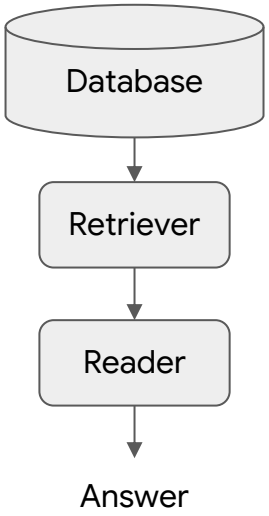
DevRev's AI Agent 007

Tooling up for Success

Team 31

Tool - Based Question Answering

Typical Q&A pipeline



Tool based Q&A pipeline additionally requires -

1. Query analysis
2. Tools selection
3. Specifying tool arguments
4. Tool execution sequencing

Existing Work

Tool-based LLMs

ToolLlama
ReAct
Chameleon
ART
Toolformer

Nexus
ChatCoT
GEAR

ToolBench
APIBank

APIBench

Existing Work

Tool-based LLMs

ToolLlama
ReAct
Chameleon
ART
Toolformer
TALM
ToolAlpaca
ToolDec
Nexus
ChatCoT
GEAR
Gorilla

Datasets

ToolBench
APIBank
APIBench

Existing Work

Tool-based LLMs

ToolLlama Nexus
ReAct
Chameleon ChatCoT
ART
Toolformer GEAR
TALM
ToolAlpaca Gorilla
ToolDec

Datasets

ToolBench APIBench
APIBank

Innovative explorations (including ours!)

1. ControlLLM Reflexion
   EXPEL Multi-agent
2. a. Prompting methods: Analogical, Step-back...
   b. JSON-Python-Typescript
   c. Finetune Mistral, Vicuna, OpenChat ...
3. **a. Reasoning viA Planning (RAP)**
   b. LLM enforcer

Existing Work

Tool-based LLMs

ToolLlama Nexus

ReAct

Chameleon ChatCoT

ART

Toolformer GEAR

TALM

ToolAlpaca Gorilla

ToolDec

Datasets

Innovative explorations (including ours!)

1. ControlLLM Reflexion

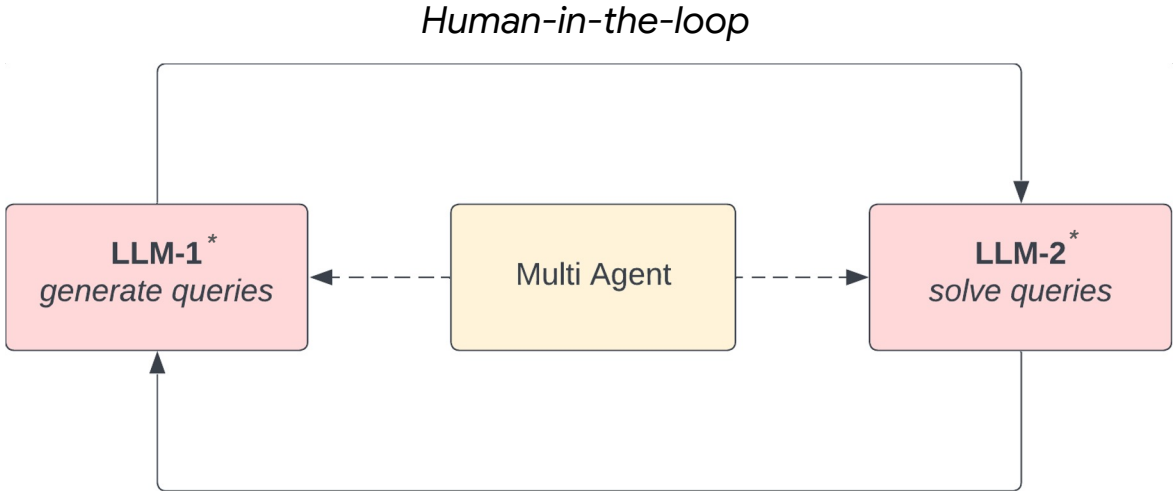
   EXPEL Multi-agent

2. a. Prompting methods: Analogical, Step-back...
   b. JSON-Python-Typescript
   c. Finetune Mistral, Vicuna, OpenChat ...

3. **a. Reasoning viA Planning (RAP)**
   b. LLM enforcer

- Most existing solutions don't generalise on unseen tools
- Hallucinations in arguments is a big issue

