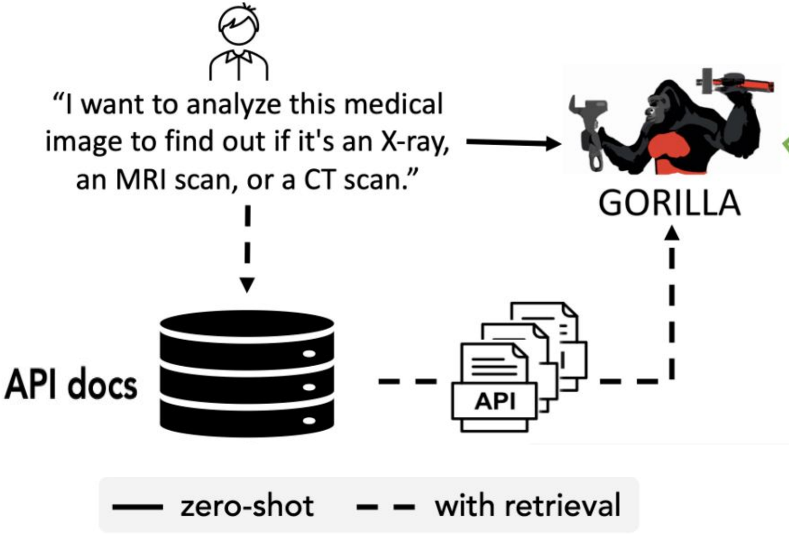
Chat-style conversation

User: <instruction>
Assistant: <API call>

Each datapoint is a conversation with one round each for the user and the agent.

- API Docs
- GPT-4 (self-instruct)
- Instruction–API pairs
- Chat-style conversion
- Instruction fine-tuning (LLaMA-7B)
- *Gorilla is Ready!*

RAT - Retriever Aware Training



Training **without RAT**
User: <instruction>
Assistant (target): <API call>

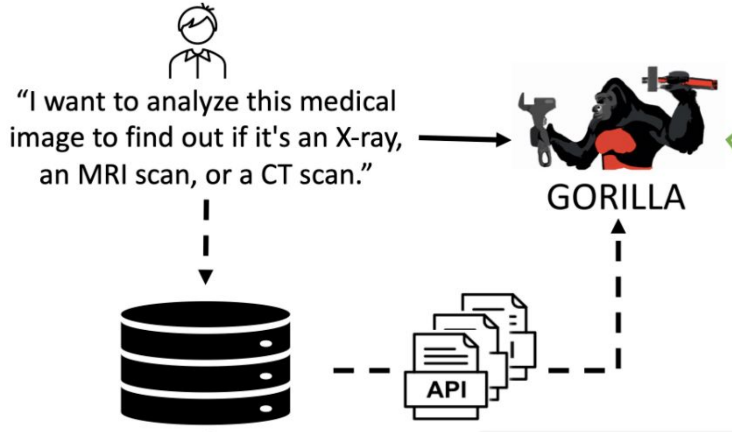
Training **with RAT**
User: <instruction>
Use this API documentation for reference:
<retrieved_API_doc_JSON>
Assistant (target): <API call>

What is RAT? | What is the problem with RAT? | What does Gorilla learn from RAT? | Advantage of RAT

- Retrieved document is relevant
- Retrieved document is irrelevant

Gorilla Inference - How does Gorilla behave at test time?

"I would like to identify the objects in an image" - Simple task
"I am going to the zoo, and would like to track animals" - Vague goal



→ Prompts - Natural Language

→ Two modes: zero-shot and with retrieval

- **Zero-shot:**
  User Prompt → Gorilla LLM → returns the API call

- **Retrieval:**
  *User Prompt → Retriever (either of BM25 or GPT-Index) → **API documentation + user prompt** → Gorilla LLM → returns the API call*

