DevRev's AI Agent 007

Tooling up for Success

Team 15

Agenda

01 Inference

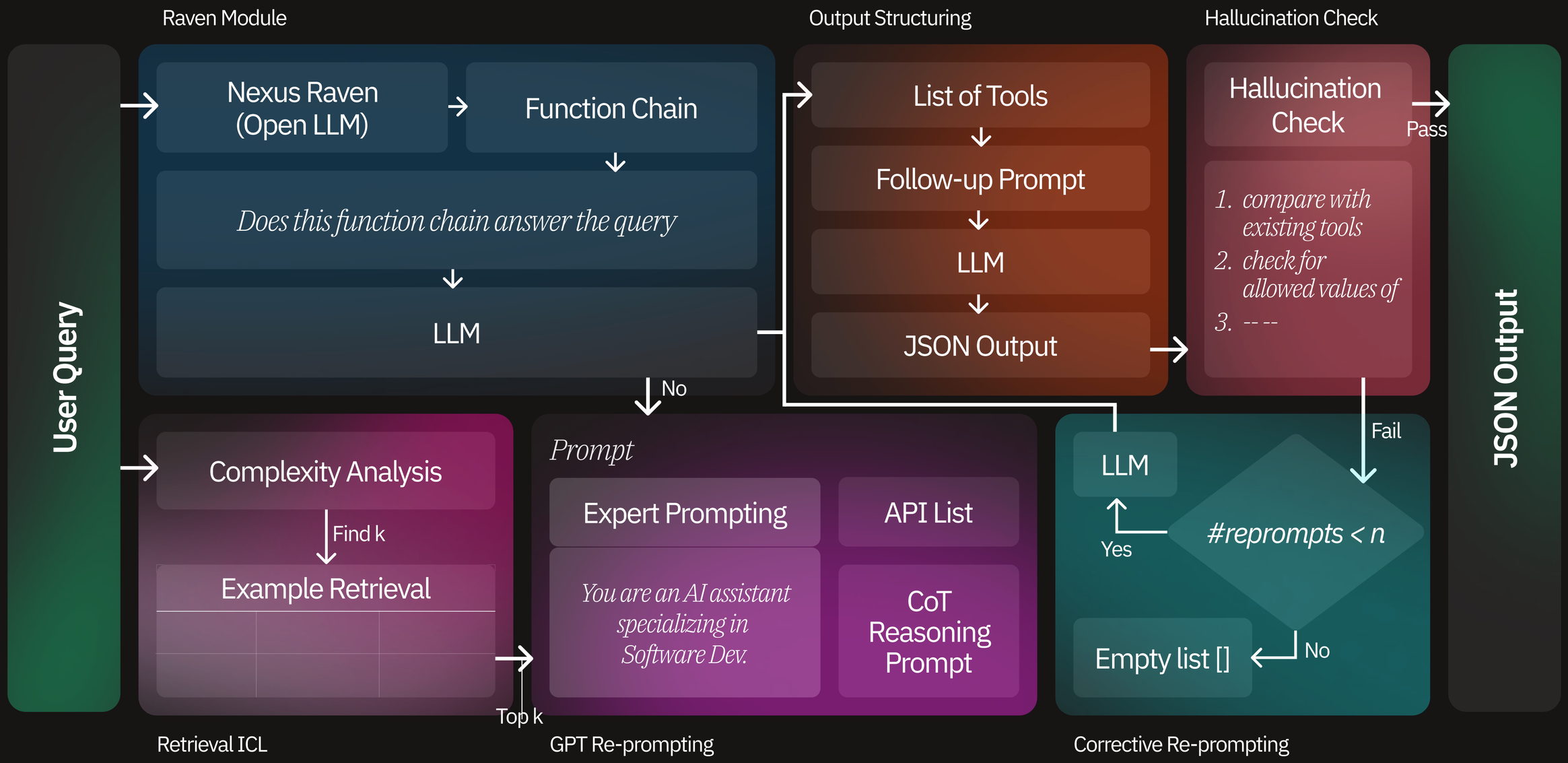
02 Tool Management

03 Data Generation

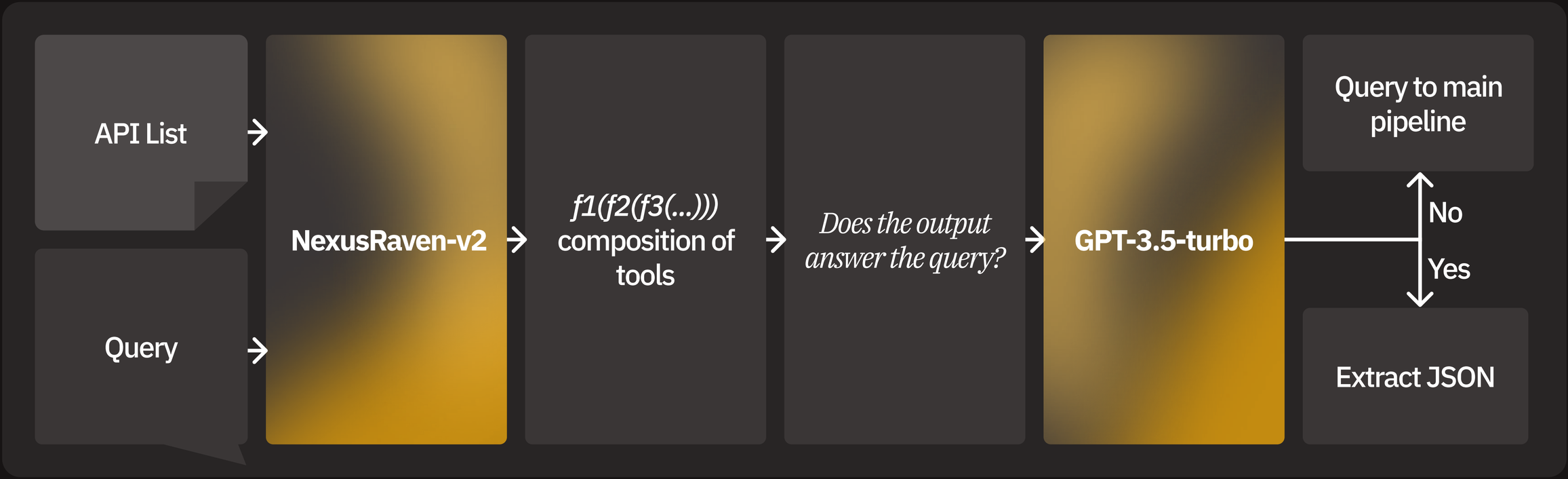
04 Evaluation

05 Experimentation

Inference Pipeline



Nexus Raven

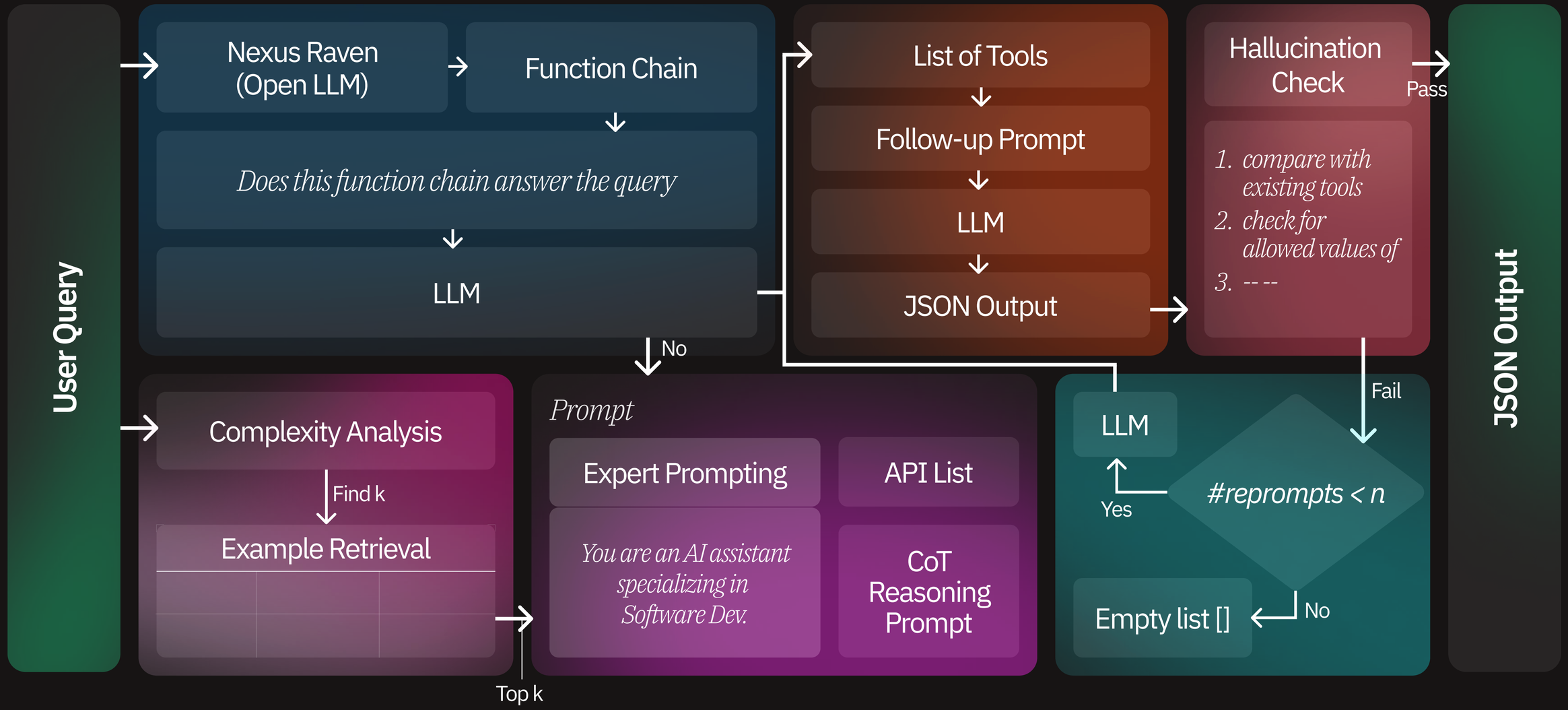


Sample Raven Outputs:

Query: Summarize issues similar to `don:core:dvrv-us-1.`

Output: **`summarize_objects`**`(objects=`**`get_similar_work_items`**`(work_id='don:core:dvrv-us-1'))`

Inference Pipeline



Chain of Thought

Prompt

Expert Prompting

You are an AI assistant specializing in Software Dev.

API List

CoT Reasoning Prompt

Chat History

Expert Prompt

List of APIs

Few Shot CoT examples

User Query

Chain of Thought

Prompt

Expert Prompting

You are an AI assistant specializing in Software Dev.

API List

CoT Reasoning Prompt

Chat History

Expert Prompt

List of APIs

Few Shot CoT examples

User Query

Chain of Thought

Prompt

Expert Prompting

You are an AI assistant specializing in Software Dev.

API List

CoT Reasoning Prompt

Chat History

Expert Prompt

List of APIs

Few Shot CoT examples

User Query

Chain of Thought

Prompt

Expert Prompting

You are an AI assistant specializing in Software Dev.

API List

CoT Reasoning Prompt

Chat History

Expert Prompt

List of APIs

Few Shot CoT examples

User Query

Chain of Thought

Prompt

Expert Prompting

You are an AI assistant specializing in Software Dev.

