ReAct Meets Industrial IoT: Language Agents for Data Access

Authors: James Rayfield, Shuxin Lin, Nianjun Zhou, Dhaval Patel

EMNLP 2025 Industry Track

https://openreview.net/pdf?id=luETrQw0j6

https://github.com/IBM/ReActXen

"Small Language Models are Future of Agentic AI"

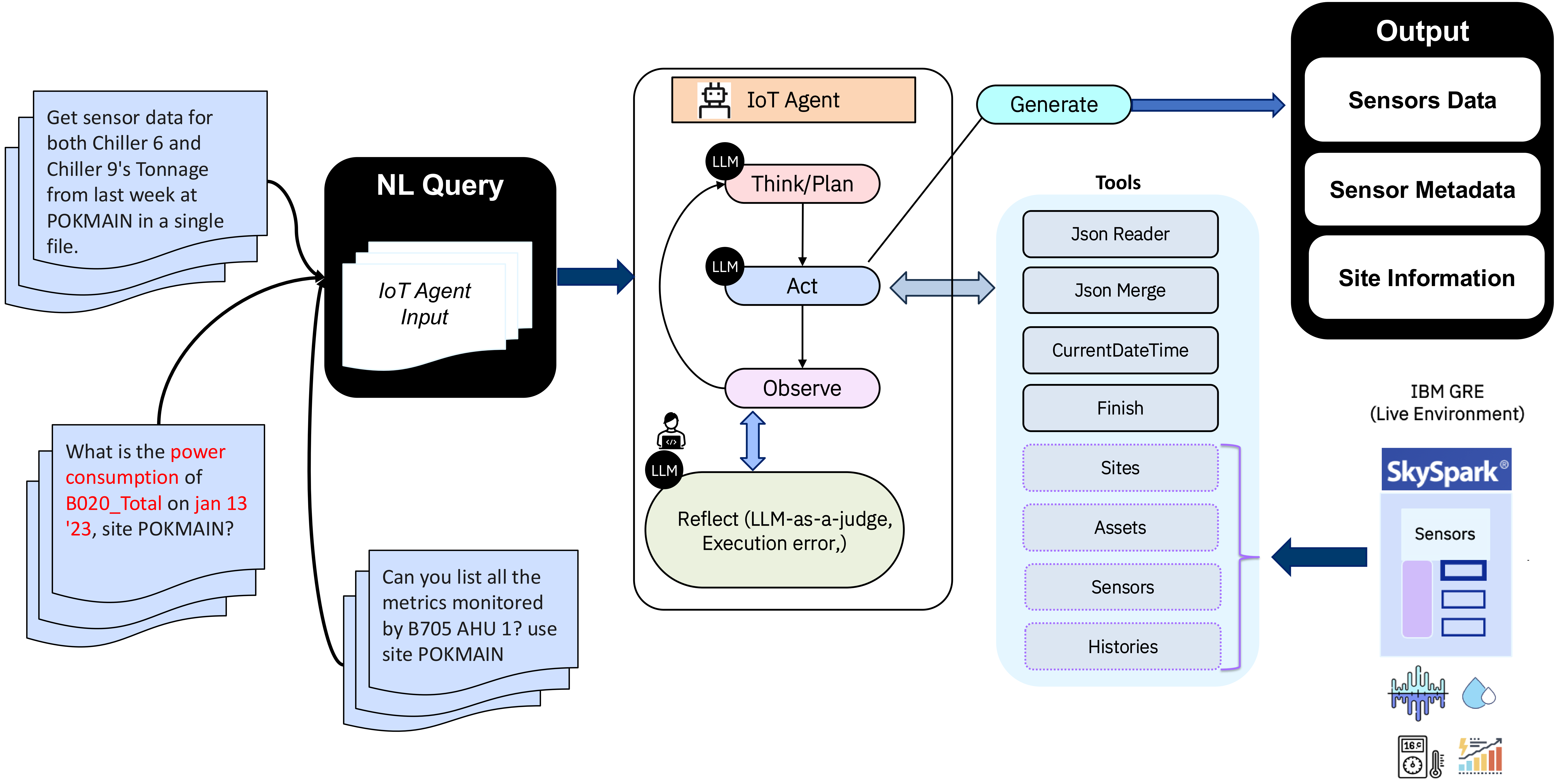
IoT Agent: Content

"LLM does not have awareness about our Domain, Entity, Task, etc. – It's an Agent's Responsibility to fill this Gap"

- ReAct-based IoT Agent Architecture as a Reference example
- Tools Design and Development (Base class is from Langchain)
- Agent Input and Output
- Baseline Result using ReAct (Optimized for Granite/Mistral/…)
- Agent Deficiencies using ReAct
- Proposed AI Agent Runtime Design (Enhancing the core capability)
  - ReActXen - ReAct with Self-Ask, Review, Reflect, Distillation
- Deep Dive of Results
  - Experiment 1: SOTA Comparison – ReAct/CoT/HuggingGPT/ReAct-Review-Reflect
  - Experiment 2: Impact of Reflection
  - Experiment 3: Ability to Self-Learn without any help (i.e., no in-context examples)
- Synthetic Data Generation for Industrial Query
  - Experiment 4: Execution

Architecture: IoT Agent

From NL to Data



Architecture: Tool Descriptions for IoT Agent

From NL to Data

IoT Agent's Tool

Tool	Description	Parameters All strings, dates in ISO 8601
Json Reader	Reads a JSON text file, verifies that it parses correctly, and returns a single line equivalent JSON string	file_name
Json Merge	Reads two JSON text files, parses them, verifies that they are JSON arrays of the same type, merges them, and returns a single-line JSON file	file_name_1, file_name_2
CurrentDateTime	Returns the current time as an ISO 8601 string, and an equivalent English text date and time	none
Finish	Finishes the agent execution and specifies a return string	Return string
Sites	Returns a list of sites from SkySpark	None
Assets	Returns a list of Assets at a Site	site_name
Sensors	Returns a list of sensors for an Asset at a Site	site_name, asset_name
Histories	Returns a list of sensor values for the specified Asset(s) at the specified Site	site_name, asset_name_list, start (date), final (date), sensor_name

IoT Agent : Input and Output

From NL to Data

Get sensor data for both Chiller 6 and Chiller 9's Tonnage from last week at POKMAIN in a single file.

The asset history for sensor Chiller 6 Chiller % Loaded on asset Chiller 6 at POKMAIN site from 2020-06-01T00:00:00-04:00 to 2020-06-30T23:59:59-04:00 has been downloaded and is listed in file /var/folders/fz/l1h7gpv96rv

Can you list all the metrics monitored by B705 AHU 1? use site POKMAIN

found 27 sensors for asset_name B705 AHU 1 and site_name POKMAIN. file_path contains a JSON array of Sensor data"

What is the power consumption of B020_Total on jan 13 '23, site POKMAIN?

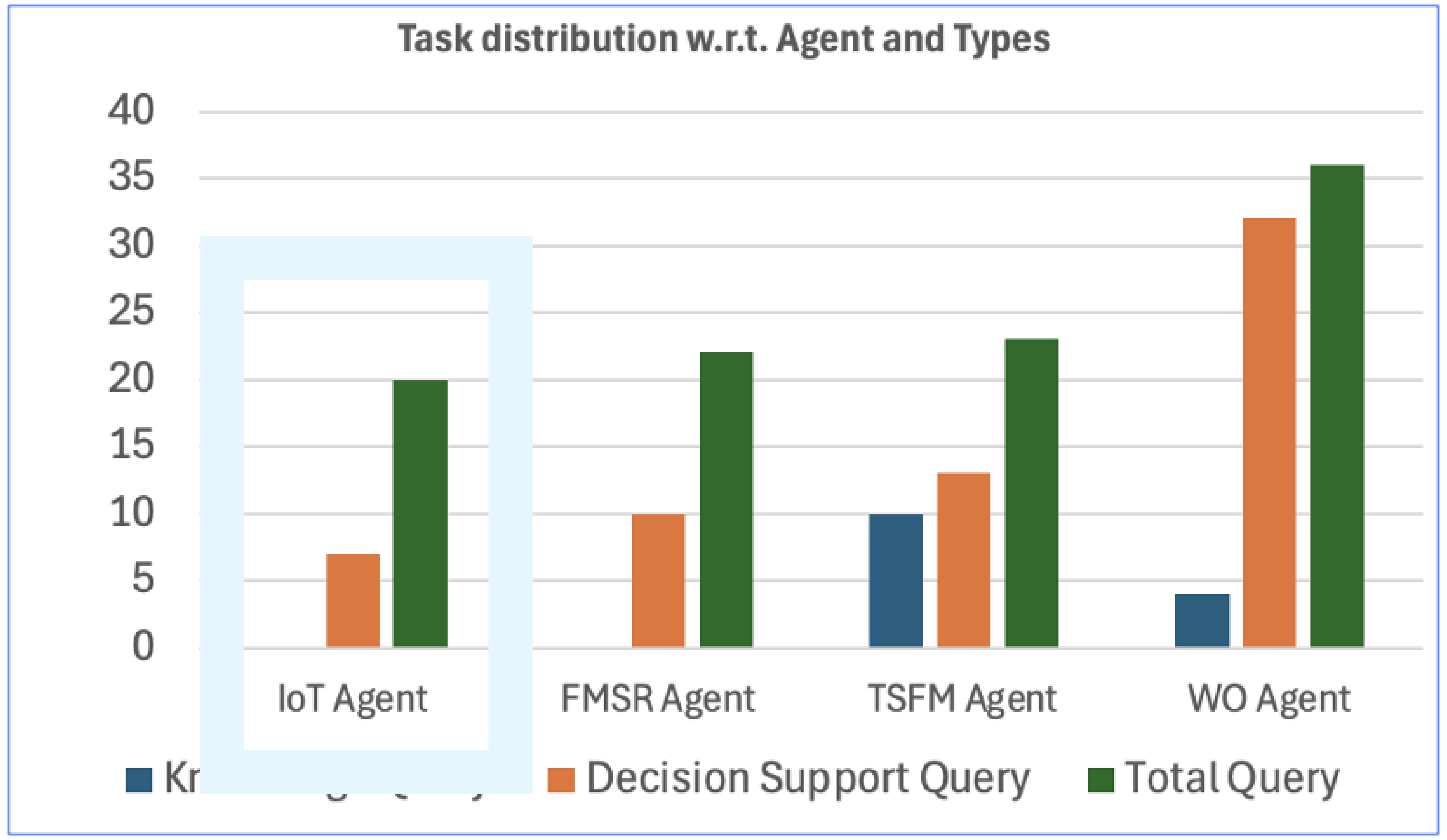
The asset B020 does not exist at site POKMAIN