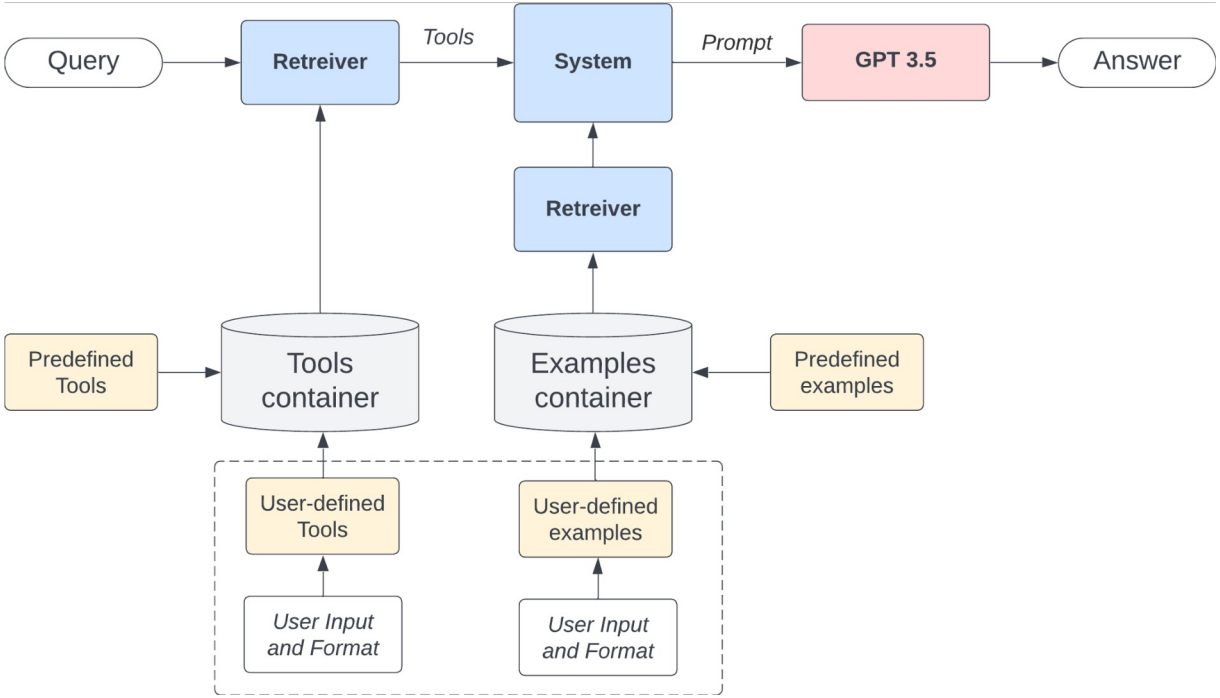
Data Generation



Our golden dataset: ~70 high quality QA pairs & new tools

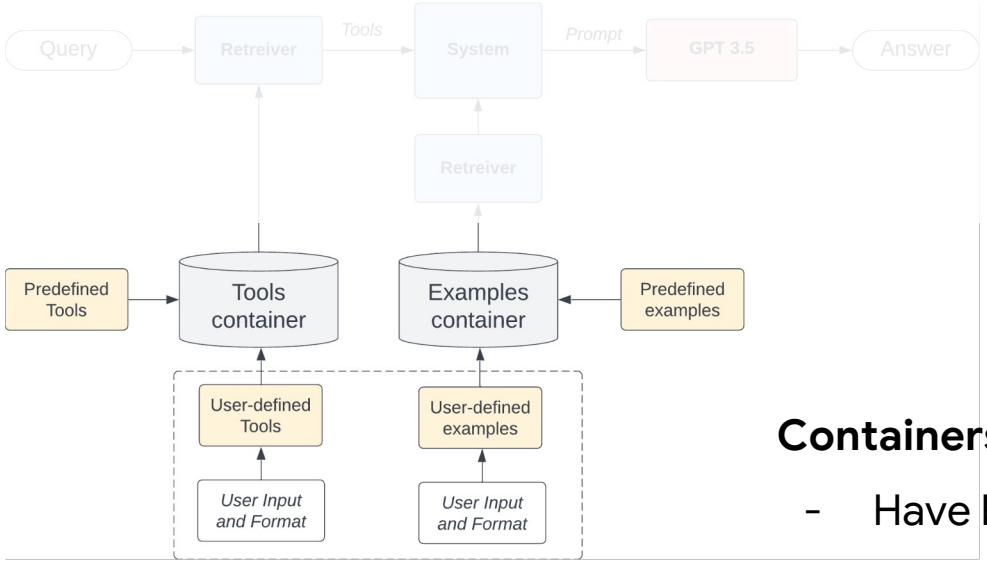
**GPT-4 with ControlLLM or RAP*

Main Solution: RE-GAINS



RE-GAINS: Retrieval **E**nhanced **G**eneration via **A**ctions **IN**sights and **S**tates

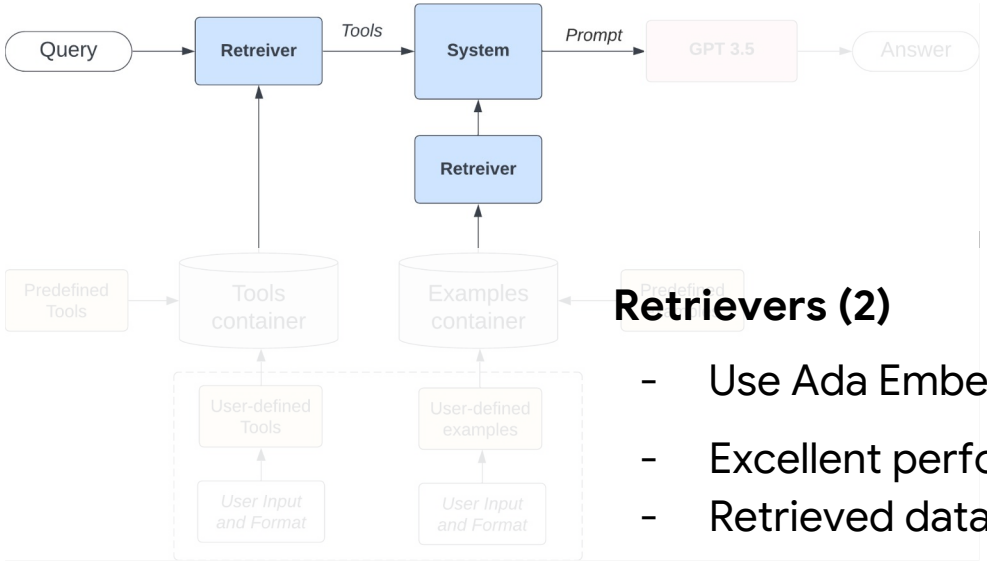
Main Solution: RE-GAINS



Containers (2)

- Have both preset and user-entered data
- built-in type check for input format
- Make the solution effective on unseen tools

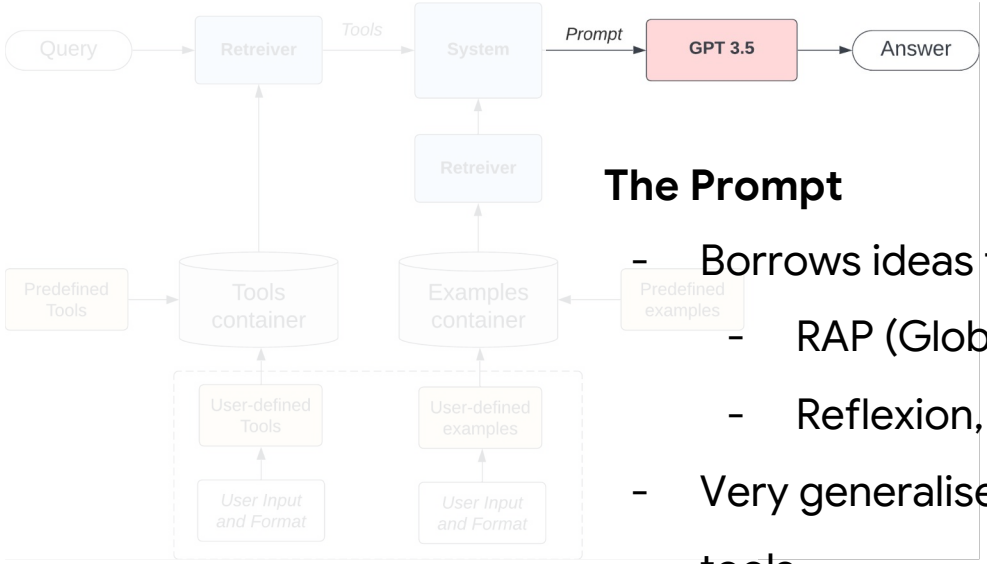
Main Solution: RE-GAINS



Retrievers (2)