API List

CoT Reasoning Prompt

Chat History

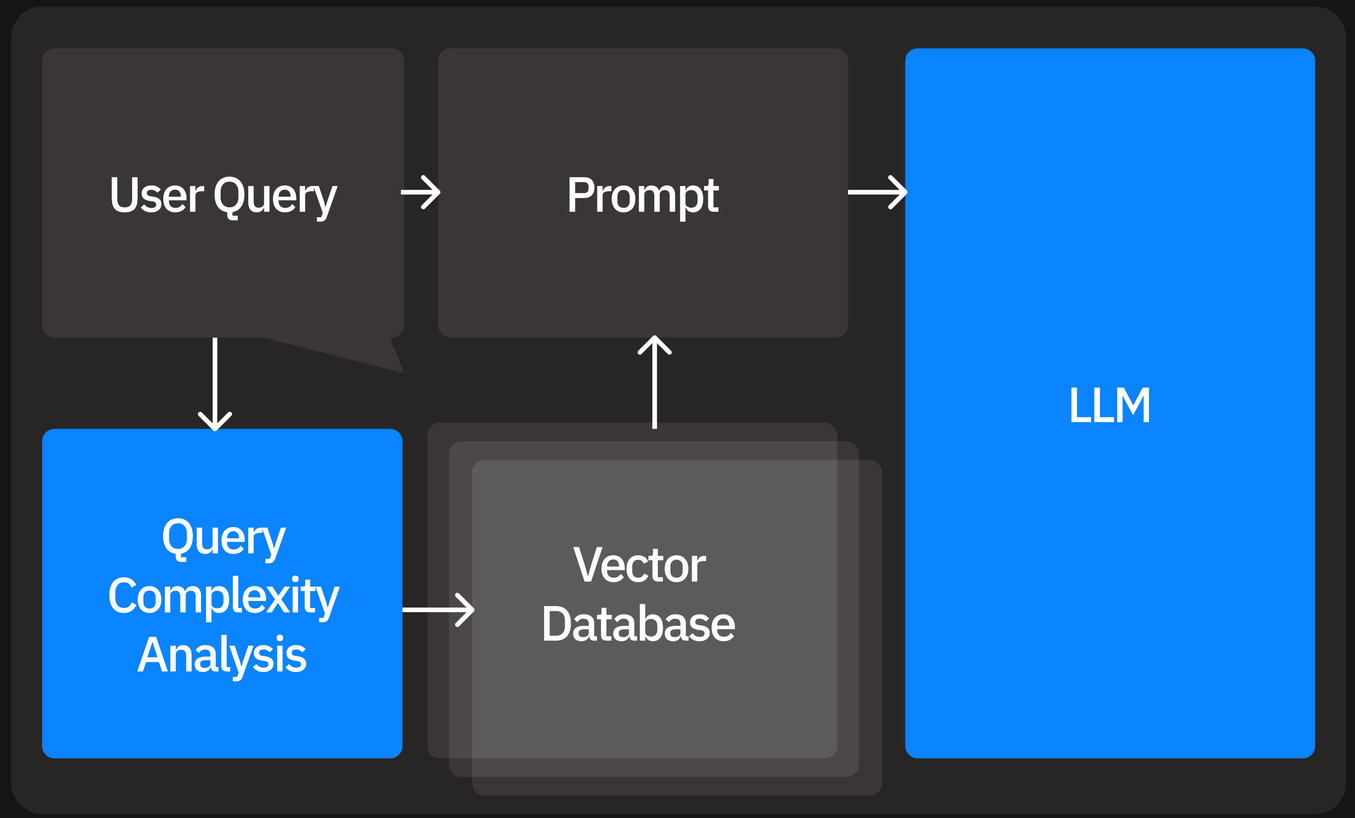
Expert Prompt

List of APIs

Few Shot CoT examples

User Query

Retrieval - ICL



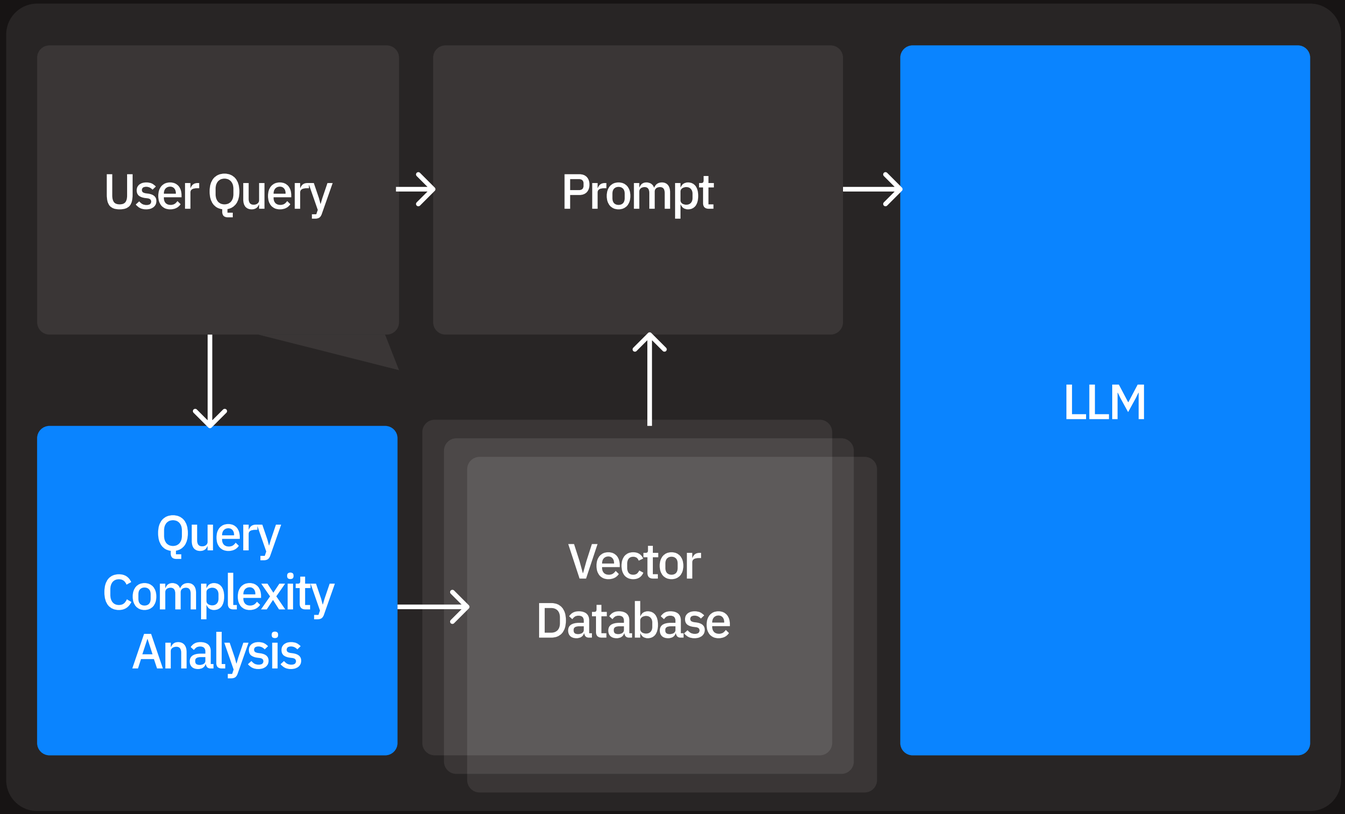
Embeddings and Vector DB

FAISS

MMR Search

Dynamic Top K

Retrieval - ICL



Embeddings and Vector DB

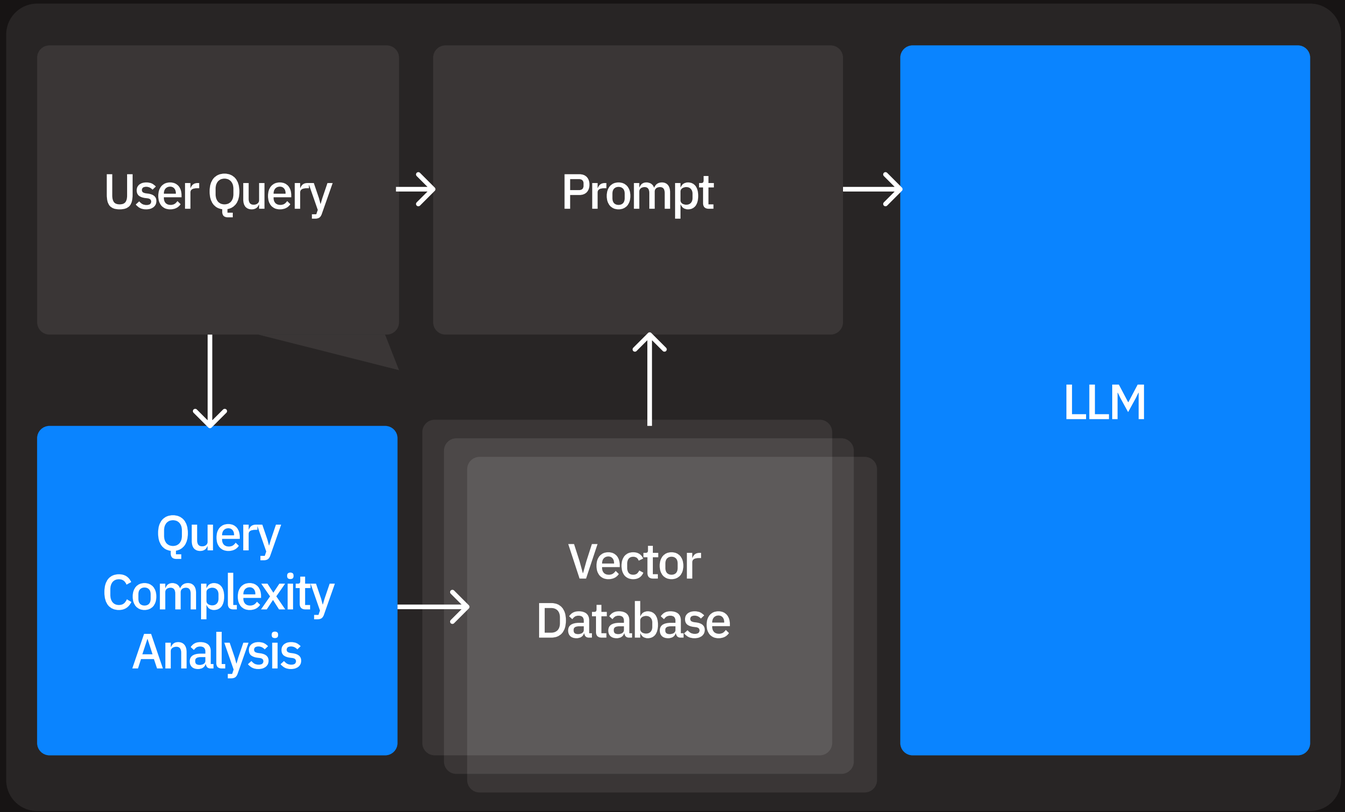
Store embeddings of query-solution pairs generated using Sentence Transformer embeddings. A set of examples stored initially, later automated with GPT-4.