IoT Agent Evaluation : Baseline

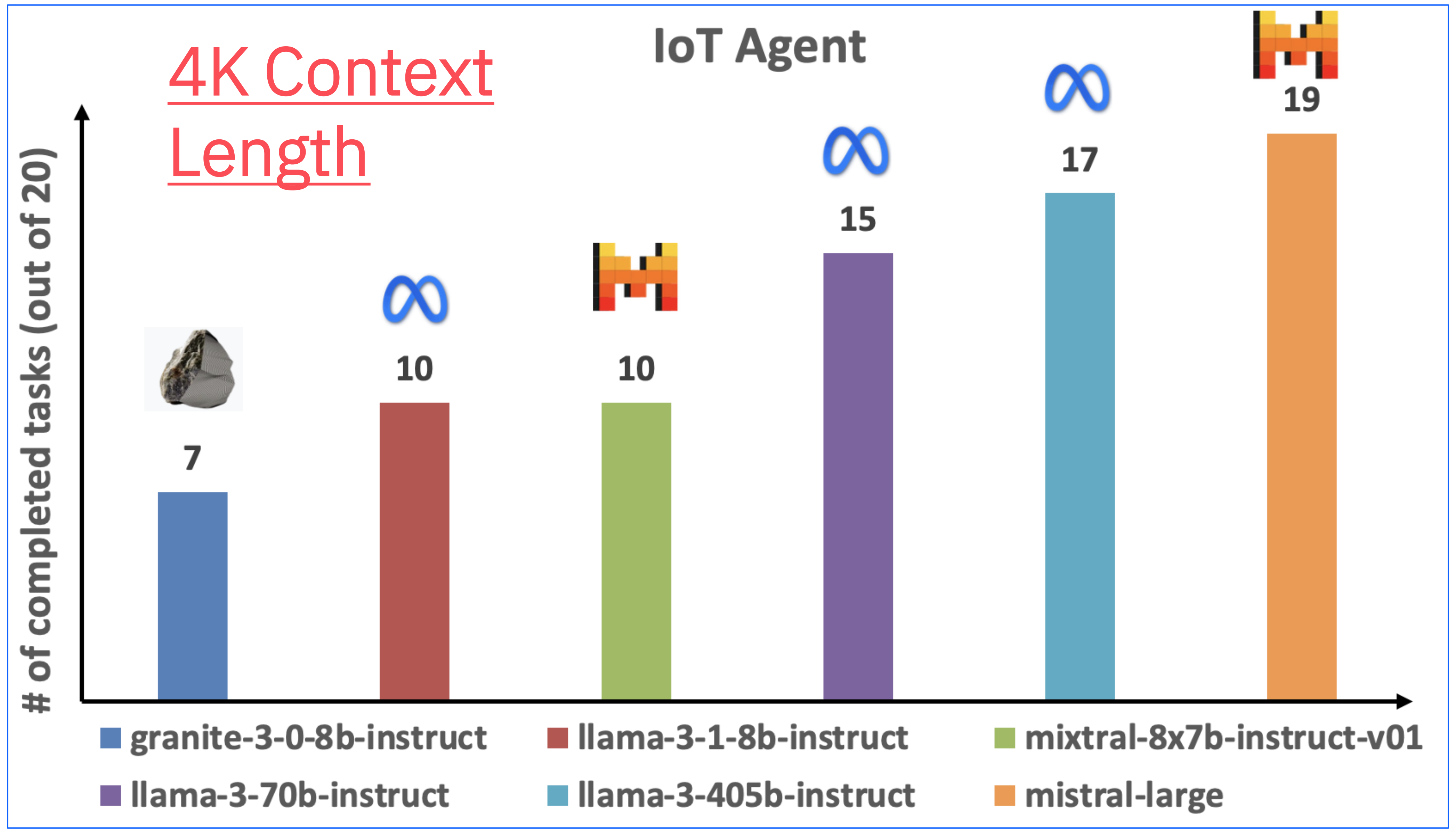
The IoT agent connects to the live IoT system, SkySpark, to retrieve operational data for various time series metrics and KPIs. The agent is tested to download data for three major scenarios:

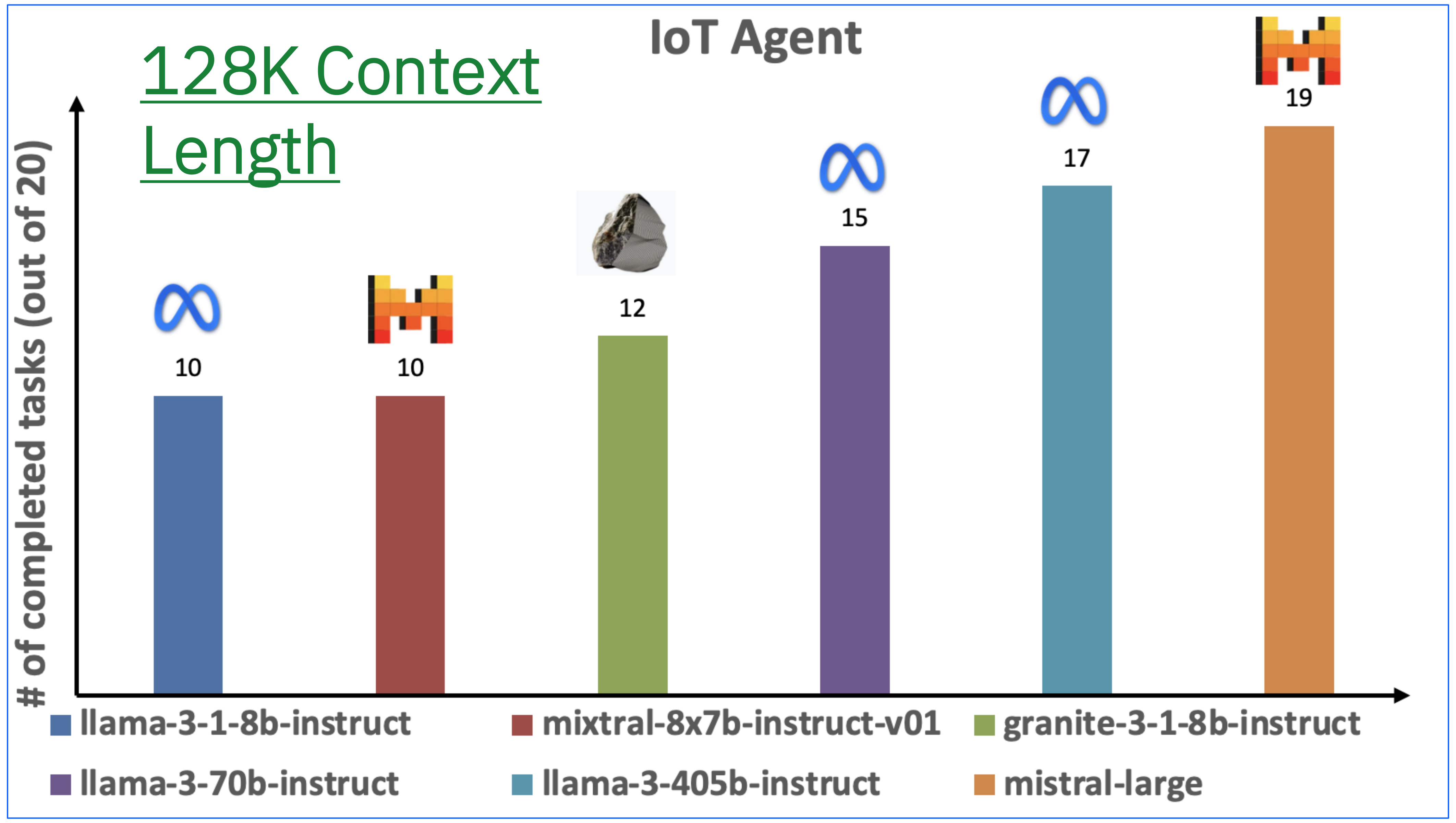
- Sensor : a single sensor or multiple sensors
- Sites : from a single site or multiple sites
- Duration : for a specific day (interval query) or at a particular point in time (point query).