- Use Ada Embeddings to retrieve relevant examples/tools
- Excellent performance on large containers
- Retrieved data is augmented to the system prompt

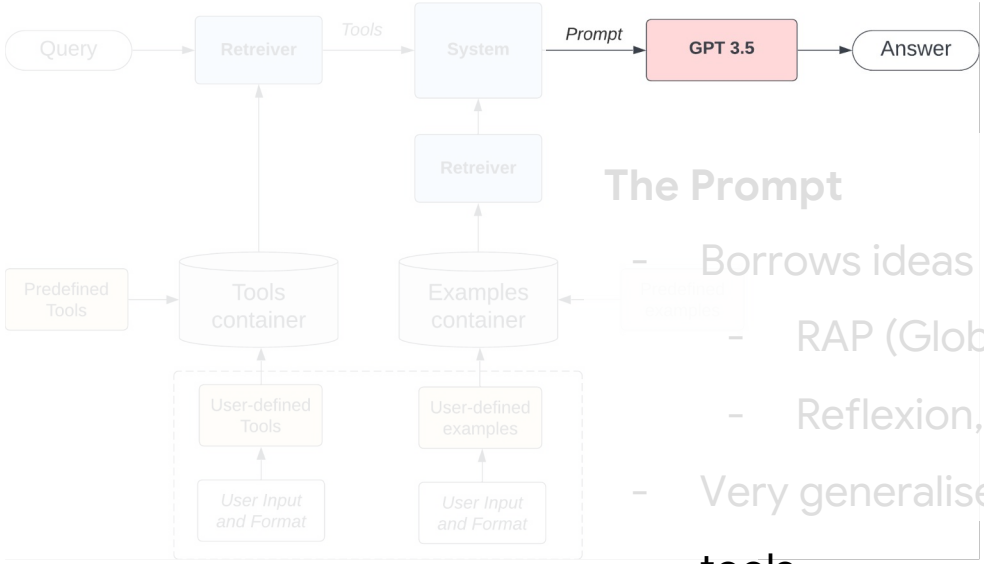
Main Solution: RE-GAINS



The Prompt

- Borrows ideas from
  - RAP (Global State, Actions)
  - Reflexion, Expel (Insights)
- Very generalised; Effective on unseen and new domain tools

Main Solution: RE-GAINS



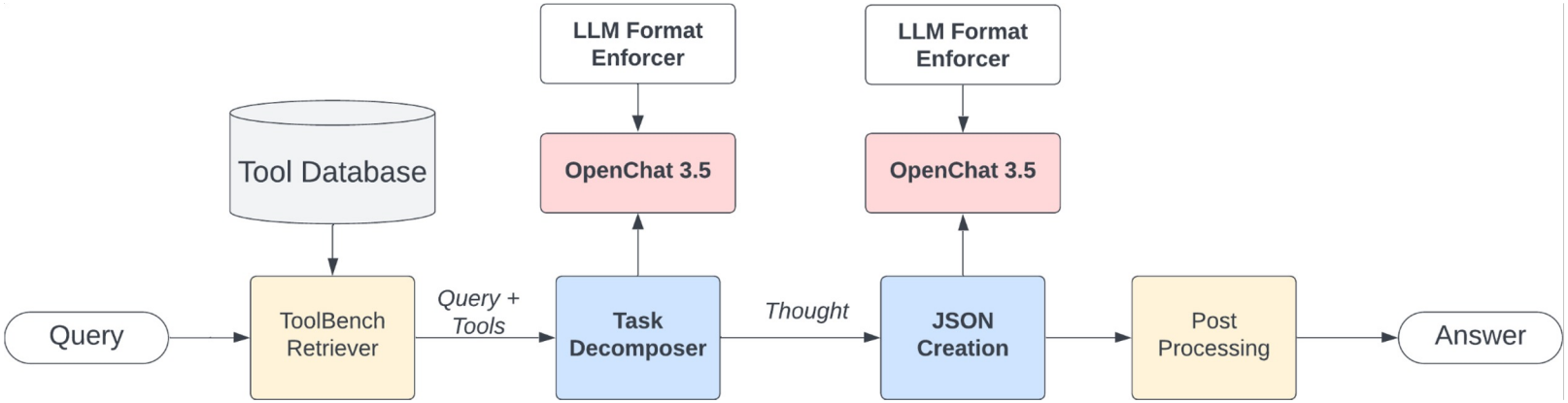
The Prompt

- Borrows ideas from
  - RAP (Global State, Actions)
  - Reflexion, Expel (Insights)
- Very generalised; Effective on unseen and new domain tools

Bonus

- Effective sub-questions help chaining argument values

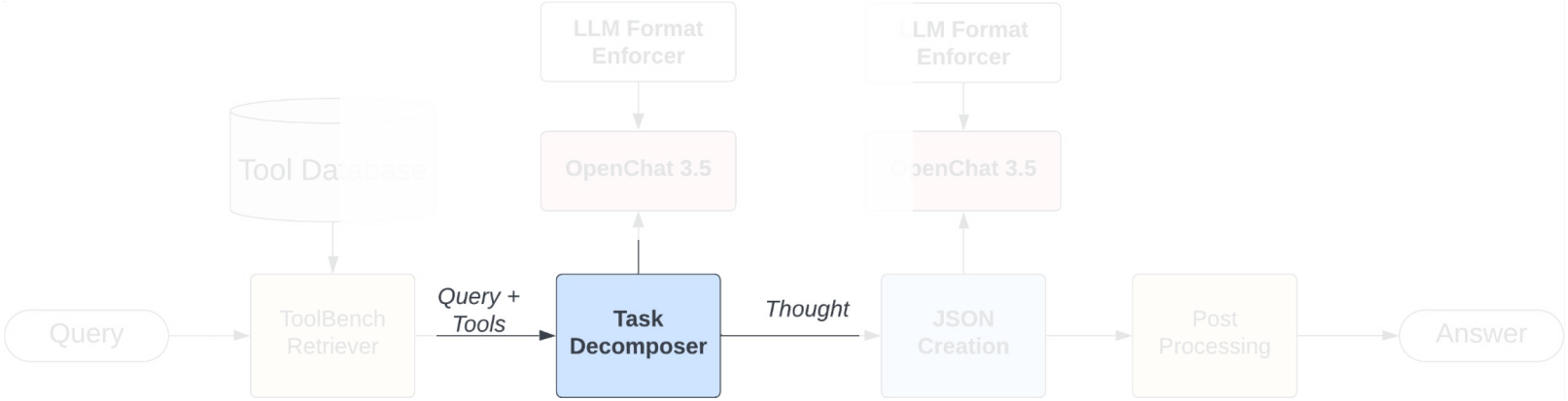
Alternate Solution : EnChANT



EnChANT : Enforced **C**reation of **A**ctions a**n**d **T**houghts

Completely open-source!