FAISS

MMR Search

Dynamic Top K

Retrieval - ICL



Embeddings and Vector DB

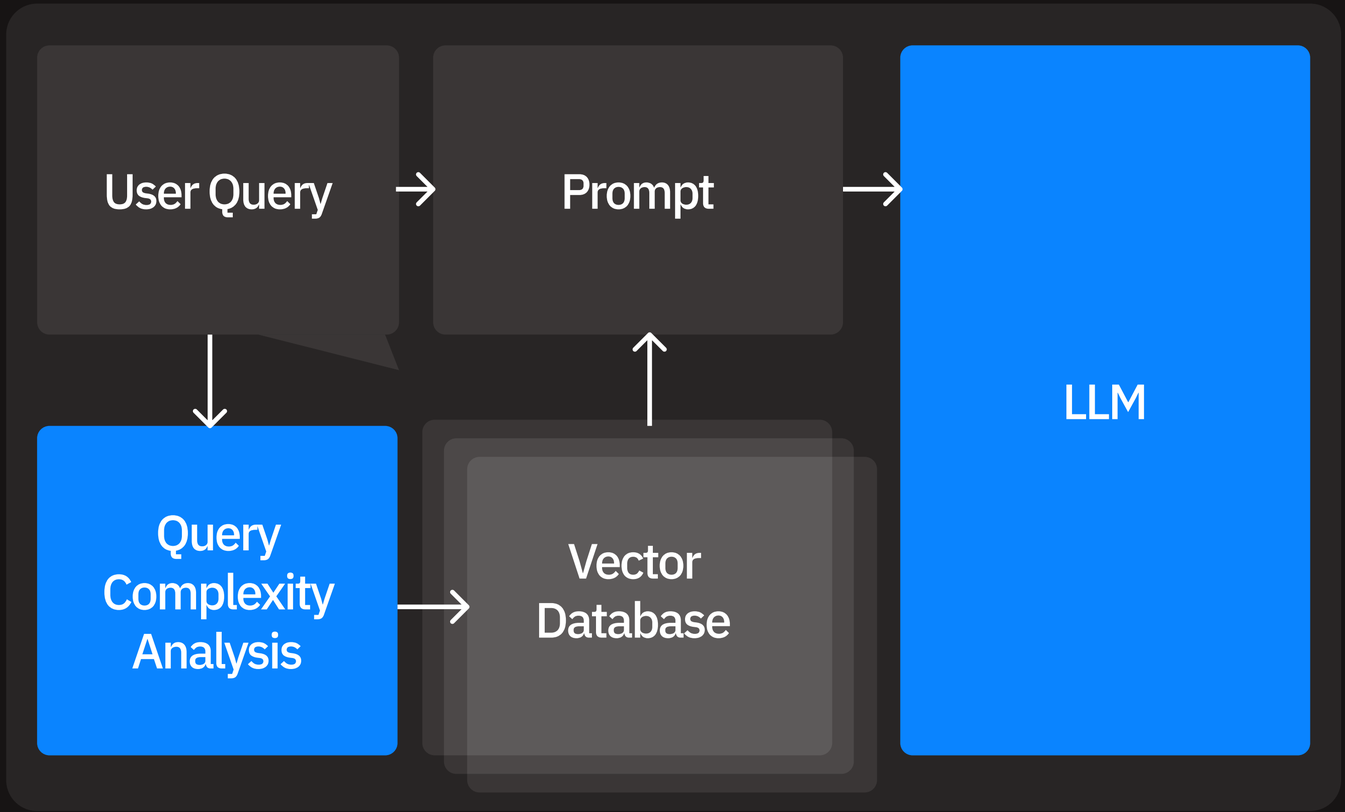
FAISS

FAISS used as vector store. It uses nearest-neighbor search implementation, optimizes memory-speed-accuracy tradeoff.

MMR Search

Dynamic Top K

Retrieval - ICL



Embeddings and Vector DB

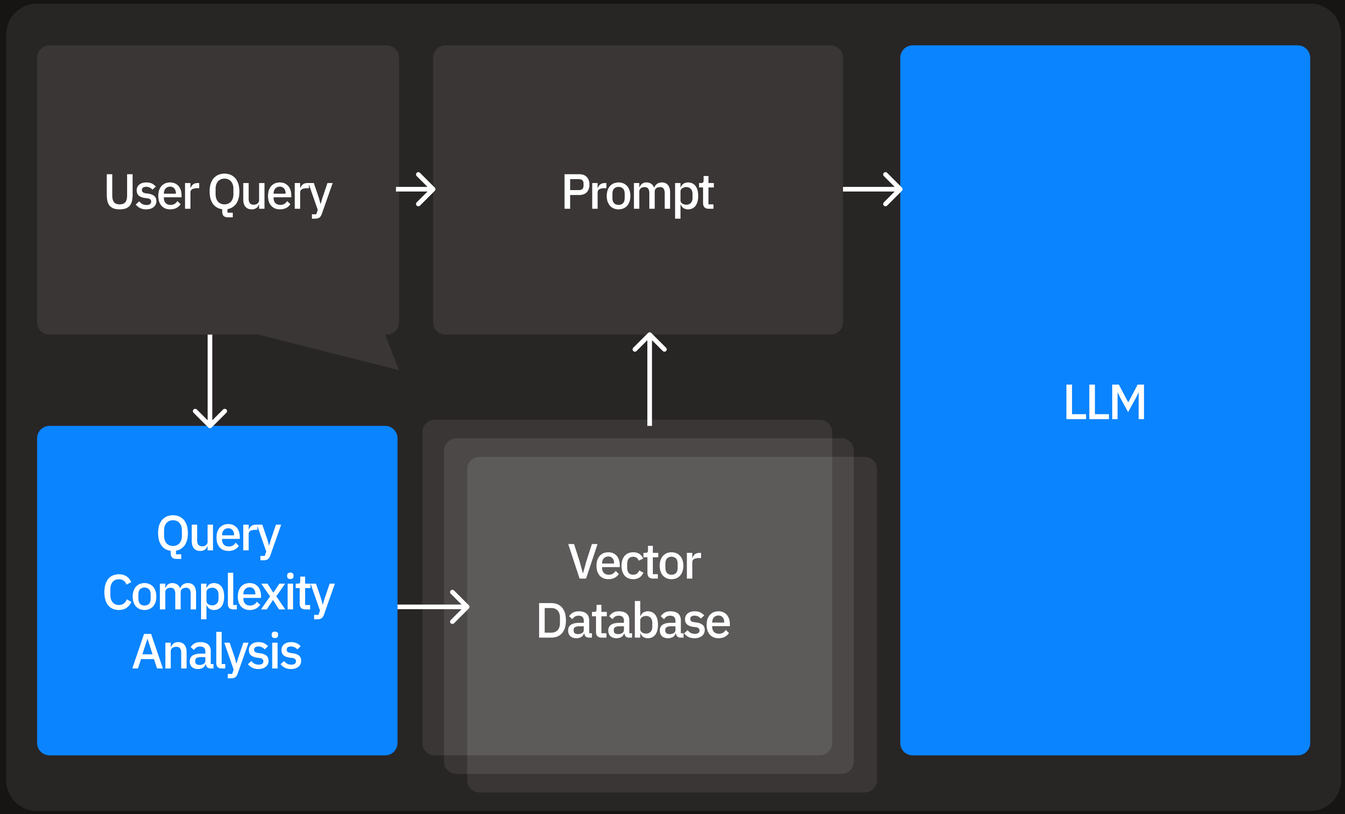
FAISS

MMR Search

We use MMR search to select similar examples while also optimizing for diversity.

Dynamic Top K

Retrieval - ICL



Embeddings and Vector DB

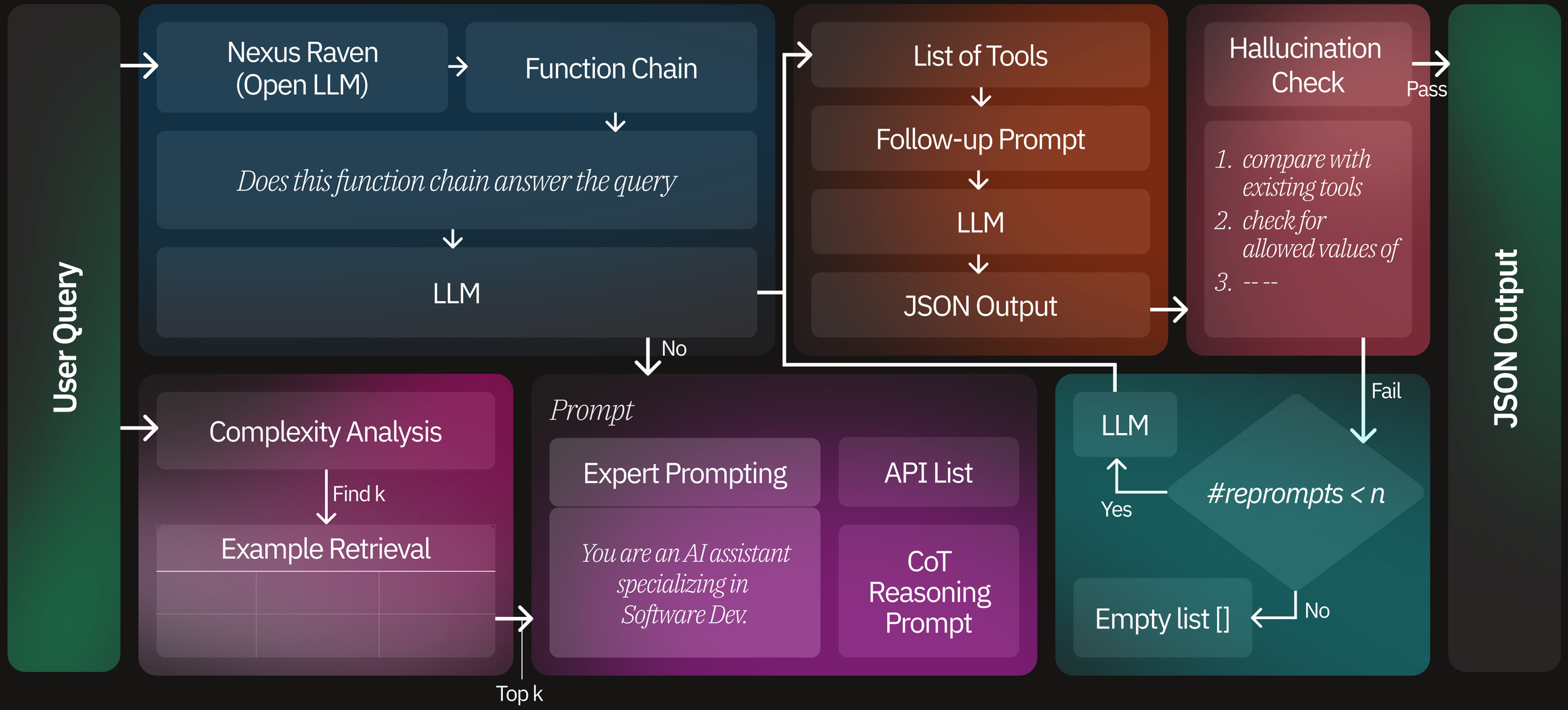
FAISS

MMR Search

Dynamic Top K

Decide number of examples based on query complexity. Used length as metric, others include Gunning Fog Index & SMOG grade.