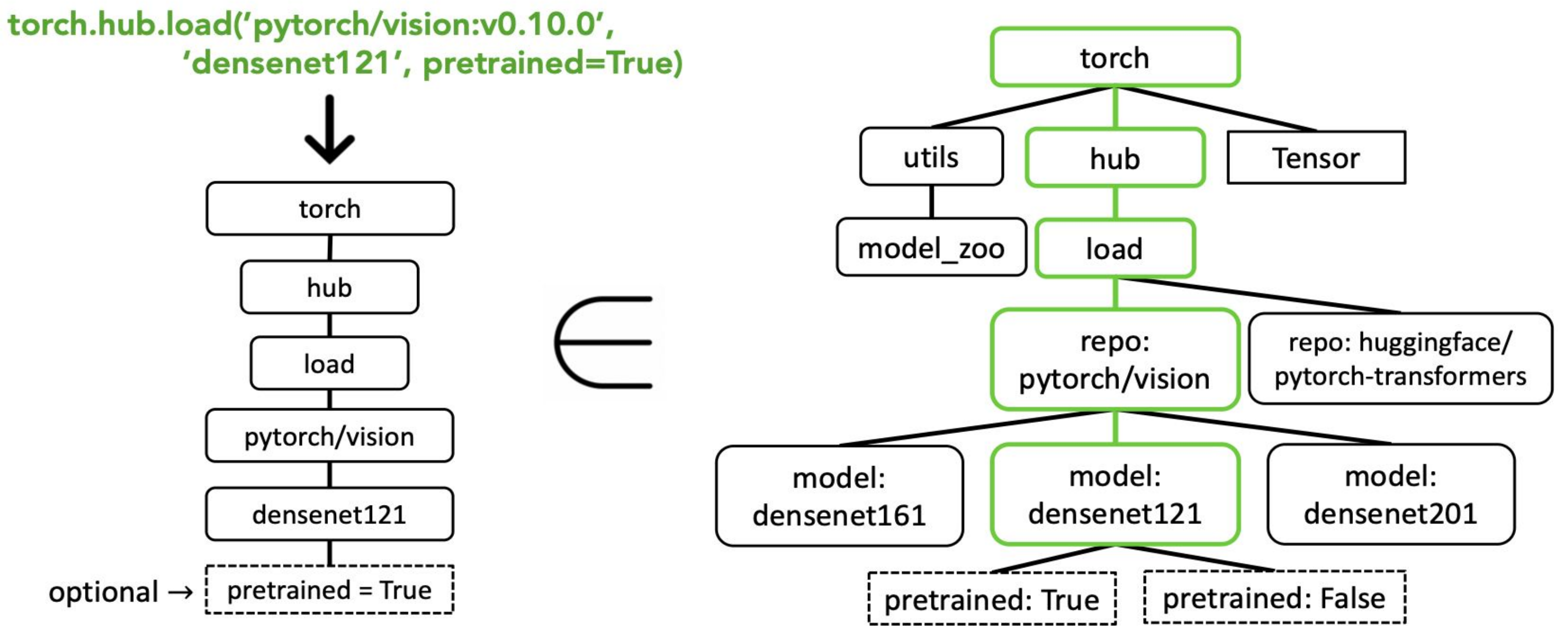
concatenated along with the message
"Use this API documentation for reference."

AST Metric

→ AST is structured representation of code

→ Matching Generated AST and Referenced AST is sufficient

→ with AST, evaluate whether the generated API call is hallucinated without actually executing it.



Evaluation

Evaluation

- How does Gorilla compare to other LLMs on API Bench (*test set*)?

  → Fine Tuning without Retrieval

  → Fine Tuning With Retrieval

  → AST as Evaluation Metrics

- How well does Gorilla adapt to test-time changes in API documentation?

  → Test-Time Documentation Change

- How well can Gorilla handle questions with constraints?