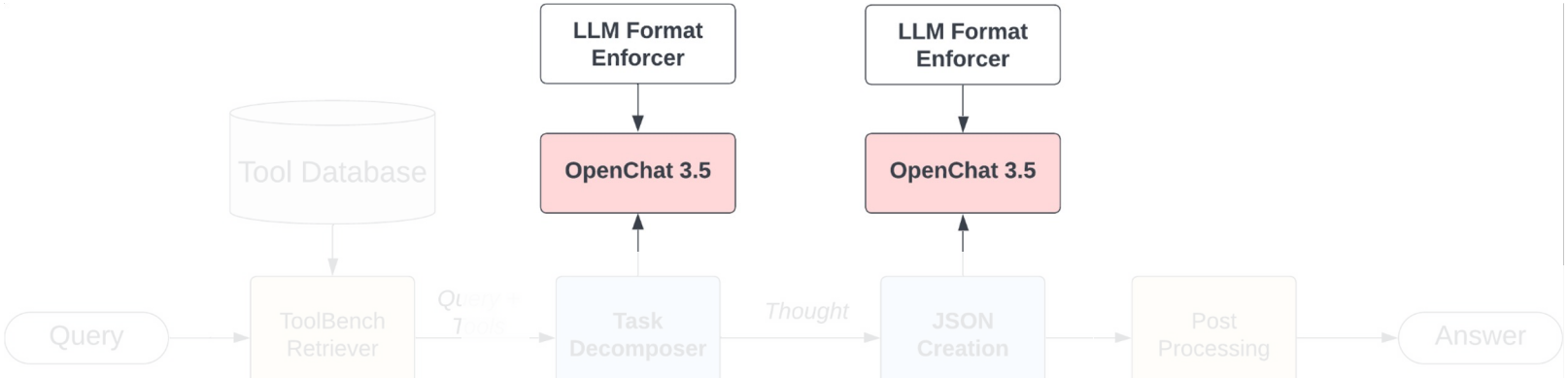
Alternate Solution : EnChANT



Task Decomposer

- ***Thought:*** *improves reasoning*
- ***Tool:*** *ensures each subtask requires a specific tool*
- ***Task:*** *instructions to generate JSON in the next step*

Alternate Solution : EnChANT



LLM Format Enforcer with vLLM (2)

1. **Controlling decomposition**: *each subtask mapped to only one valid tool*
2. **Enforcing *JSON Schema*** *prevents argument name hallucinations & ensures presence of all required arguments*
3. **Upto 24x faster than HF** *using vLLM (Paged Attention)*

Alternate Solution : EnChANT

Merits

- **Best performance** among open-source configurations
- **No hallucinations** in tool/argument names and types, with LLM Enforcer
- Reasoning process is interpretable

Demerits

- ~35s per query

Alternate Solution : EnChANT

Merits

- **Best performance** among open-source configurations
- **No hallucinations** in tool/argument names and types, with LLM Enforcer
- Reasoning process is interpretable

Demerits

- ~35s per query

- ***Data is self-contained***
- *Improvements on latency in future products will make this solution viable*

Latency & Cost

Implementation	Model Name	Latency (s)	Cost ($)
RAP prompt	gpt-4-1106-preview	17.03	0.080
Main Solution *	gpt-3.5-turbo	8.62	0.009
Alternate Solution	openchat-3.5	35.68	0.008

For a similar number of tokens (2600 input/260 output)

- **Both solutions: 10x Cheaper** than GPT-4
- **Main solution: 2x Faster** than GPT-4

* *Retriever overhead 10^{-6} per query*

Results

Implementation	Model Name	IR ↓	NR ↑	HR ↓	MR ↓	BLEU Score ↑	ROUGE-L-F1 Score ↑
ControlLLM	gpt-4-1106-preview	0.022	0.978	0.052	0.152	0.953	0.729
RAP	gpt-4-1106-preview	0.014	0.986	0.041	0.029	0.984	0.847
RAP	openchat-3.5	0.437	0.563	0.875	0.875	0.278	0.125
RAP	llama-70b	0.596	0.404	0.254	0.456	0.456	0.332
ControlLLM	openchat-3.5	0.250	0.750	0.250	0.367	0.483	0.304
ControlLLM	llama-70b	0.602	0.398	0.405	0.337	0.432	0.286
Main Solution	gpt-3.5-turbo-1106	0.040	**0.960**	**0.082**	0.154	**0.941**	0.725
Alternate Solution	openchat-3.5	0.214	**0.786**	**0.005**	0.255	**0.742**	0.666

on DevRev & Golden dataset

***IR**: Irrelevant Tool Inclusion Rate*

***NR**: Necessary Tool Inclusion Rate*

***HR**: Resource Hallucination Rate*

***MR**: Missing Tool Rate*

Results

Implementation	Model Name	IR ↓	NR ↑	HR ↓	MR ↓	BLEU Score ↑	ROUGE-L-F1 Score ↑
ControlLLM	gpt-4-1106-preview	0.022	0.978	0.052	0.152	0.953	0.729
RAP	gpt-4-1106-preview	0.014	0.986	0.041	0.029	0.984	0.847
RAP	openchat-3.5	0.437	0.563	0.875	0.875	0.278	0.125
RAP	llama-70b	0.596	0.404	0.254	0.456	0.456	0.332
ControlLLM	openchat-3.5	0.250	0.750	0.250	0.367	0.483	0.304
ControlLLM	llama-70b	0.602	0.398	0.405	0.337	0.432	0.286
Main Solution	gpt-3.5-turbo-1106	0.040	0.960	0.082	0.154	0.941	0.725
Alternate Solution	openchat-3.5	0.214	0.786	0.005	0.255	0.742	0.666

on DevRev & Golden dataset

- **GPT4** with ControlLLM / RAP usually gives **near-perfect results**

***IR**: Irrelevant Tool Inclusion Rate*
***NR**: Necessary Tool Inclusion Rate*
***HR**: Resource Hallucination Rate*
***MR**: Missing Tool Rate*