- Two sites (pokmine and RCHmain)



Baseline Results : Granite 3.0 8b vs Granite 3.1 8b using ReAct





Deficiencies of Small LLMs

As shown in the previous slide, we were able to improve the number of tasks from 7 to 12 using longer context length. We investigated the main deficiencies of older granite 3.1 model:

Math Reasoning Issues:
- Struggles with date-offset calculations (e.g., "last week" is never interpreted correctly)

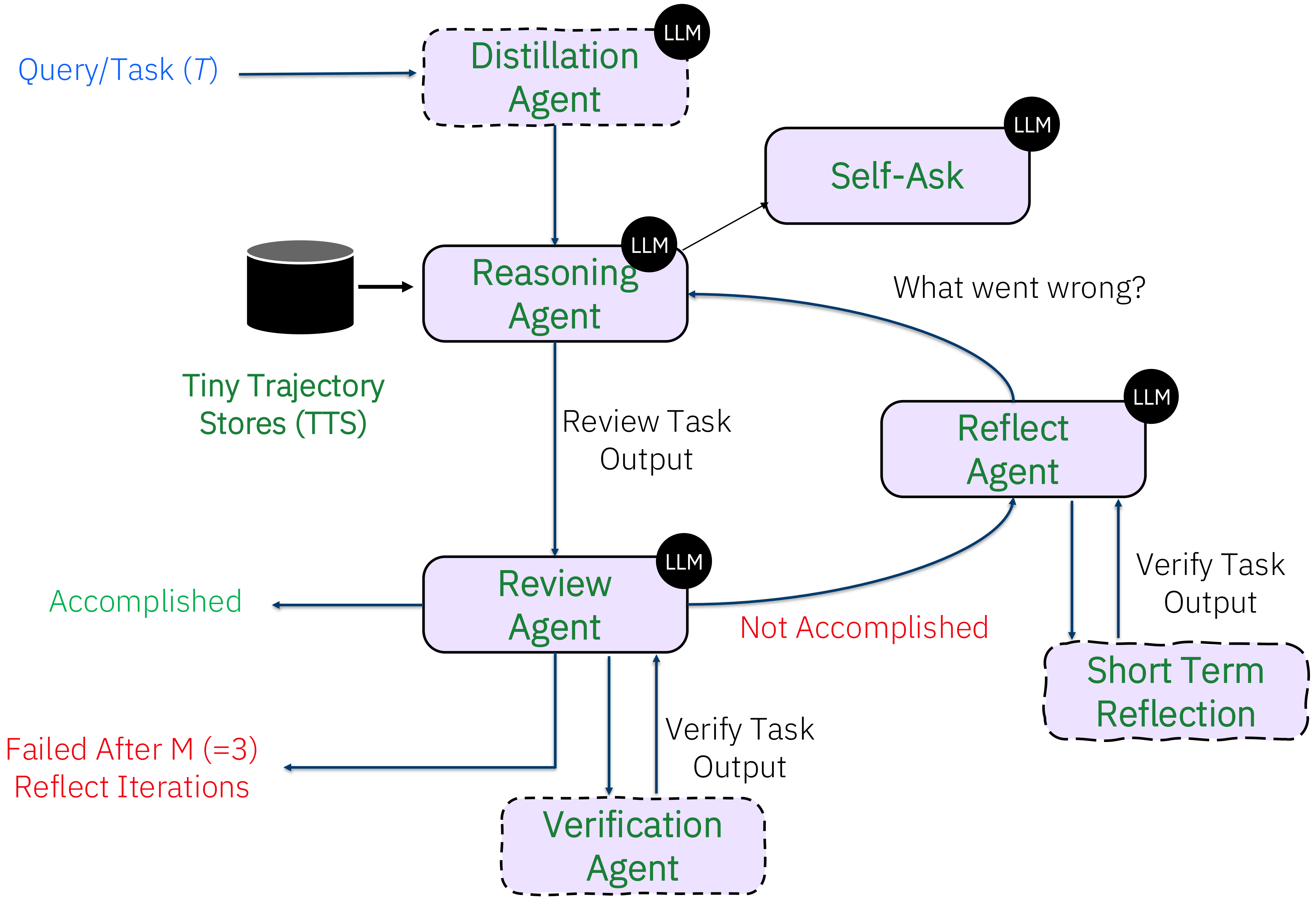
Tool-Related Issues:
- Hallucinates non-existent tools.
- Hallucinates non-existent parameters for tools.

Workflow-Related Issues:
- Does not attempt alternate tools on subsequent passes.
- Struggles to break down complex problems (e.g., questions involving two chillers).

String Scanning Issues:
- If the sensor name does not match exactly, it often fails to identify the correct sensor
- When reading lists (e.g., chillers), only partial list outputs are generated.

Early Quits:
- Declares victory on partial task accomplishment (e.g., "I downloaded the file, and I am done").

Iterative Task Solving with ReAct (with Self-Ask), Review, Reflexion and Distillation Agent



Algorithm 1 Reinforcement via Multiple Verbal Feedback

Initialize ReAct agent M_{react}, Review agent M_{review}, Reflect agent $M_{reflect}$, Distillation agent $M_{distill}$, TTS T
Receive query q
$q' \leftarrow$ expand query q using $M_{distill}$ (optional)
Set $mem \leftarrow [T]$
Set $t \leftarrow 0$
while $t <$ max task reflexion trials **do**
  $ans, traj \leftarrow$ Generate solution for q' using M_{react}
  if ans is **generated then**
    $review_t \leftarrow$ evaluate $\langle q', ans, \tau_t \rangle$ using M_{review}
    if M_{review} **Accomplished/Error then**
      break
    end if
  end if
  Generate **reflection** $reflect_t$ using $M_{reflect}$
  Adjust mem based on $reflect_t$ and $review_t$
  Increment t
end while
Return final solution ans

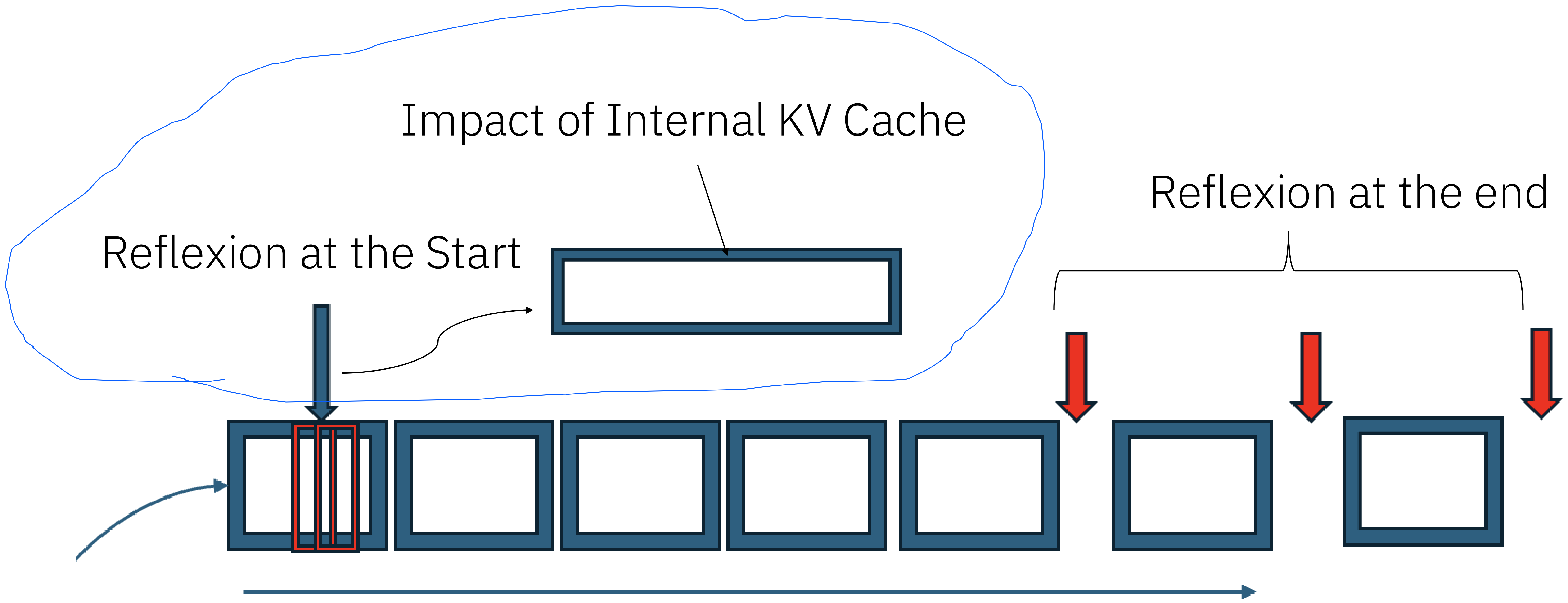
Reasoning Agent can use ReAct/CoT/HuggingGPT(Plan-Execute) Strategy to Solve the problem. ReView and Reflect can work orthogonal to any reasoning strategy.