  → API Calls with Constraint

  → Fine Tuning vs Prompting

Fine Tuning without Retrieval

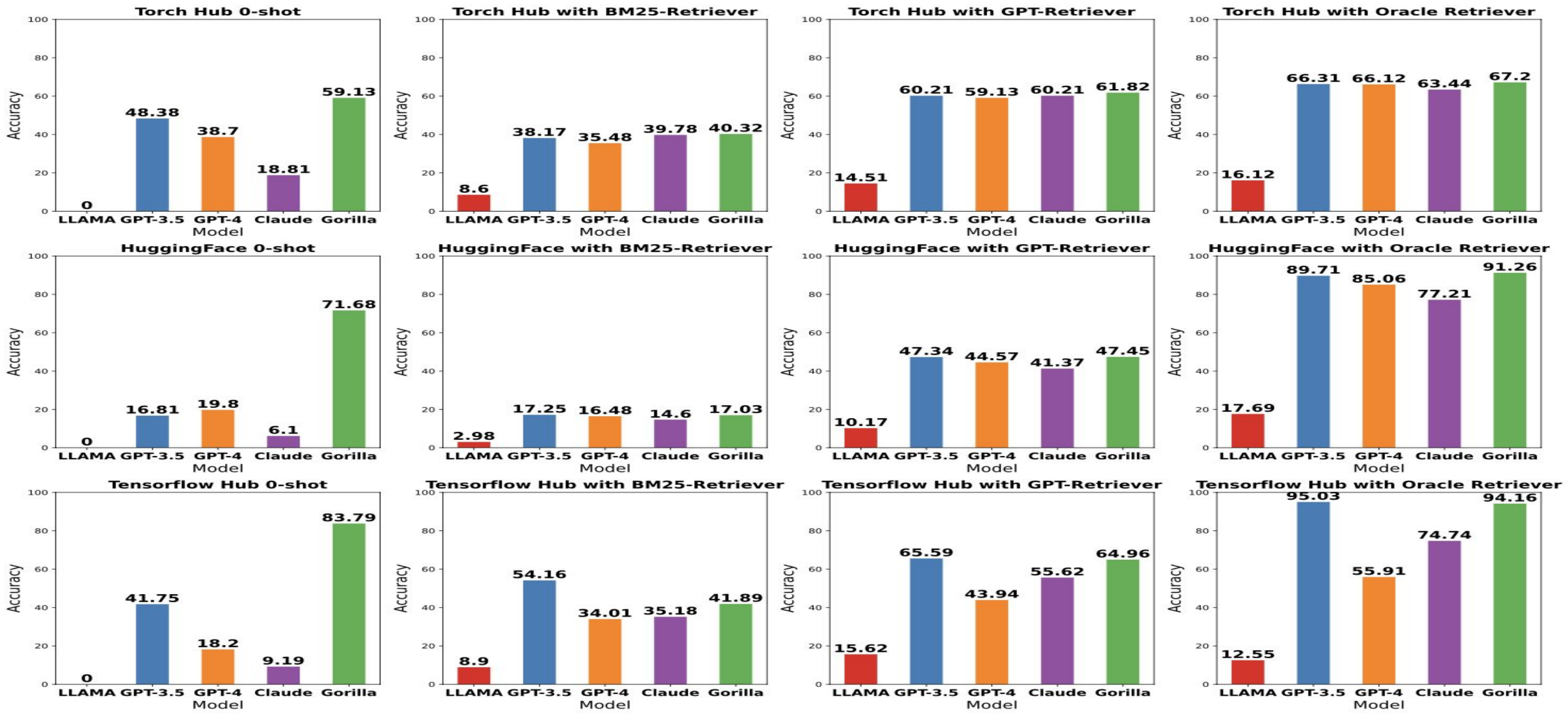


Figure 10: **Performance**: We plot each model's performance on different configurations. We see that Gorilla performs extremely well in the zero-shot setting. While even when the oracle answer is given, Gorilla is still the best.

Table 1: **Evaluating LLMs on Torch Hub, HuggingFace, and Tensorflow Hub 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 (BM-25)	8.60	76.88	14.51	3.00	77.99	19.02	8.90	77.37	13.72
GPT-3.5 (BM-25)	38.17	6.98	54.83	**17.26**	8.30	74.44	**54.16**	3.64	42.18
GPT-4 (BM-25)	35.48	11.29	53.22	16.48	15.93	67.59	34.01	37.08	28.90
Claude (BM-25)	39.78	5.37	54.83	14.60	15.82	69.58	35.18	21.16	43.64
Gorilla (BM-25)	**40.32**	**4.30**	55.37	17.03	**6.42**	76.55	41.89	**2.77**	55.32
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

Fine Tuning with Retrieval

Table 2: **Understanding the effect of different retrieval techniques used with Gorilla**

	Gorilla without Retriever				Gorilla with Oracle retriever			
	zero-shot	BM25	GPT-Index	Oracle	zero-shot	BM25	GPT-Index	Oracle
Torch Hub (overall) ↑	59.13	37.63	60.21	54.83	0	40.32	61.82	67.20
HuggingFace (overall) ↑	71.68	11.28	28.10	45.58	0	17.04	47.46	91.26
TensorHub (overall) ↑	83.79	34.30	52.40	82.91	0	41.89	64.96	94.16
Torch Hub (Hallu) ↓	6.98	11.29	4.30	15.59	100	4.30	0	0
HuggingFace (Hallu) ↓	10.95	46.46	41.48	52.77	99.67	6.42	8.19	7.08
TensorHub (Hallu) ↓	5.40	20.43	19.70	13.28	100	2.77	2.33	1.89