Results

Implementation	Model Name	IR ↓	NR ↑	HR ↓	MR ↓	BLEU Score ↑	ROUGE-L-F1 Score ↑
ControlLLM	gpt-4-1106-preview	0.022	0.978	0.052	0.152	0.953	0.729
RAP	gpt-4-1106-preview	0.014	0.986	0.041	0.029	0.984	0.847
RAP	openchat-3.5	0.437	0.563	0.875	0.875	0.278	0.125
RAP	llama-70b	0.596	0.404	0.254	0.456	0.456	0.332
ControlLLM	openchat-3.5	0.250	0.750	0.250	0.367	0.483	0.304
ControlLLM	llama-70b	0.602	0.398	0.405	0.337	0.432	0.286
Main Solution	gpt-3.5-turbo-1106	0.040	0.960	0.082	0.154	0.941	0.725
Alternate Solution	openchat-3.5	0.214	0.786	0.005	0.255	0.742	0.666

on DevRev & Golden dataset

- RAP / ControlLLM with open-source models give average results

***IR**: Irrelevant Tool Inclusion Rate*
***NR**: Necessary Tool Inclusion Rate*
***HR**: Resource Hallucination Rate*
***MR**: Missing Tool Rate*

Results

Implementation	Model Name	IR ↓	NR ↑	HR ↓	MR ↓	BLEU Score ↑	ROUGE-L-F1 Score ↑
ControlLLM	gpt-4-1106-preview	0.022	0.978	0.052	0.152	0.953	0.729
RAP	gpt-4-1106-preview	0.014	0.986	0.041	0.029	0.984	0.847
RAP	openchat-3.5	0.437	0.563	0.875	0.875	0.278	0.125
RAP	llama-70b	0.596	0.404	0.254	0.456	0.456	0.332
ControlLLM	openchat-3.5	0.250	0.750	0.250	0.367	0.483	0.304
ControlLLM	llama-70b	0.602	0.398	0.405	0.337	0.432	0.286
Main Solution	gpt-3.5-turbo-1106	0.040	**0.960**	**0.082**	0.154	**0.941**	0.725
Alternate Solution	openchat-3.5	0.214	**0.786**	**0.005**	0.255	**0.742**	0.666

on DevRev & Golden dataset

- **Main Solution is at par with GPT4**
- High BLEU and ROUGE scores

***IR**: Irrelevant Tool Inclusion Rate*
***NR**: Necessary Tool Inclusion Rate*
***HR**: Resource Hallucination Rate*
***MR**: Missing Tool Rate*

Results

Implementation	Model Name	IR ↓	NR ↑	HR ↓	MR ↓	BLEU Score ↑	ROUGE-L-F1 Score ↑
ControlLLM	gpt-4-1106-preview	0.022	0.978	0.052	0.152	0.953	0.729
RAP	gpt-4-1106-preview	0.014	0.986	0.041	0.029	0.984	0.847
RAP	openchat-3.5	0.437	0.563	0.875	0.875	0.278	0.125
RAP	llama-70b	0.596	0.404	0.254	0.456	0.456	0.332
ControlLLM	openchat-3.5	0.250	0.750	0.250	0.367	0.483	0.304
ControlLLM	llama-70b	0.602	0.398	0.405	0.337	0.432	0.286
Main Solution	gpt-3.5-turbo-1106	0.040	**0.960**	**0.082**	0.154	**0.941**	0.725
Alternate Solution	openchat-3.5	0.214	**0.786**	**0.005**	0.255	**0.742**	0.666

on DevRev & Golden dataset

- **Alternate Solution has lowest Hallucinations**

***IR**: Irrelevant Tool Inclusion Rate*
***NR**: Necessary Tool Inclusion Rate*
***HR**: Resource Hallucination Rate*
***MR**: Missing Tool Rate*

Results

Implementation	Model Name	IR ↓	NR ↑	HR ↓	MR ↓	BLEU Score ↑	ROUGE-L-F1 Score ↑
ControlLLM	gpt-4-1106-preview	0.022	0.978	0.052	0.152	0.953	0.729
RAP	gpt-4-1106-preview	0.014	0.986	0.041	0.029	0.984	0.847
RAP	openchat-3.5	0.437	0.563	0.875	0.875	0.278	0.125
RAP	llama-70b	0.596	0.404	0.254	0.456	0.456	0.332
ControlLLM	openchat-3.5	0.250	0.750	0.250	0.367	0.483	0.304
ControlLLM	llama-70b	0.602	0.398	0.405	0.337	0.432	0.286
Main Solution	gpt-3.5-turbo-1106	0.040	**0.960**	**0.082**	0.154	**0.941**	0.725
Alternate Solution	openchat-3.5	0.214	**0.786**	**0.005**	0.255	**0.742**	0.666

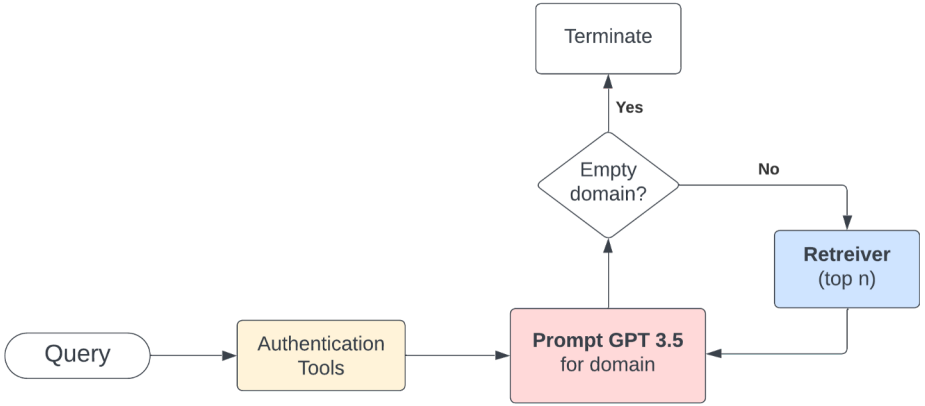
on DevRev & Golden dataset

- Alternate Solution has lowest Hallucinations
- **Beats other open-source models**

***IR**: Irrelevant Tool Inclusion Rate*
***NR**: Necessary Tool Inclusion Rate*
***HR**: Resource Hallucination Rate*
***MR**: Missing Tool Rate*