Justification Why it worked

Internal Working of the iterative feedback to Agent

As shown in the following picture, typical reflexion happen toward end of the scratchpad, whereas our feedback (review/reflect) goes at the start in the conversation. So we created a mixture of feedback at different point in time.

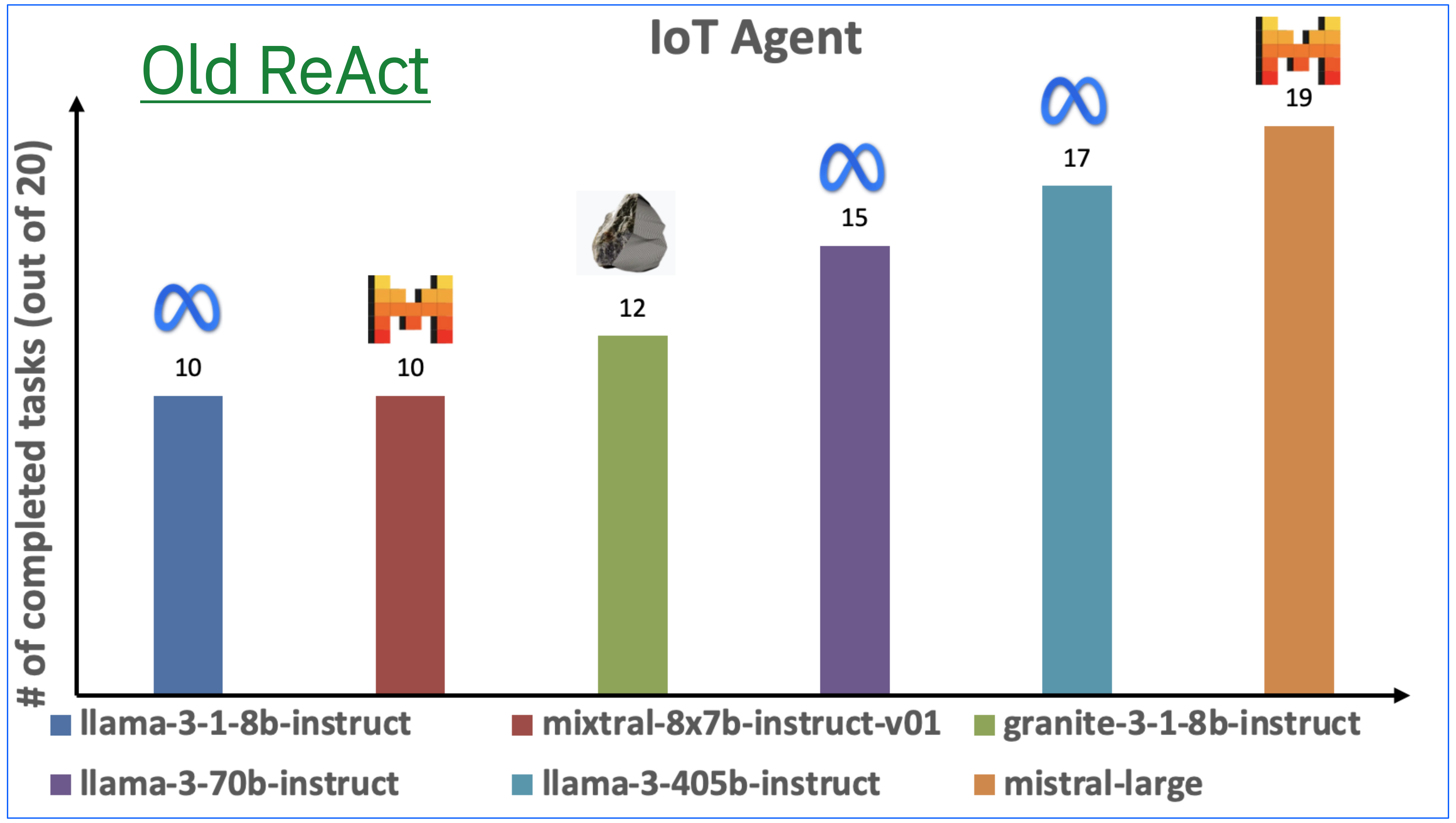
This results into a new "regime" into the LLM's Auto-regressive reaction without perturbation of the temperature parameter

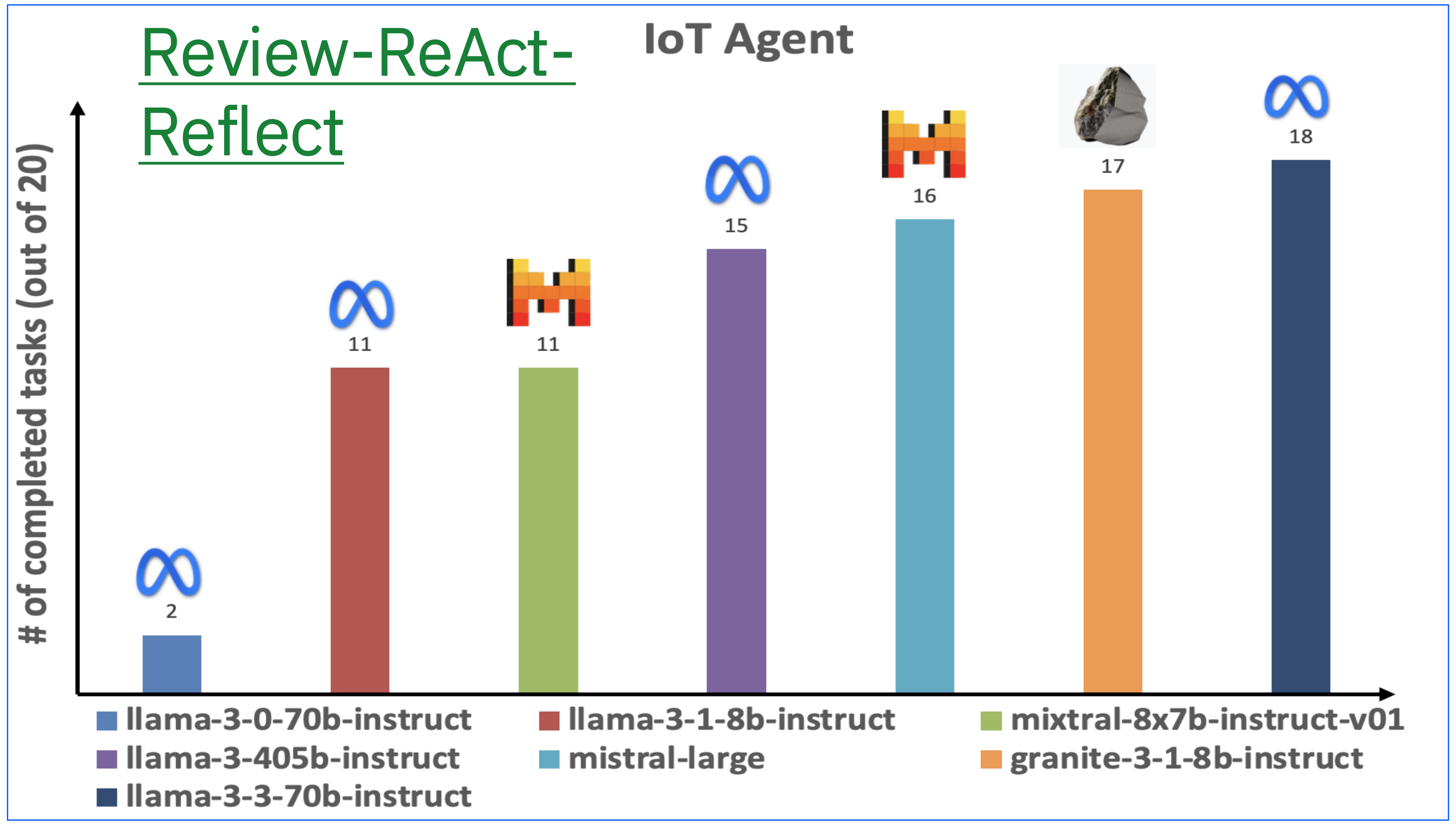


Revisit Experimental Results: IoT Agent Evaluation

After all the innovations wrapped inside our existing ReAct, we can improve the number of tasks from 7 to 12 to 17 using new enhanced ReAct workflow.

Following chart show the comparison of new ReAct. Note that the new ReAct has a new system prompt, few more tools and, as a result, it also has impact on the performance across all the other models.