AST as a Hallucinations Metric

Table 3: **Proposed AST evaluation metric has strong correlation with human evaluation**

	Accuracy
Gorilla AST metric (proposed)	0.78
Eval by Human	0.78
Code Executable (Eval by Human)	0.72

Test-Time Documentation Change

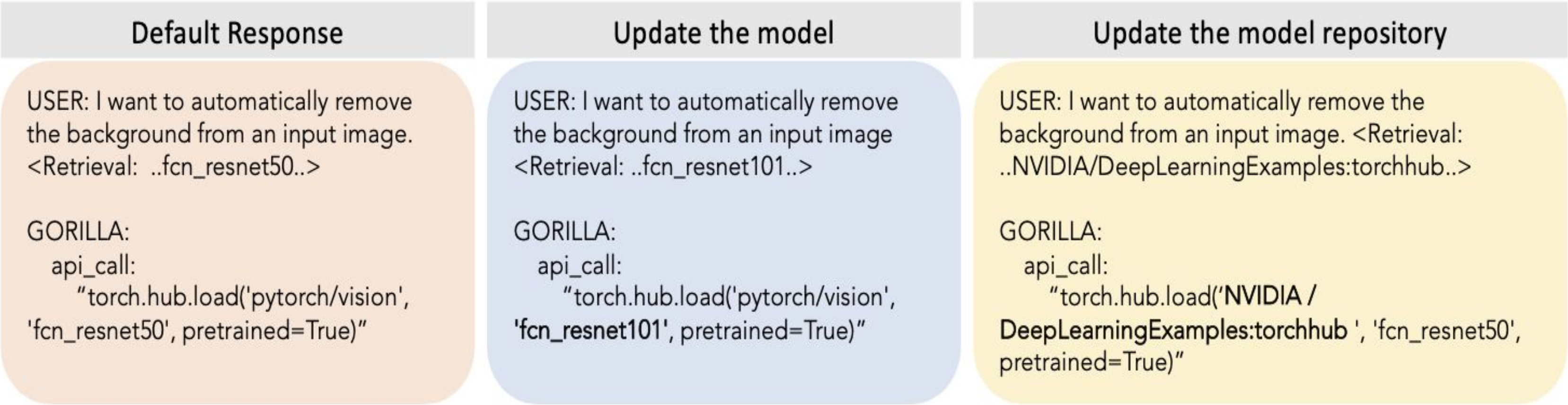


Figure 6: **Gorilla's retriever–aware training enables it to react to changes in the APIs.** The second column demonstrates changes in model upgrading FCN's ResNet–50 backbone to ResNet–101. The third column demonstrate changes in model registry from `pytorch/vision` to `NVIDIA/DeepLearningExamples:torchhub`

API Calls With Constraint

Table 4: **Evaluating LLMs on constraint-aware API invocations**

	GPT-3.5				GPT-4				Gorilla			
	0-shot	BM25	GPT-Index	Oracle	0-shot	BM25	GPT-Index	Oracle	0-shot	BM25	GPT-Index	Oracle
Torch Hub (overall)	**73.94**	62.67	81.69	80.98	62.67	56.33	71.11	69.01	71.83	57.04	71.83	78.16
Torch Hub (Hallu)	19.01	30.98	14.78	14.08	**15.49**	27.46	**14.08**	**9.15**	19.71	39.43	26.05	16.90
Torch Hub (err)	7.04	6.33	3.52	4.92	21.83	16.19	14.78	21.83	8.45	3.52	2.11	4.92
Accuracy const	43.66	**33.80**	**33.09**	69.01	43.66	29.57	29.57	59.15	**47.88**	30.28	26.76	67.60

	LLAMA				Claude			
	0-shot	BM25	GPT-Index	Oracle	0-shot	BM25	GPT-Index	Oracle
Torch Hub (overall)	0	8.45	11.97	19.71	29.92	**81.69**	**82.39**	**81.69**
Torch Hub (Hallu)	100	91.54	88.02	78.87	67.25	**16.19**	15.49	13.38
Torch Hub (err)	0	0	0	1.4	2.81	2.11	2.11	4.92
Accuracy const	0	6.33	3.52	17.60	17.25	29.57	31.69	**69.71**