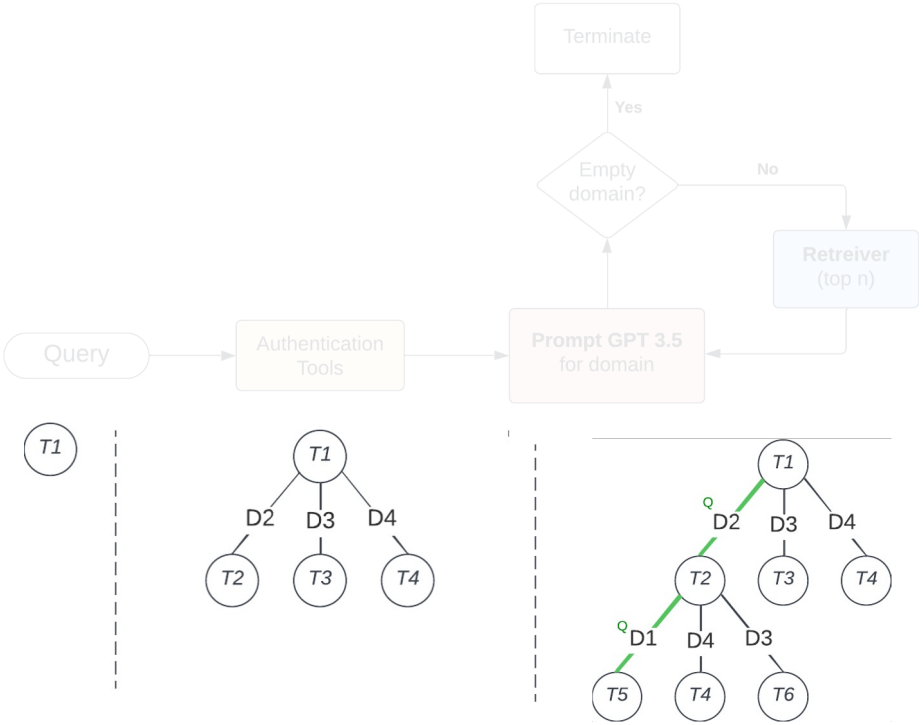
Experiments and Future Ideas: Reasoning Tree

- **We propose domains** which classify tools wrt functionality
- LLM + Retriever create next set of leaf nodes (tools)

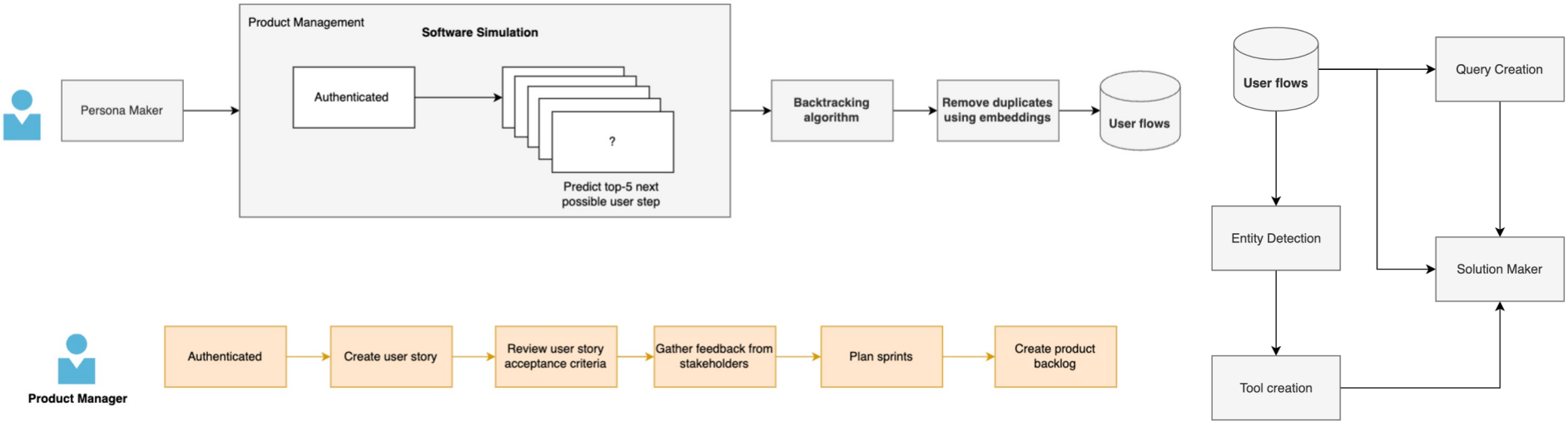


Experiments and Future Ideas: Reasoning Tree

- **We propose domains** which classify tools wrt functionality
- LLM + Retriever create next set of leaf nodes (tools)
- Tree represents a large **reasoning space**
- Q-values evaluate best path
- *Observed to be significantly better than CoT*