Inference Pipeline



Hallucination Check

Tool Name Validation	We validate each tool name in the JSON response against the list of available tools.
Argument Name Validation	We check each argument's name in the JSON response to ensure it exists in the list of available arguments.
Argument Value Formatting	We maintain args_in_list_dict, to ensure that arguments that should be in a list format are indeed lists.
Argument Value Validation	We validate argument values against the allowed values defined in allowed_args_dict (for those arguments for which this is applicable).

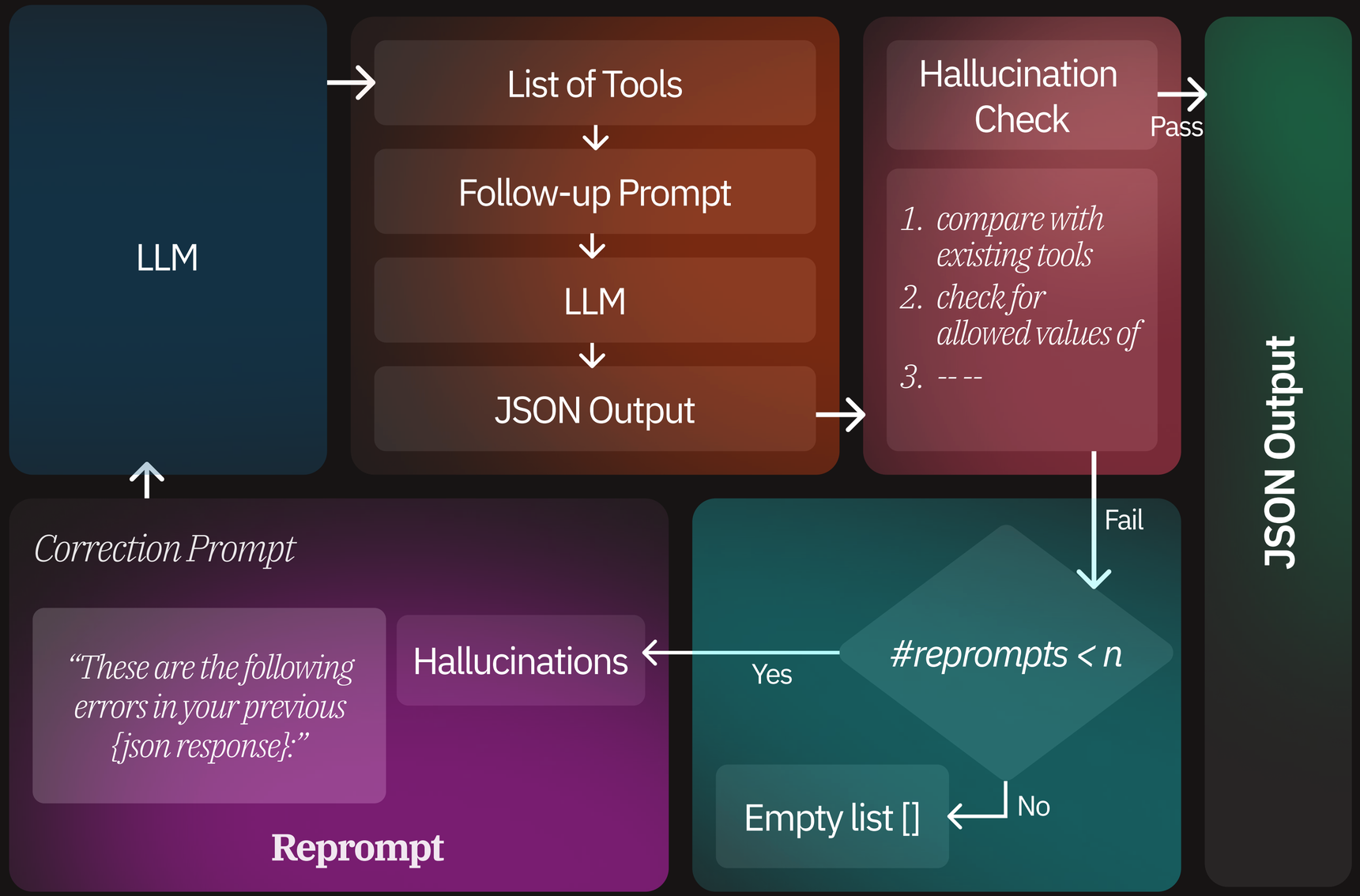
Hallucination Check

Tool Name Validation	We validate each tool name in the JSON response against the list of available tools.
Argument Name Validation	We check each argument's name in the JSON response to ensure it exists in the list of available arguments.
Argument Value Formatting	We maintain args_in_list_dict, to ensure that arguments that should be in a list format are indeed lists.
Argument Value Validation	We validate argument values against the allowed values defined in allowed_args_dict (for those arguments for which this is applicable).

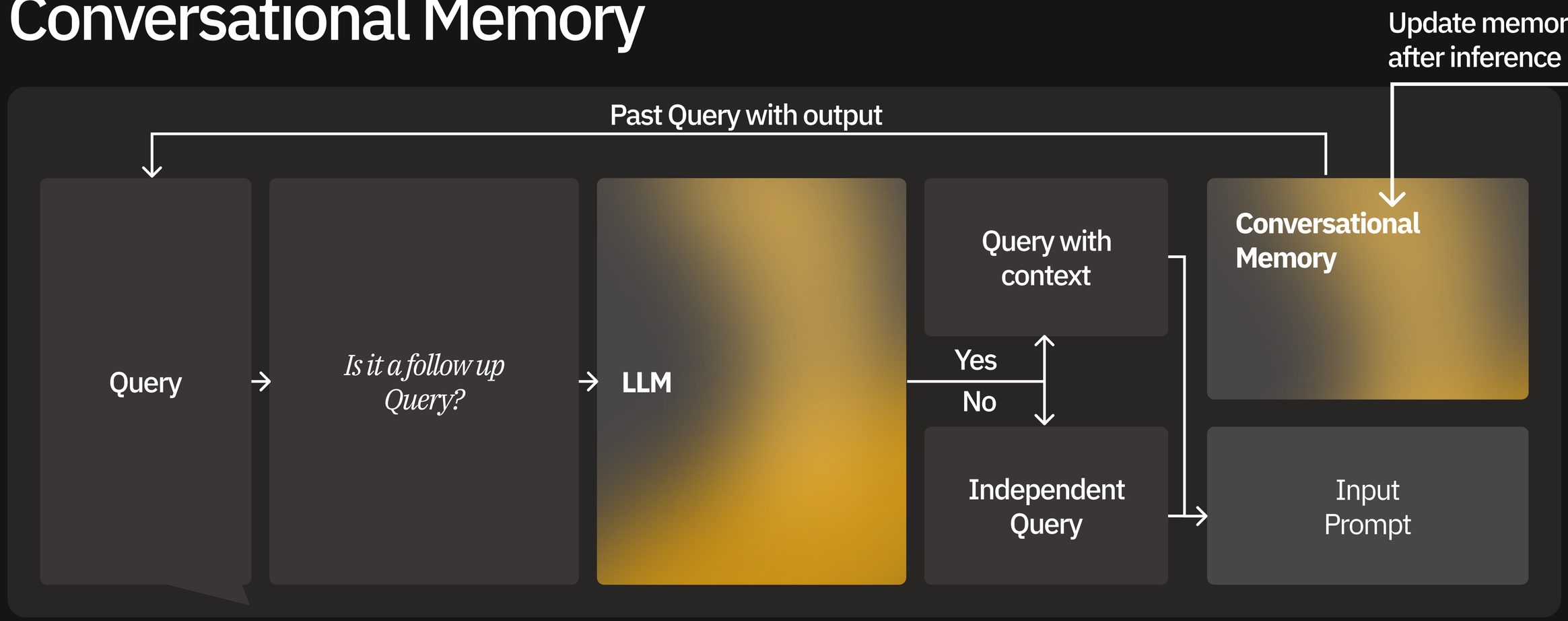
Hallucination Check

Tool Name Validation	We validate each tool name in the JSON response against the list of available tools.
Argument Name Validation	We check each argument's name in the JSON response to ensure it exists in the list of available arguments.
Argument Value Formatting	We maintain args_in_list_dict, to ensure that arguments that should be in a list format are indeed lists.
Argument Value Validation	We validate argument values against the allowed values defined in allowed_args_dict (for those arguments for which this is applicable).