What we innovated: A Design Pattern for AI Agent

Here is how we overcame granite's deficiencies using our latest framework:

Math Reasoning Related Issue:
- Integrate Self-Ask as part of ReAct to encourage mathematical reasoning alongside other types of reasoning tasks.

Tool Related Issues:
- Transition from few-shot to many-shot examples for tool training using "Tiny Trajectory Samplers."

Workflow Related Issues:
- Monitor action and thought patterns while ReAct executes, detect loops, and provide feedback, with the ability to quit if necessary.
- Introduce task-oriented Reflection and Review Agents to restart the execution, focusing on two key aspects: (i) Why was the task not accomplished? (ii) What went wrong during the process?

String Scanning Related Issues:
- Enhance input queries using a Distillation Agent to bring domain-specific terms into context as early as possible, with the hope of improving disambiguation.

Early Quits
- The Review Agent aims to verify the output of ReAct to avoid task repetition and ensure the task is correctly completed.

Self-Ask with Reasoning based Problem Solver such as ReAct

Self-Ask is a concept integrated into the system prompt of any reasoning-based problem solver. In the iterative process of problem-solving, a language model (LLM) may become stuck by generating repetitive thoughts or may lack the ability to generate a thought that translates into a valid action. To address this, we define specific *trigger points* in the system prompt that alert the LLM to use the Self-Ask process.

The LLM monitors for these trigger points, and when a situation is detected, the Self-Ask component is activated. The LLM then asks itself: *"What do I need to do next?"*

Self-Ask is broken down into three distinct phases, similar to the T-A-O loop used in agent reasoning – assuming it is part of ReAct based reasoning process:

- Think: The LLM realizes it needs to generate a new thought in order to proceed with a particular sub-task, which signals the start of the Self-Ask process.
- Act: The LLM then formulates a question it needs to ask itself. It collects all relevant information from its recent history so that the question is fully answerable.
- Observe: The LLM treats this process as a kind of internal query. It performs the Self-Ask action, using the entire scratchpad to get the answer. The outcome of this query may involve retrieving external knowledge, resolving temporal ambiguity, or other reasoning tasks.

Self-Ask with Reasoning based Problem Solver such as ReAct

The output generated during the Self-Ask phase is appended to the overall thought process, continuing the LLM's problem-solving loop.

In summary, the **Think-Act-Observe** process of Self-Ask is entirely self-managed by the LLM. If Reasoning process is CoT or HuggingGPT, Self-Ask can be adapted to adjust the underlying Reasoning Method. The same concept can be extended to create other self-ABC (Action, Behaviour, Context) tools within the system prompt.

Additionally, this idea can be applied to a **Finish** phase, where the LLM self-asks whether a particular task or question has been completed satisfactorily.

System Prompt of Any Reasoning Strategy such as ReAct

....

....

```
## Here are Guidance on Tool Usage:
### 1. Default to Self-Ask When Tools Are Missing: Fallback to Self-Ask when tools are unavailable, using logic
### 2. Prioritize Step-by-Step Explanations: Provide step-by-step explanations to ensure clarity and transparer
### 3. Fallback for Common Operations: Manually solve common operations like arithmetic, string manipulations,
### 4. Clearly Identify the Steps: Explicitly state when reasoning is used and solve problems step-by-step.
```

Self-Ask : (Short) Example to Show Math Reasoning

- Question: how do I calculate the start date for last week or past week?
- Thought 1: I should first obtain the current date and then subtract 7 days from the current date.
- Action 1: currentdatetime
- Action Input 1: {}
- Observation 1: {"currentDateTime": "2025-01-08T19:40:04.664816+00:00", "currentDateTimeDescription": "Today's date is 2025-01-08 and time is 19:40:04."}
- Thought 2: Now, I have the current date and I should proceed to generate the start date for last week.
- Action 2: **Self-Ask**
- Action Input 2: what is the start date for last week if current date is 2025-01-08T19:40:04.664816+00:00
- Observation 2: based on current date, the last week starts on 2025-01-01T00:00:00.000000+00:00
- Thought 3: Now I have the start date for last week
- Action 3: Finish
- Action Input 3: the start date for last week is 2025-01-01T00:00:00.000000+00:00.

Review Agent

The Review Agent plays a crucial role in evaluating the output produced by a ReAct Agent, categorizing the result into three categories: Accomplished, Not Accomplished, and Partially Accomplished.

The Review Agent makes this decision based on the
- input query (\(\mathcal{Q}\)),
- the final answer generated by the ReAct(or any) Agent,
- and the trajectory (plan) followed to generate the answer.

Currently, we employ an LLM-based approach to build a zero-shot evaluation model, but a classification model could also be used to provide similar feedback once the agent has collected a sufficient training dataset.

Let:

- $\mathcal{Q}$ = input query
- A_{final} = final answer generated by the ReAct Agent
- $T_{\text{trajectory}}$ = trajectory followed to generate A_{final}
- S_{feedback} = feedback provided by the Review Agent
- C_{status} = completion status (Accomplished, Not Accomplished, Partially Accomplished)

We can express the evaluation and feedback process as:

$$C_{\text{status}} = \text{ReviewAgent}(\mathcal{Q}, A_{\text{final}}, T_{\text{trajectory}})$$

Where:

- If $C_{\text{status}} = \text{Accomplished}$, the task is considered successfully completed.
- If $C_{\text{status}} = \text{Partially Accomplished}$, the agent has made some progress, but more work is needed.
- If $C_{\text{status}} = \text{Not Accomplished}$, the task is not completed, and feedback S_{feedback} is provided.

The feedback is used by the ReAct Agent to adjust its actions in the next iteration:

$$S_{\text{feedback}} = \text{GenerateFeedback}(\mathcal{Q}, A_{\text{final}}, T_{\text{trajectory}}, C_{\text{status}})$$

Review Agent

If the task is not accomplished, the Review Agent provides verbal guidance or suggestions on why the task was not considered complete.

Note that the Review Agent is invoked only when the ReAct Agent declares a victory, i.e., when the task is marked as completed.

This feedback is then used by the ReAct Agent in subsequent iterations to improve its performance. We continue the review process until the task is deemed finished.

```
 Review Agent Feedback: {'status': 'Accomplished', 'reasoning': "The agent successfully identified and corrected errors in the input format an
d sensor names. It retrieved the correct sensor data for both Chiller 6 and Chiller 9's Tonnage from last week at POKMAIN and merged the data i
nto a single file. The final response provides a clear and accurate outcome with the location of the merged file.", 'suggestions': ''}
```

Distill Agent

In normal scenario, Agent act on the input question, however, it may be possible that input question can be further simplified and contextualized with domain specific information that help the Agent's Reasoning process to improve.

<u>One such example is :</u>

- Domain Specific Named entity (Asset, sites, and etc)
- Date time decipher (last week, quarter one, etc)

The task of Distill agent is to look at the input question and few domain-specific in-context example to extract various vital information and then these information should be passed as an additional content to the Agent for solution:

Task: Here is an {Input Question} with {Distilled Information}

Reflect Agent

There are three cases when an external reflexion is needed:

- Agent is exhausted
- Review Agent is “Partially Accomplished” or
- Agent is in “Not Accomplished”

The Reflexion agent focus on a specific task of providing feedback on what went wrong and is based on input question, trajectory and final answer. The feedback of reflect agent is clubbed together with the Review Agent for generating the a revised system prompt.