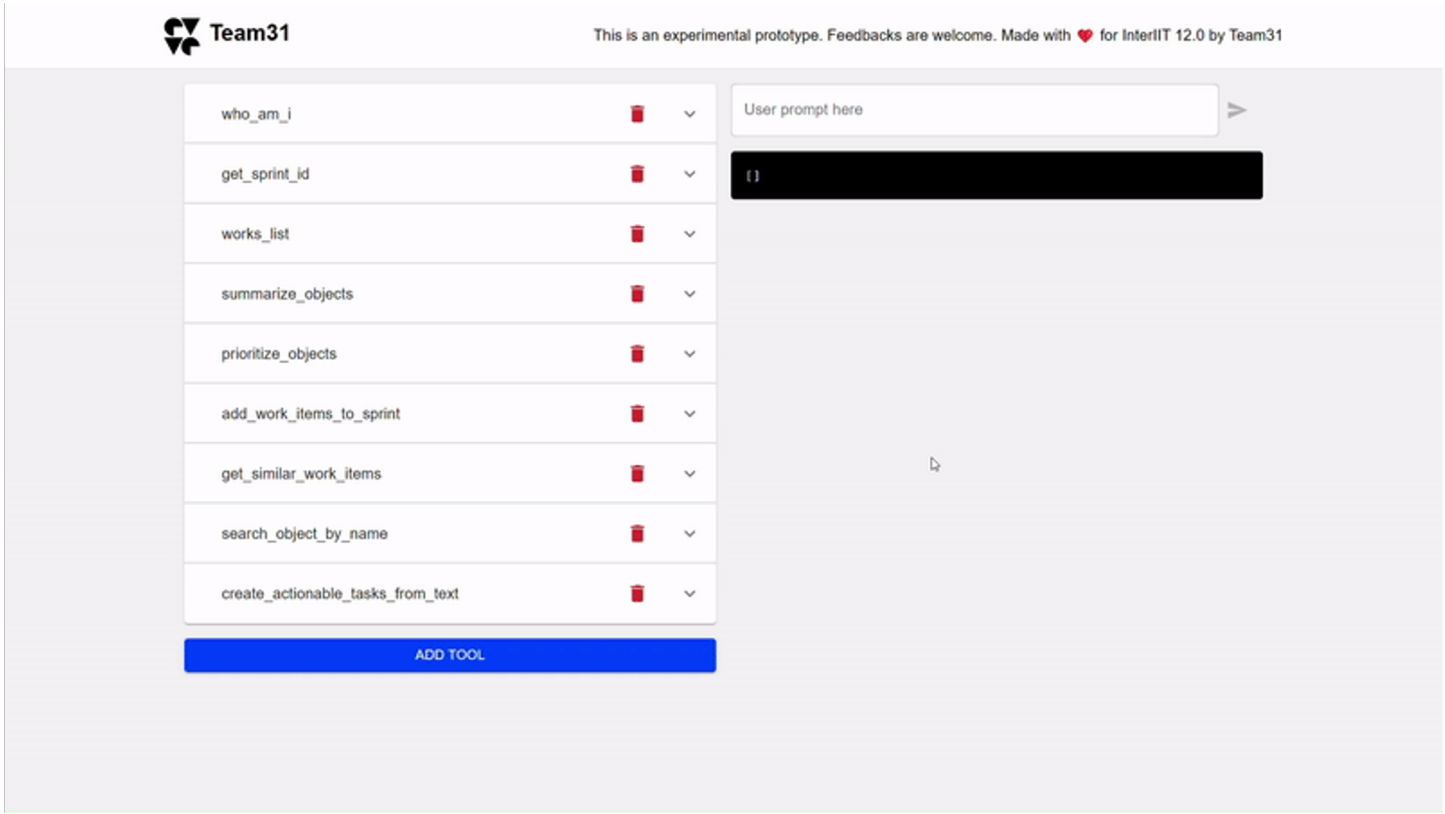


Experiments and Future Ideas: Domain Agnostic Data Generation



- We simulate a persona in a workflow environment to get user flows
- user flows - create entities - tools - queries - solutions

Website Demo



Thank you!

Team 31

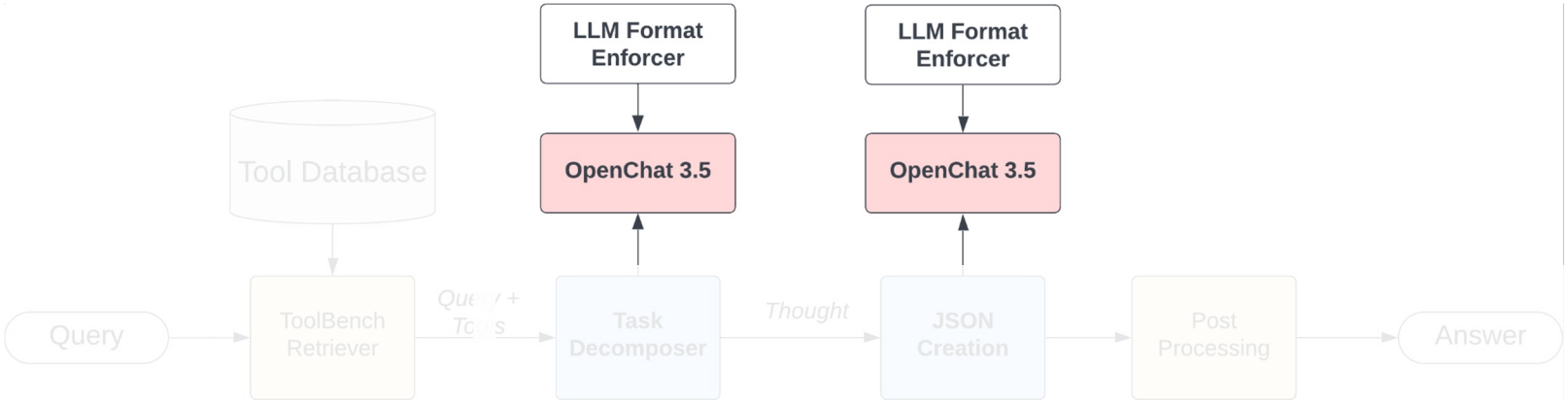
Appendix

Thought-tool-task for interpretation

Example :Summarize work items similar to don:core:dvrv-us-1:devo/0:issue/1”

```
[
    {
        "thought": "First, we need to get the list of work items similar to the work item id",
        "tool_name": "get_similar_work_items",
        "task": "Use the get_similar_work_items tool with the argument 'work_id' = 'don:core:dvrv-us-1:devo/0:issue/1'"
    },
    {
        "thought": "Next, we need to summarize the work items obtained from the previous task",
        "tool_name": "summarize_objects",
        "task": "Use the summarize_objects tool with the output of the first task, argument 'objects' = $$PREV[0]"
    }
]
```

LLM Enforcer working



How it works?

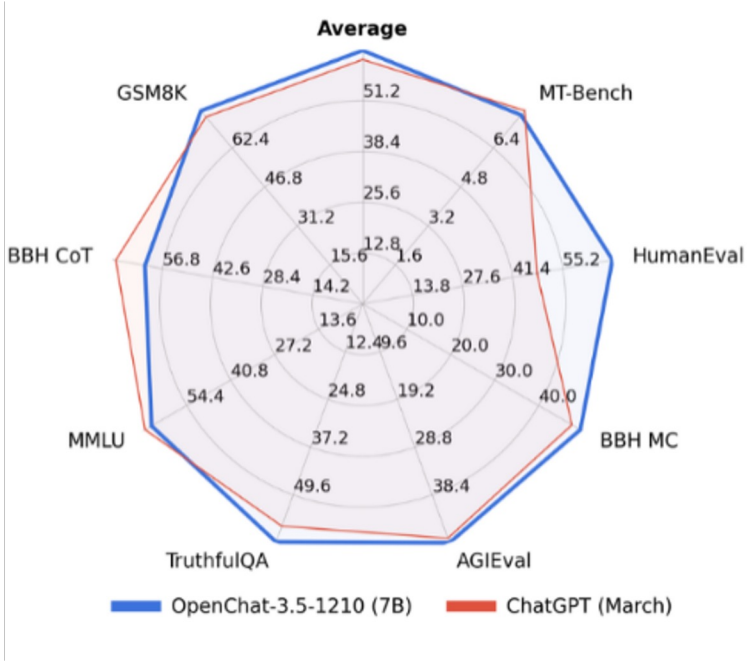
- *when models are predicting next tokens*
- *it filters out all invalid tokens*
- *leaving only those that fit the schema*