Corrective Reprompting



Conversational Memory



1. **“Is the current query a follow-up of the previous query?”**
2. If not a follow-up query, then the memory is cleared.
3. Following prompts are modified accordingly after the end of each conversation.
4. This straightforward augmentation proves effective in managing conversational scenarios.

Agenda

01 Inference

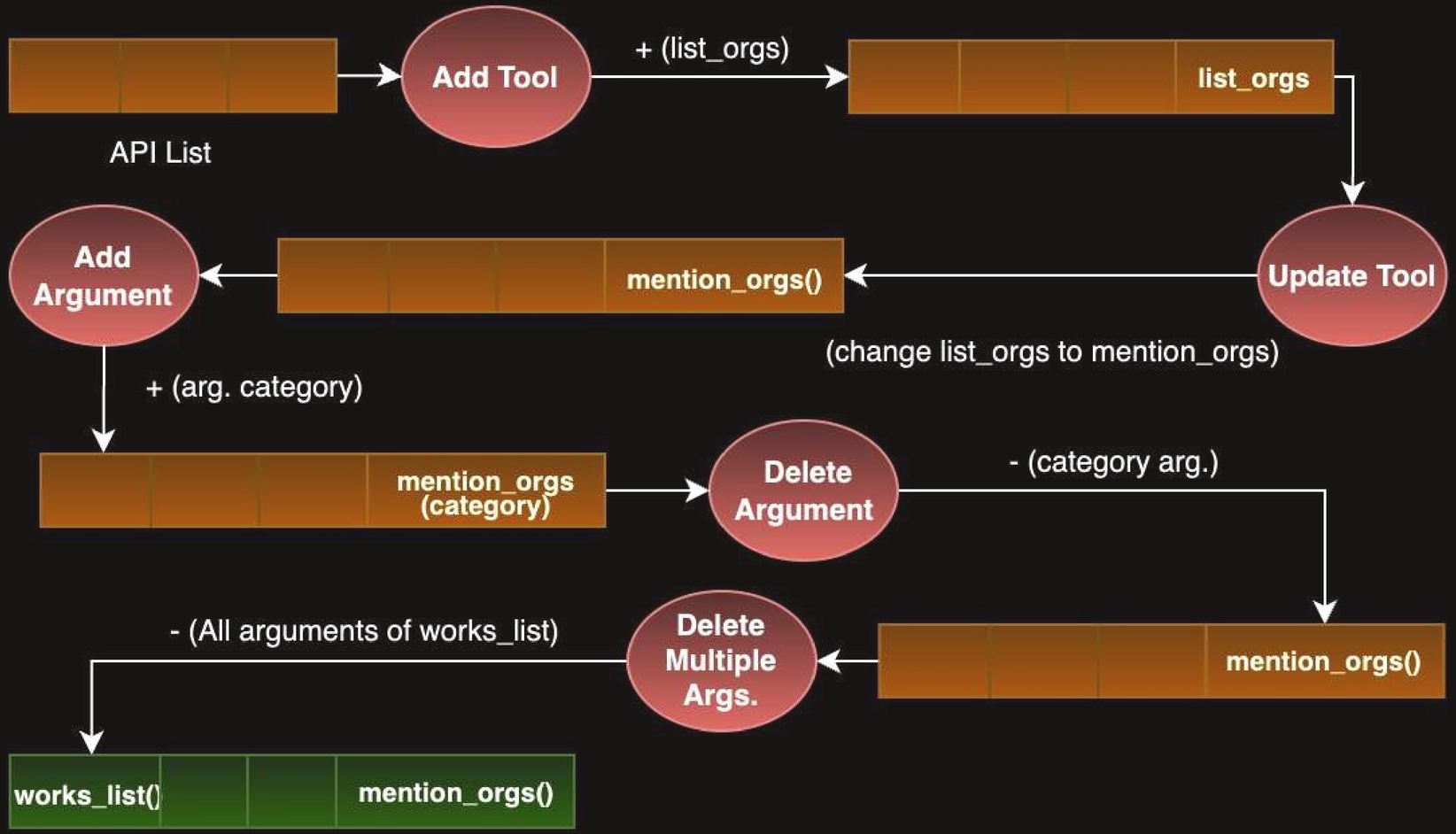
02 Tool Management

03 Data Generation

04 Evaluation

05 Experimentation

Tool Management



Agenda

01 Inference

02 Tool Management

03 Data Generation

04 Evaluation

05 Experimentation

Data Generation

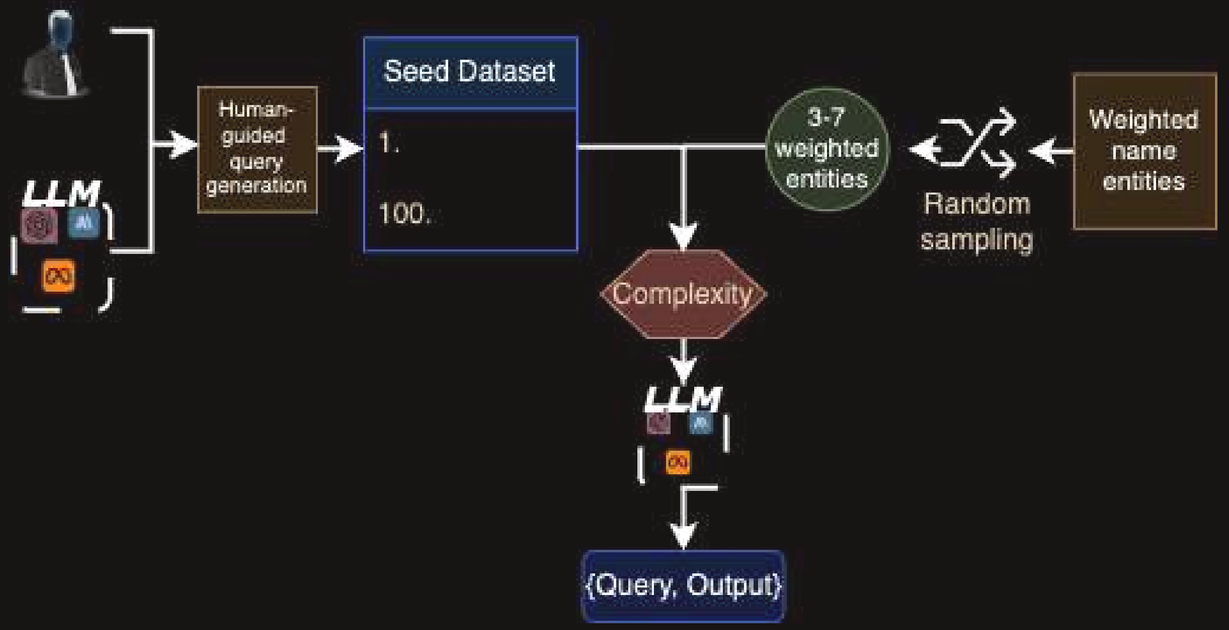
Query Generation

We originally developed a dataset with approximately 500 queries. Here we used the **Human Guided Query Generation** method to develop approximately 150 queries and the rest were automatically generated using **A.G.U.T.E**. Afterwards we used **templating** of entities in existing queries randomly to double our dataset size to approximately 1000 queries.