Fine Tuning Vs Prompting : Gorilla 0-Shot vs GPT 3-Shot

Table 5: **Evaluating Gorilla 0-shot with GPT 3-shot incontext examples**

	HF (Acc ↑)	**HF (Hall ↓)**	**TH (Acc ↑)**	**TH (Hall ↓)**	**TF (Acc ↑)**	**TF (Hall ↓)**
GPT-3.5 (0-shot)	16.81	35.73	41.93	10.75	41.75	47.88
GPT-4 (0-shot)	19.80	37.16	54.30	34.40	18.20	78.65
GPT-3.5 (3 incont)	25.77	32.30	73.11	72.58	71.82	11.09
GPT-4 (3 incont)	26.32	35.84	**75.80**	**13.44**	77.37	11.97
Gorilla (0-shot)	**58.05**	**28.32**	**75.80**	16.12	**83.79**	**5.40**

RAT Robustness

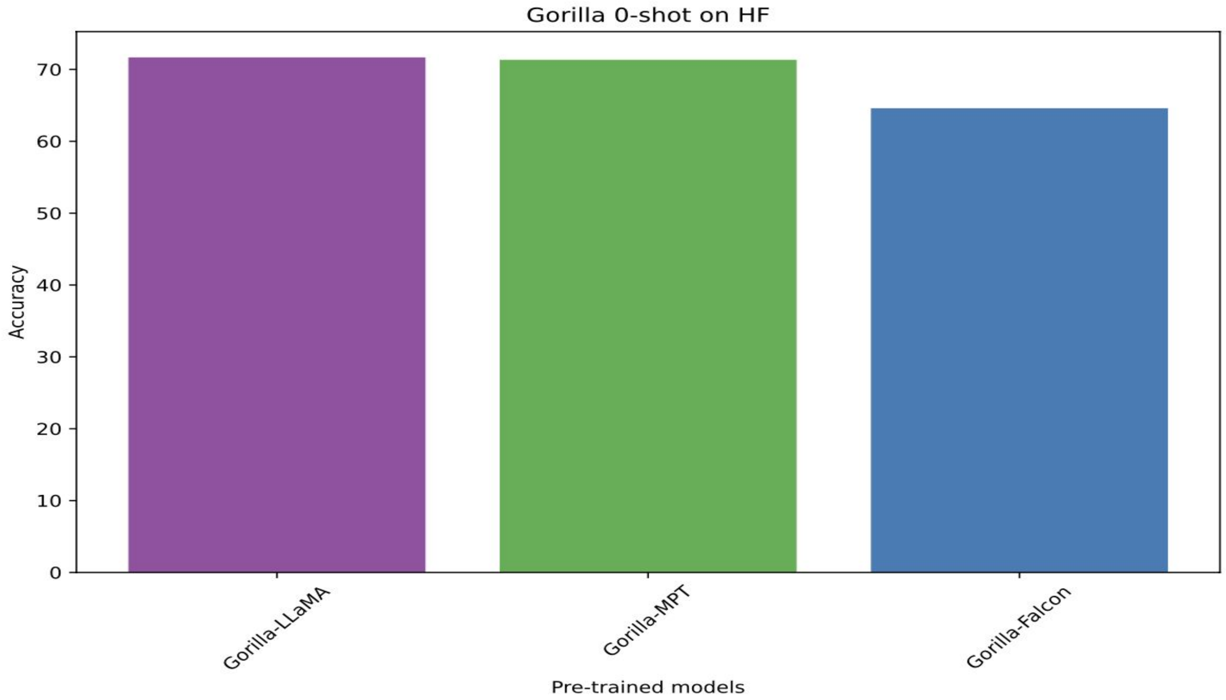


Figure 13: For the same train-eval dataset, our fine-tuning recipe, RAT, is robust to the underlying base model.

Conclusion

Using off-the-shelf LLMs to generate API calls is a challenging task due to their unawareness to what API calls are available and also frequent updation.

Gorilla's fine tuning on APIBench dataset with novel RAT technique enables LLMs to generate accurate API calls , adapting to the frequent updation in the documentations, maintaining its relevance to be able to utilize tools, services, agents exposed through APIs.

Limitations

- Hardware and Runtime Constraint
- Security Concern for Credentials
- Need to provide API to be used in Gorilla OpenFunction
- Works better for single domain prompts
- Dependency on retriever accuracy, works best with oracle retriever

Future Work

- Gorilla CLI , Gorilla OpenFunctions
- Adding capabilities for more service providers like AWS , GCP, Uber etc
- Integrating with plugins
- Addressing Security Concerns
- Handling Multiple domain prompts

Thank you!!