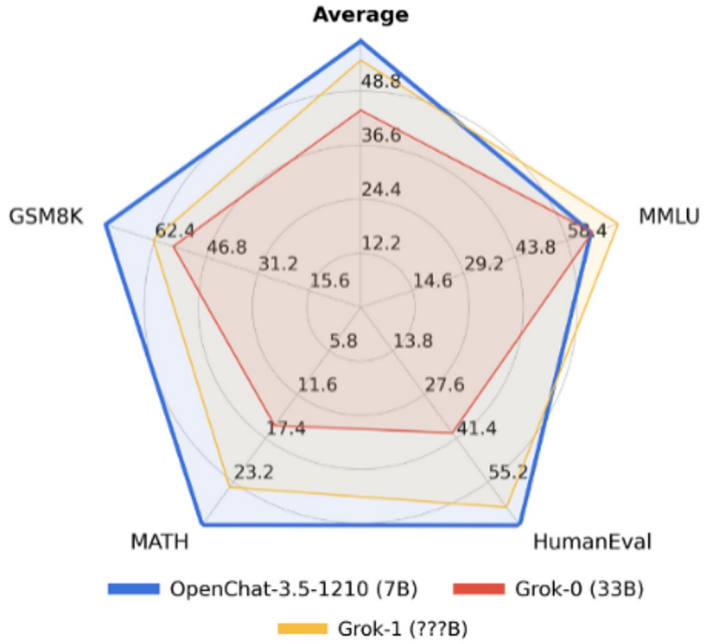
Retriever benchmarks

	OpenAI	ToolBench Retriever
Top 5	0.7625	0.7325
Top 7	0.8562	0.8367
Top 9	0.9479	0.9362

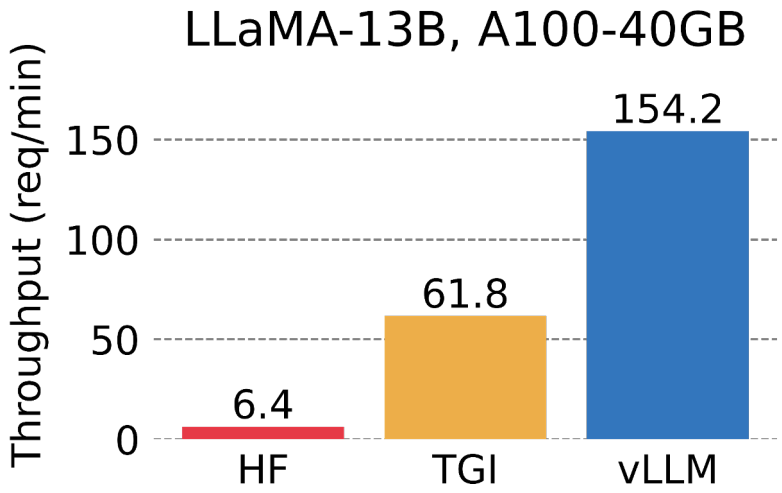
Benchmarking the two major Dense retrievers we use: on 9 DevRev + 6 new tools. The Top 'N' score indicates the average percentage of tools needed to solve the query that are in the list of top 'N' fetched tools.

Why Openchat is best in open-source models





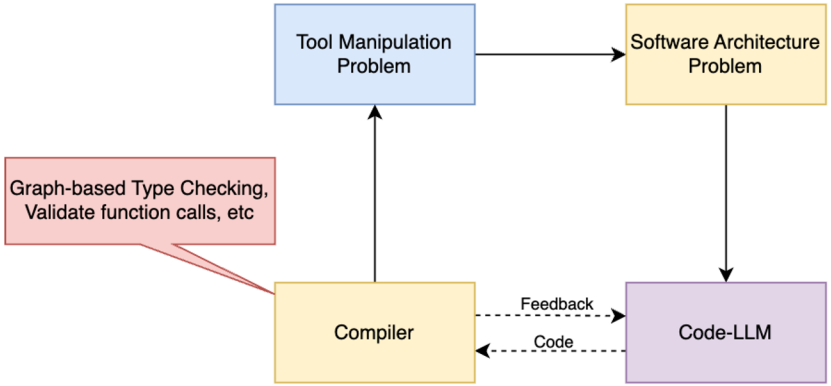
Why vLLM

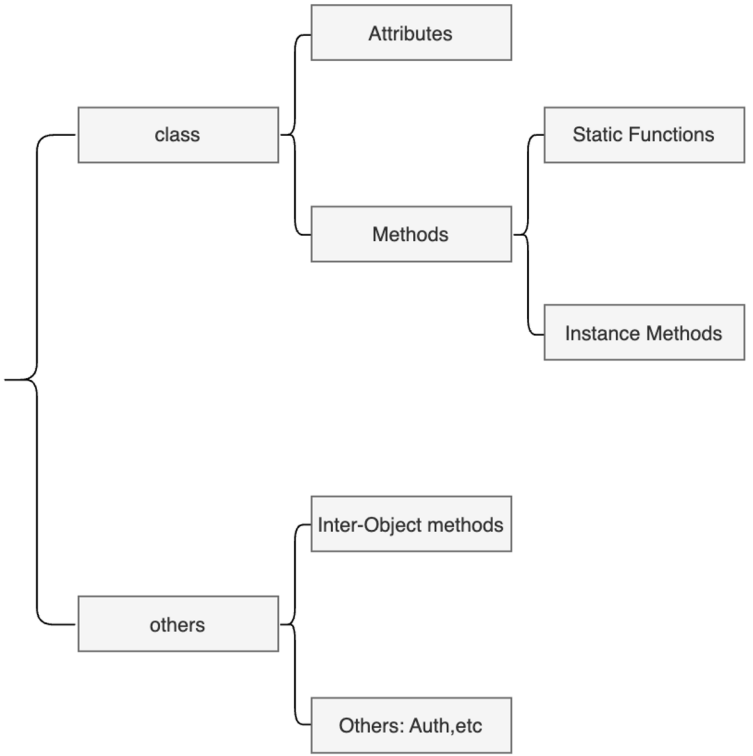


vLLM achieves 14x - 24x higher throughput than HF

Future Ideas: Software Architecture

- Minimize number of tokens generated
- Perfect blend of algorithms and LLMs





Main Solution

You are a query solver. You will be given tools. Using those tools, you have to solve the query.
To solve the query, at each point, you have to ask sub-questions. These sub-questions are "What is the next tool to use, its arguments and argument values?". Look at answers to the previous sub-questions, which will give you context of how the current set of tools have been chosen so far. Compare this to the greater context, which is, how to solve the next question.

Some important points :
1) Tool description and argument description are very important. Read them to understand what exactly a certain tool generates as output or what inputs a tool can get.
2) Output of tools in the previous step is input to tool for current statement. Compare descriptions, types and examples to get a huge clue.
3) Always check if authentication tools like "who_am_i", "team_id", "get_sprint_id", etc. are needed at any point.
4) Take care of "type" argument in "works_list" is issues, tickets or tasks are explicitly mentioned.
5) Stop once you feel the task is complete and no further tools are needed to solve the query.
6) To answer the query, you are only allowed to use the tools that we have provided.
7) If the question is simply unsolvable using any tool we have, only return [].

Tools (in JSON format) :
{tools}

Given input as a query, generate output as a list of the sub-questions and answers at each step. Also output a JSON as shown in examples.

Examples :
{examples}