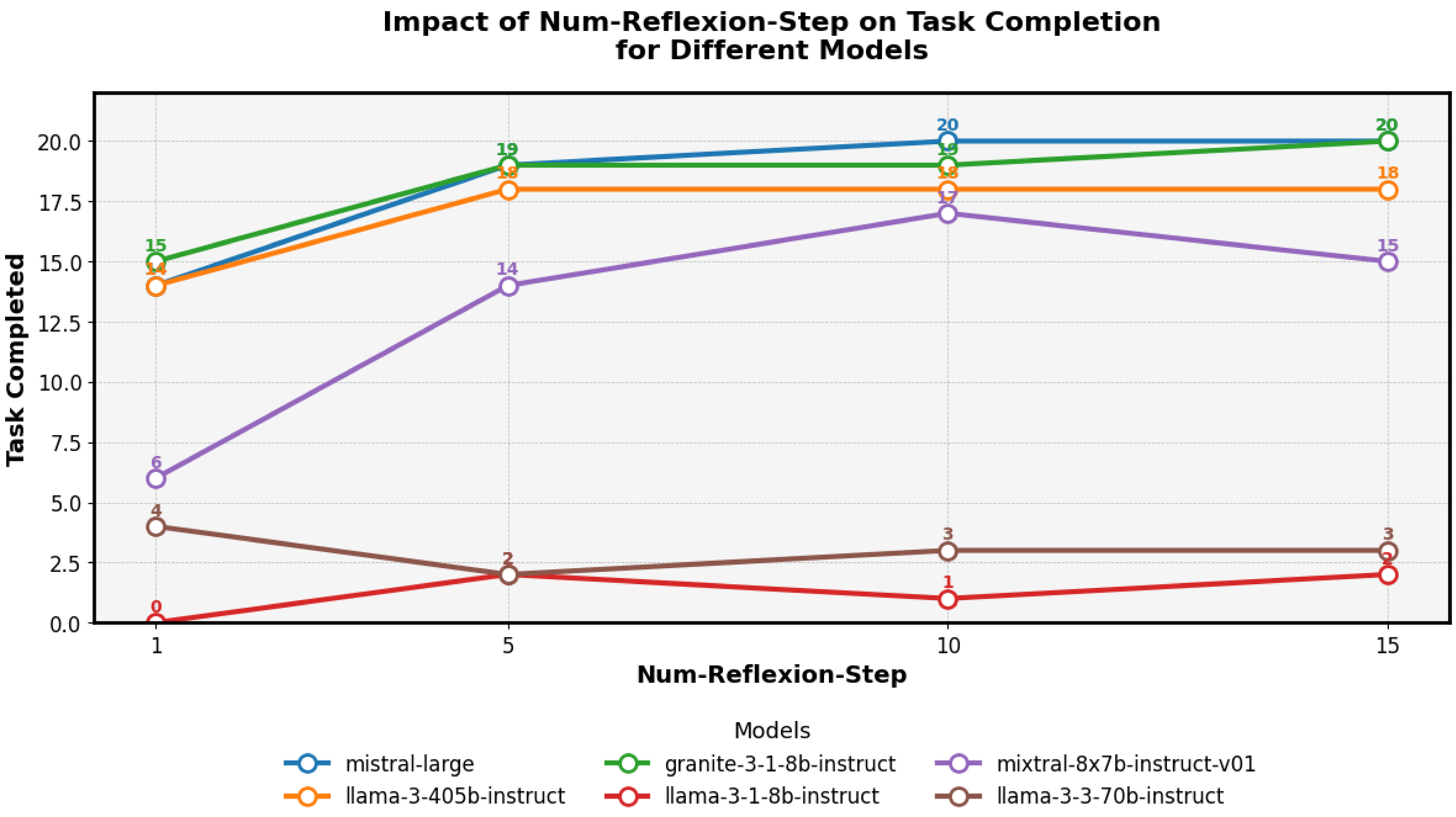
```
Observation 15: Error: Execution failed while calling the Tool. Details: equip and siteRef->campusId == "Site 14"
Process is completed now
  Task Execution Status (Finished): False
  Reflect Agent Feedback : I failed to retrieve the asset data for each site due to repeated errors. I should have checked if the site names we
re correctly formatted and if the tool was functioning properly.
```

Iterative Task Solving with ReAct, Review and Reflexion with Self-Ask

Component	Purpose
ReAct Agent	*Focus on Think-Act-Observe style of problem solving; with loop detection ability (thought repeat/action repeat), and Text/Code Generation based on Action Style*
Reflect Agent	*Focus on unsuccessful task and provide guidance on what went wrong*
Review Agent	*Focus on cross verifying the ReAct agent output such as has it completed the task or not?*
Self-Ask	*An inbuilt LLM based tool in ReAct to enhance a thought process by asking a question to itself such as* "what is the start date for last week if current date is 2025-01-23T05:04:33.720311+00:00"

Revisit Experimental Results: IoT Agent Evaluation

Experiment 1: We have varied number of reflexion rounds from 1 to 15 to demonstrate the ability of Reflect Agent to solve more tasks.



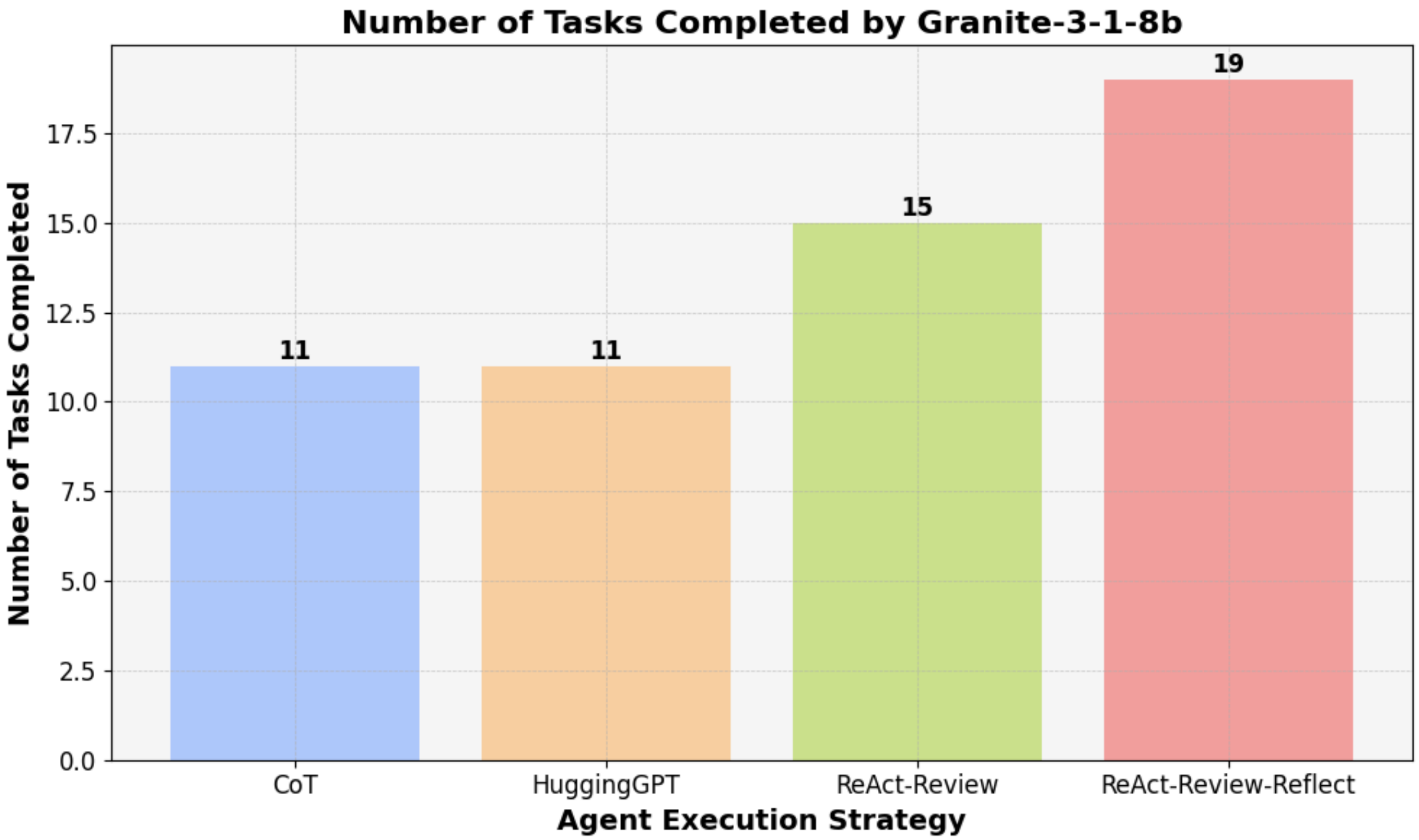
Revisit Experimental Results: IoT Agent Evaluation

Experiment 2: Comparison with out Agent Execution Strategy.

We have four Agents to compares the solutions

Baseline Agents
- Chain of Thought (CoT)
- Plan-Execute with Reflexion (HuggingGPT)

Our Agent with Self-Ask
- ReAct-Review
- ReAct-Review-Reflect (set to 5 reflections)



CoT and Plan-Execute mostly failed during the Planning Phase, as some logical queries require the LLM's ability to think directly, rather than relying on an external tool.