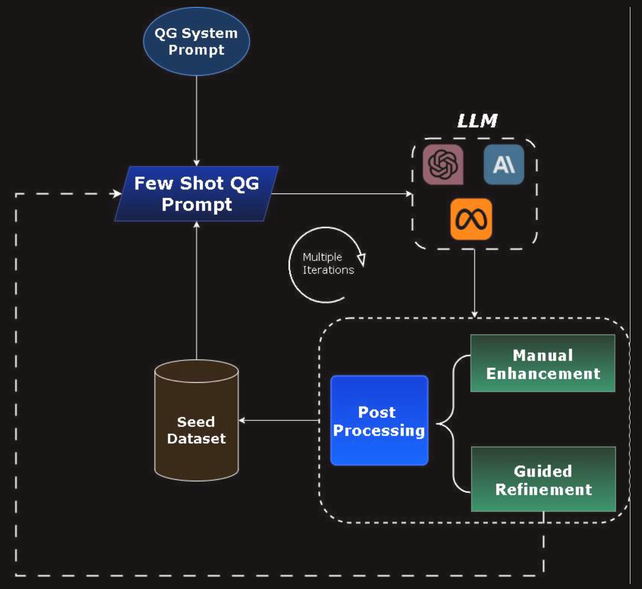
Tool Generation

We observed the emergence of more complex queries that required expanding our toolset. Consequently, as the number of queries grew, we also prompted ChatGPT for suggestions on any new tools that might be necessary to address these queries. Drawing from ChatGPT's responses, we crafted new tools with the required arguments and enhanced existing ones by adding relevant new arguments.

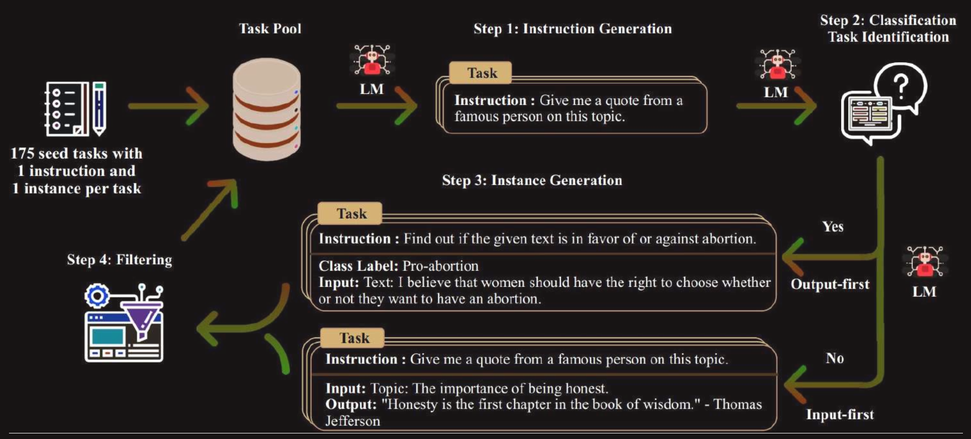
Data Generation



Automated Generation Using Tuned Entities (A.G.U.T.E)