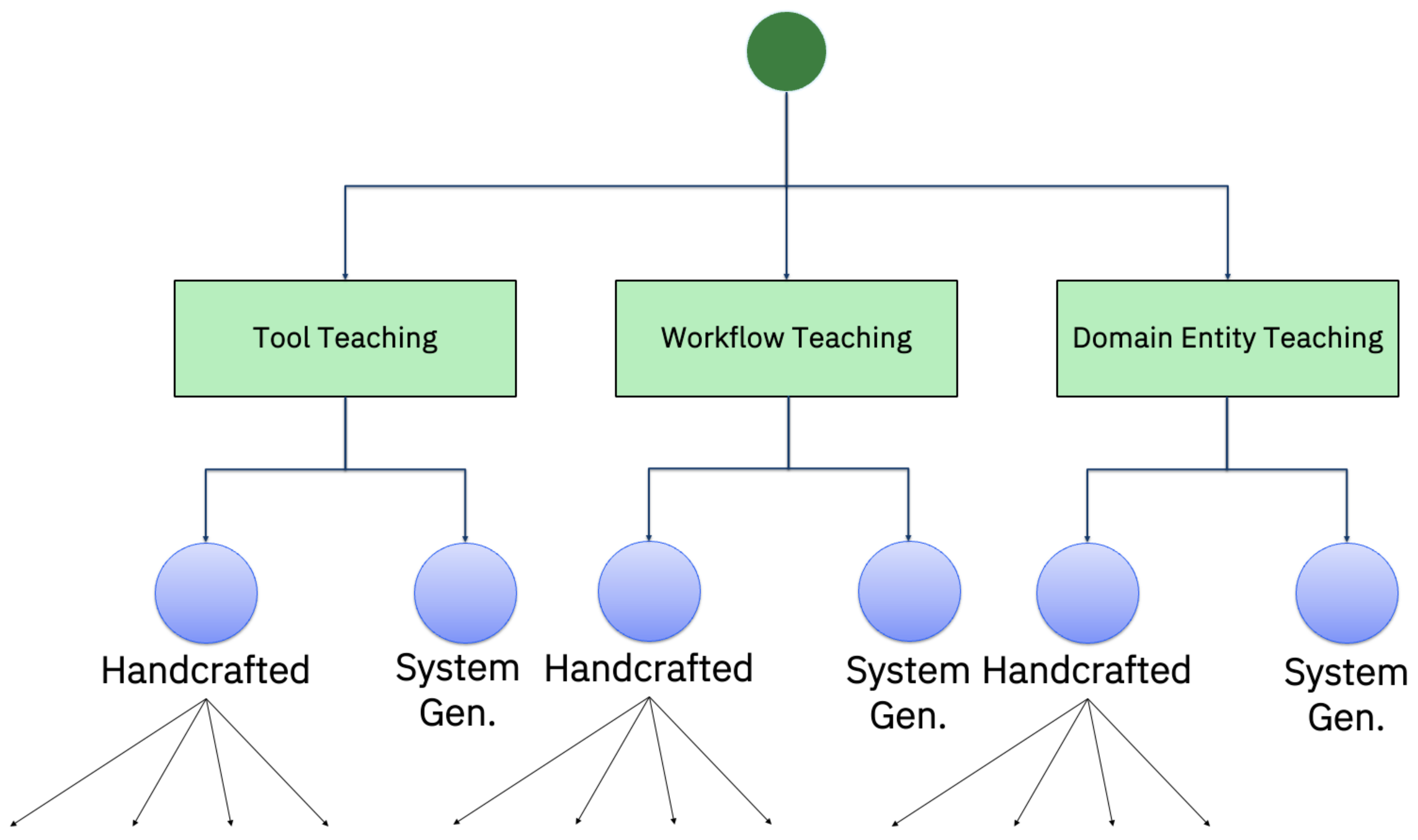
TTS: Tiny Trajectory Samples

We have prepared a hand-crafted examples for many-shot in-context learning
Our Examples are organized by three categories:

- [Tool Teaching]: How to use complex tools in two steps
- [Entity Teaching]: Describe how a core entity are represented in the system
- [Workflow Teaching]: How to solve complex workflow in two-three steps by providing partial trajectories

We call the collections as “Tiny” as we limited the number of steps in each trajectory to 1-3 steps at max

In-context Learning Examples



One Example of “Entity Teaching”

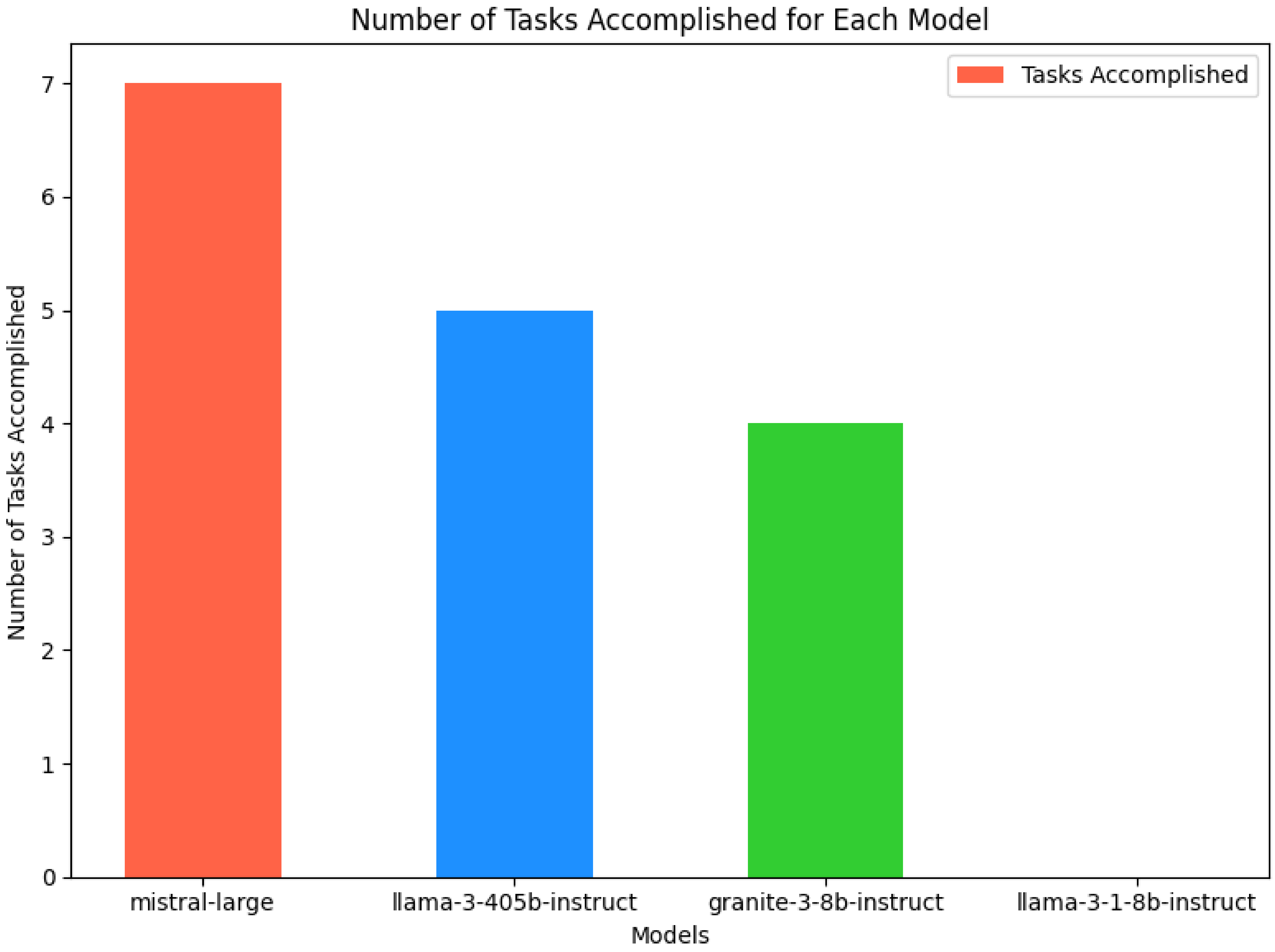
Question: How are assets represented? Give five examples.

....
....   Think – Act – Observe steps
....

Assets are represented with a site_name and an asset_name. Here are five examples: 1. {"site_name": "POKMAIN", "asset_name": "B706_LC26"} 2. {"site_name": "POKMAIN", "asset_name": "B012 SCWP 1"} 3. {"site_name": "POKMAIN", "asset_name": "B710 FCU 1-39"} 4. {"site_name": "POKMAIN", "asset_name": "B006_LC74"} 5. {"site_name": "POKMAIN", "asset_name": "B006A AHU 2"}

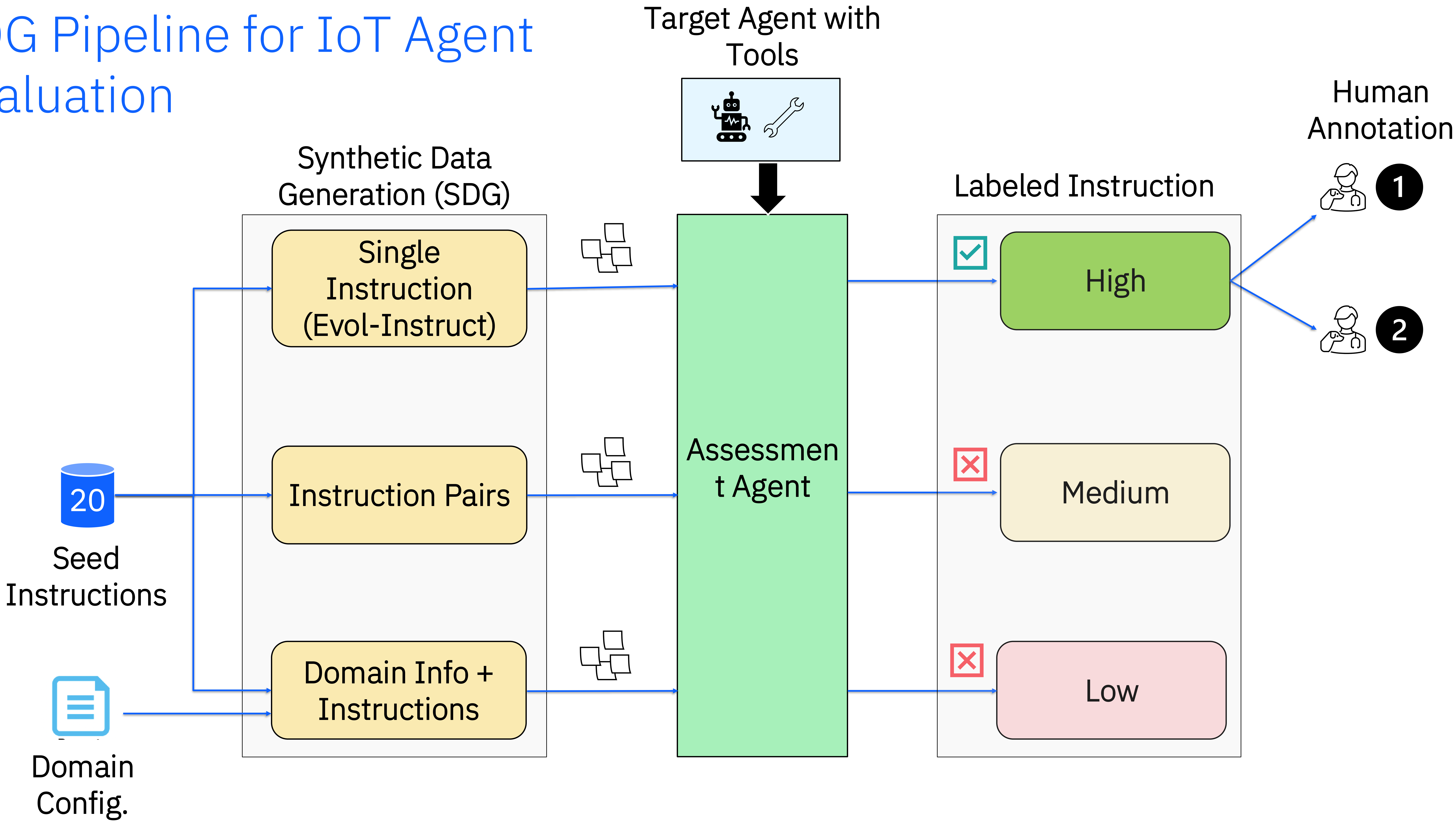
Revisit Experimental Results: IoT Agent Evaluation

Experiment 3: Remove all the in-context examples and Let Agent to solve the problem. What if an agent is given a task to solve but no hint. Its like an autonomous experience for agents to learn skills.



Enhance task decomposition for all models: Especially for models like Granite-3-8b-instruct, improving their ability to break down complex tasks into smaller sub-tasks could help increase the number of tasks they can complete.

SDG Pipeline for IoT Agent Evaluation



SDG Pipeline for IoT Agent Evaluation

ModelName	Tasks Completed @Round 1	Tasks Completed @Round-End	Tokens Sent Avg (Std)	Tokens Rece. Avg (Std)	API Calls Avg (Std)	Proc. Time Avg (Std)	Traj. Length Avg (Std)	Reflection Avg (Std)
Experiment 1: Internal Reviewer (Same LLM for Review-Reflect-ReAct								
mistral-large	101	161	337233.68 (1006062.27)	1854.19 (4278.63)	32.01 (73.55)	3.22 (7.44)	7.18 (2.45)	2.15 (3.15)
granite-3-8b-instruct	88	149	601134.13 (1360993.65)	2444.98 (4424.45)	48.88 (89.89)	5.91 (10.83)	5.78 (2.60)	3.77 (5.51)
Experiment 2: External Reviewer (Llama-405b for Review Agent)								
mistral-large	121	155	270053.28 (989989.62)	1394.52 (3648.70)	25.06 (67.39)	2.69 (6.92)	6.68 (2.39)	1.71 (2.50)
granite-3-8b-instruct	90	140	642367.70 (1495517.11)	2720.87 (5083.58)	54.65 (96.53)	6.62 (11.89)	5.73 (2.60)	4.09 (5.83)

Table 3: Summary of Model Metrics with Statistics $\mathcal{D}_{\text{Syn-I}}$

Key Difference between Open-Source Agents (CrewAI/Langchain)

- ReAct System prompt is **extended** with additional guidance on (Self-Ask, Review-Reflect Feedback, etc)

- The Concept of Review Agent with Zero-shot is missing into open-source code repository.
  - This project demonstrated the LLM can also served/act as a review of ReAct output for generating appropriate output, and that review when in-corporated in second round of execution helped to improve the performance.

- This framework can be used to access agent (underlying LLM's ability) cognitive ability to adapt the situation based on feedback, which is missing

- The "Self-consistency" based approach led to a Service Denial Situation when operating in a real live env. How Agent can work in such scenario using Sequential Feedback Mechanism is a key and not discussed in any of the open source

- We can always create a sequential process, however, what task to be conducted in that sequential flow as well as how to adjust the system prompt and what conditional logic to be implemented which can result into "Robust Framework" is missing

Related Work

- Agarwal, R., Singh, A., Zhang, L. M., Bohnet, B., Rosias, L., Chan, S., Zhang, B., Anand, A., Abbas, Z., Nova, A., Co-Reyes, J. D., Chu, E., Behbahani, F., Faust, A., & Larochelle, H. (2024). *Many-shot in-context learning*. arXiv:2404.11018.
- Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y. C., & Molchanov, P. (2025). *Small language models are the future of agentic AI.* arXiv:2506.02153.
- OpenCompass Contributors. (2023). *OpenCompass: A universal evaluation platform for foundation models.* GitHub: https://github.com/open-compass/opencompass
- Dherin, B., Munn, M., Mazzawi, H., Wunder, M., & Gonzalvo, J. (2025). *Learning without training: The implicit dynamics of in-context learning.* arXiv:2507.16003.
- IBM Granite Team. (2024). *Granite 3.0 language models.* IBM Research.
- Jabbour, J., & Reddi, V. J. (2024). *Generative AI agents in autonomous machines: A safety perspective.*arXiv:2410.15489.
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., & El Sayed, W. (2023). *Mistral 7B.* arXiv:2310.06825.
- Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2023). *SWE-Bench: Can language models resolve real-world GitHub issues?* arXiv:2310.06770.
- LangChain. (2024a). *LangChain tools documentation.* Accessed: 2025-02-09.
- LangChain. (2024b). *Wikipedia tool integration.* Accessed: 2025-02-09.

Related Work

- Liu, Z., Hu, H., Zhang, S., Guo, H., Ke, S., Liu, B., & Wang, Z. (2024). *Reason for future, act for now: A principled architecture for autonomous LLM agents. Proceedings of the 41st International Conference on Machine Learning* (Vol. 235, pp. 31186–31261). PMLR.
- Nikitin, A., & Kaski, S. (2022). *Human-in-the-loop large-scale predictive maintenance of workstations. KDD '22 Proceedings*, 3682–3690.
- OpenAI. (2024). *o1-preview.*
- Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., & Lewis, M. (2023). *Measuring and narrowing the compositionality gap in language models.* arXiv:2210.03350.
- Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). *HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face.* arXiv:2303.17580.
- Shen, Y., Song, K., Tan, X., Zhang, W., Ren, K., Yuan, S., Lu, W., Li, D., & Zhuang, Y. (2024). *TaskBench: Benchmarking large language models for task automation.* arXiv:2311.18760.
- Shetty, M., Chen, Y., Somashekar, G., Ma, M., Simmhan, Y., Zhang, X., Mace, J., Vandevoorde, D., Las-Casas, P., Gupta, S. M., Nath, S., Bansal, C., & Rajmohan, S. (2024). *Building AI agents for autonomous clouds: Challenges and design principles.* arXiv:2407.12165.
- Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., & Yao, S. (2023). *Reflexion: Language agents with verbal reinforcement learning.* arXiv:2303.11366.
- SkyFoundry. (2024). *SkySpark.* https://www.skyfoundry.com/product (Accessed: 2024-09-02).

Related Work

- Tao, W., Zhou, Y., Wang, Y., Zhang, W., Zhang, H., & Cheng, Y. (2024). *MAGIS: LLM-based multi-agent framework for GitHub issue resolution.* arXiv:2403.17927.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., et al. (2023). *LLaMA 2: Open foundation and fine-tuned chat models.* arXiv:2307.09288.
- Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., Pan, J., Song, Y., Li, B., Singh, J., et al. (2024). *OpenHands: An open platform for AI software developers as generalist agents.* arXiv:2407.16741.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). *Chain-of-thought prompting elicits reasoning in large language models.* arXiv:2201.11903.
- Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., & Jiang, D. (2023). *WizardLM: Empowering large language models to follow complex instructions.* arXiv:2304.12244.
- Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., Lin, Q., & Jiang, D. (2024). *WizardLM: Empowering large pre-trained language models to follow complex instructions. In Proceedings of the 12th International Conference on Learning Representations (ICLR).*
- Yang, H., LaBella, A., & Desell, T. (2022). *Predictive maintenance for general aviation using convolutional transformers.* arXiv:2110.03757.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). *ReAct: Synergizing reasoning and acting in language models. arXiv:2210.03629.*