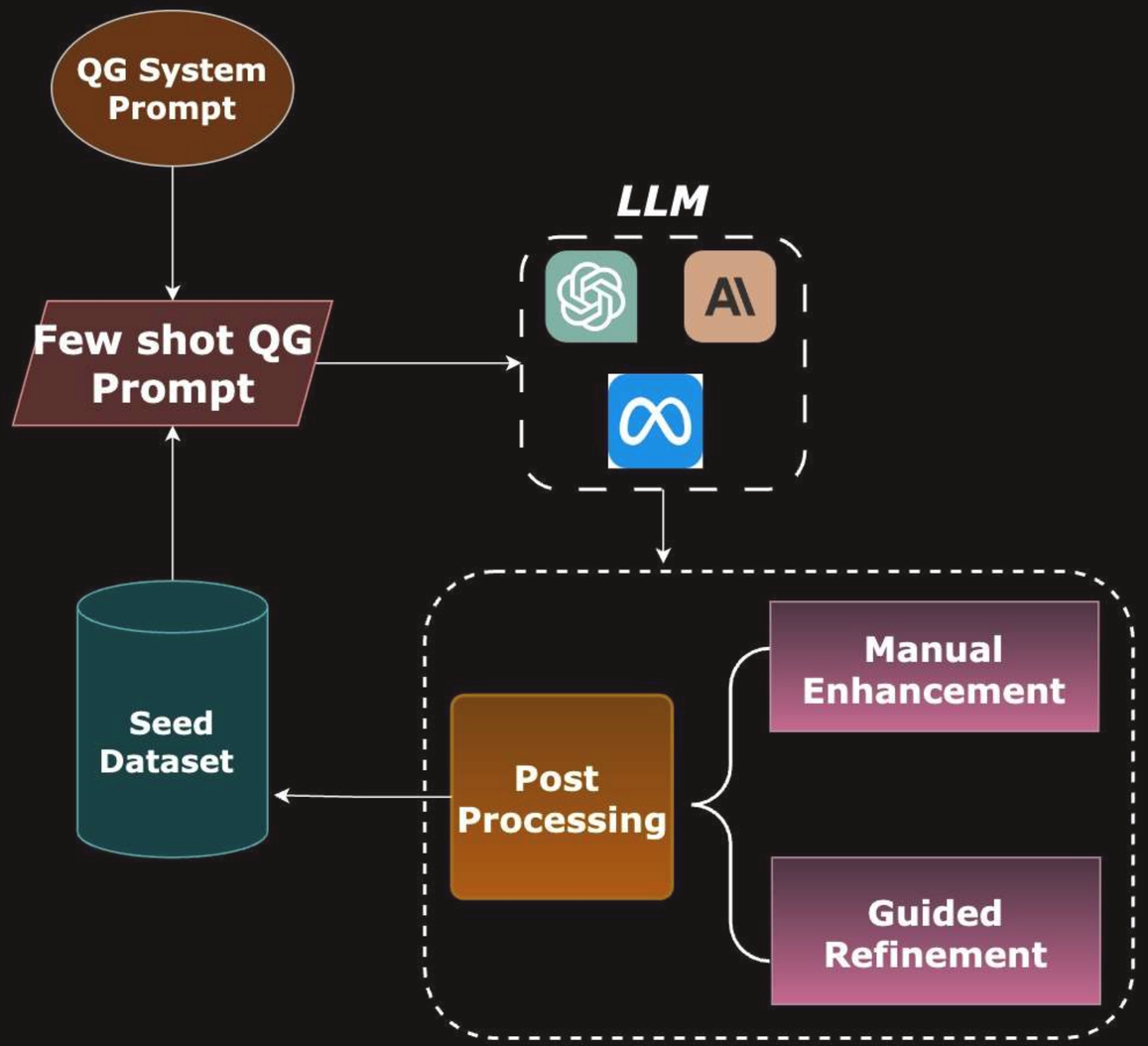


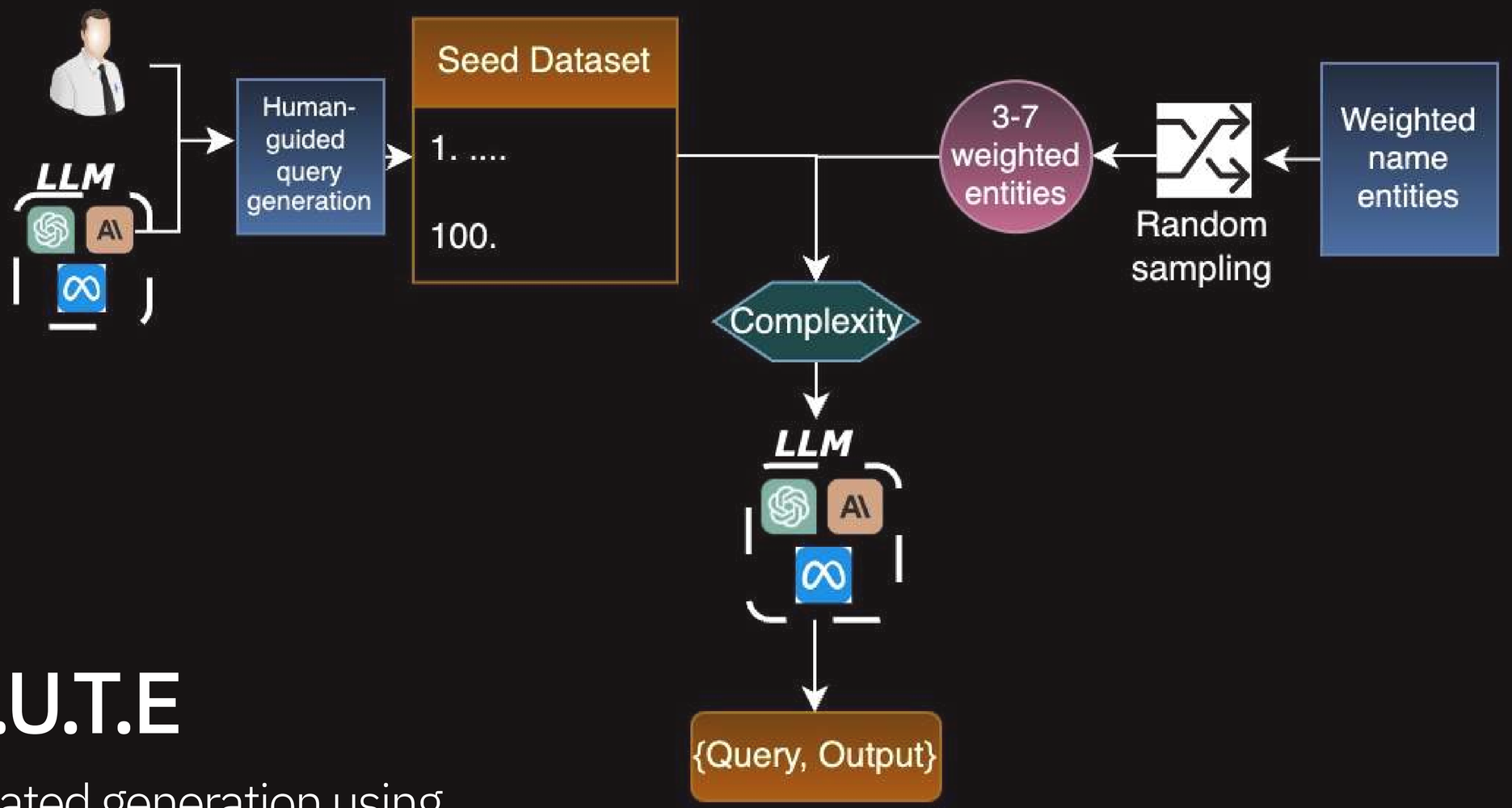
Human Guided Query Generation



Self Instruct

Human Guided Query Generation

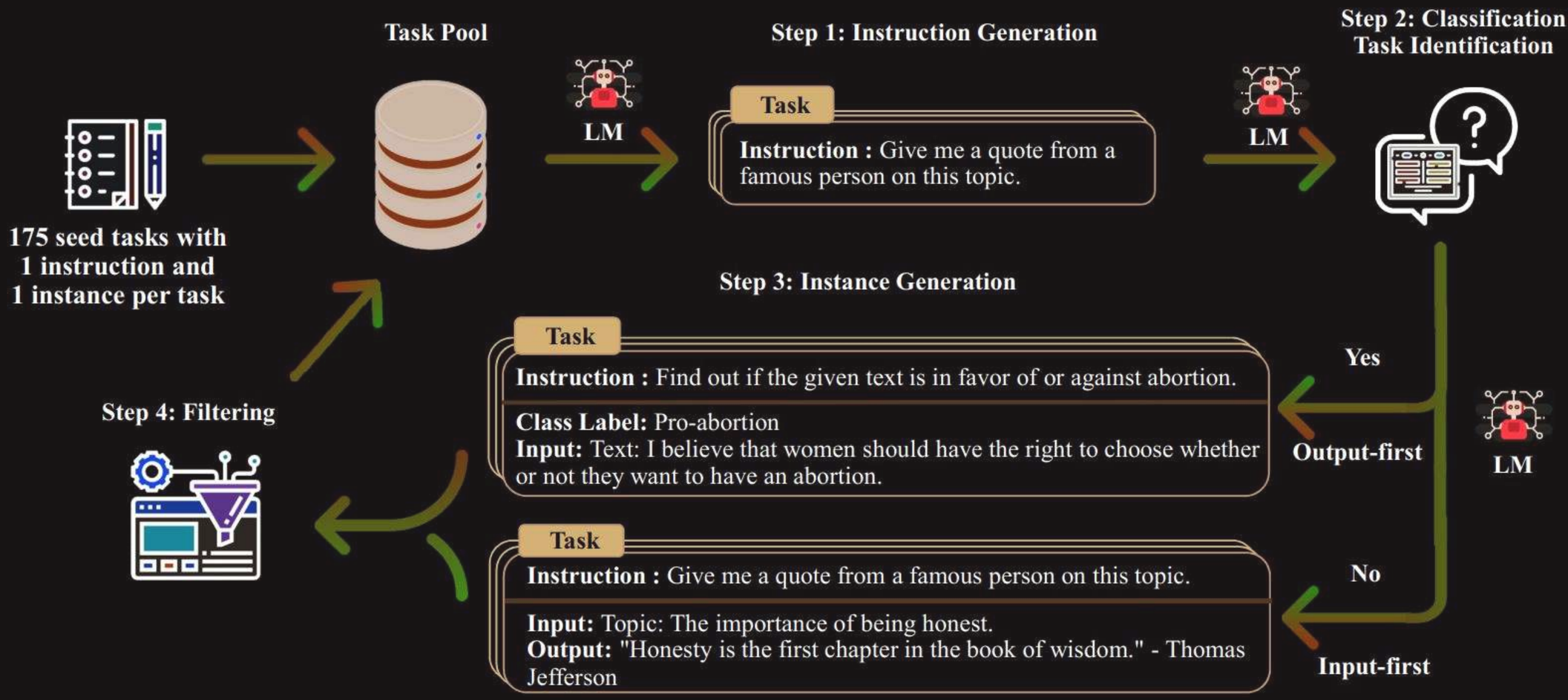




A.G.U.T.E

Automated generation using Tuned Entities

Self Instruct



Agenda

01 Inference

02 Tool Management

03 Data Generation

04 Evaluation

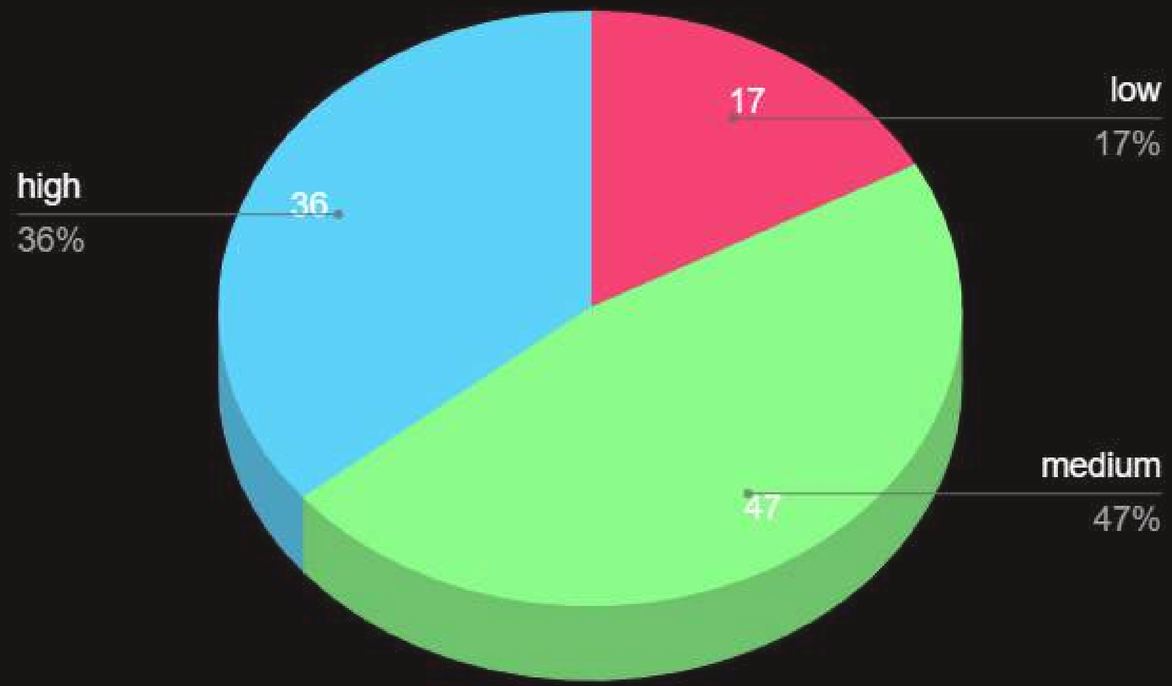
05 Experimentation

Evaluation Approach

API Selection	We ensure that all necessary APIs are utilized to address the query.
Argument Completion	We confirm all used arguments are necessary, leaving no room for oversight in addressing the query.
API Ordering	We check if all APIs are used in the correct order.
Hallucination	We check if the model hallucinates.

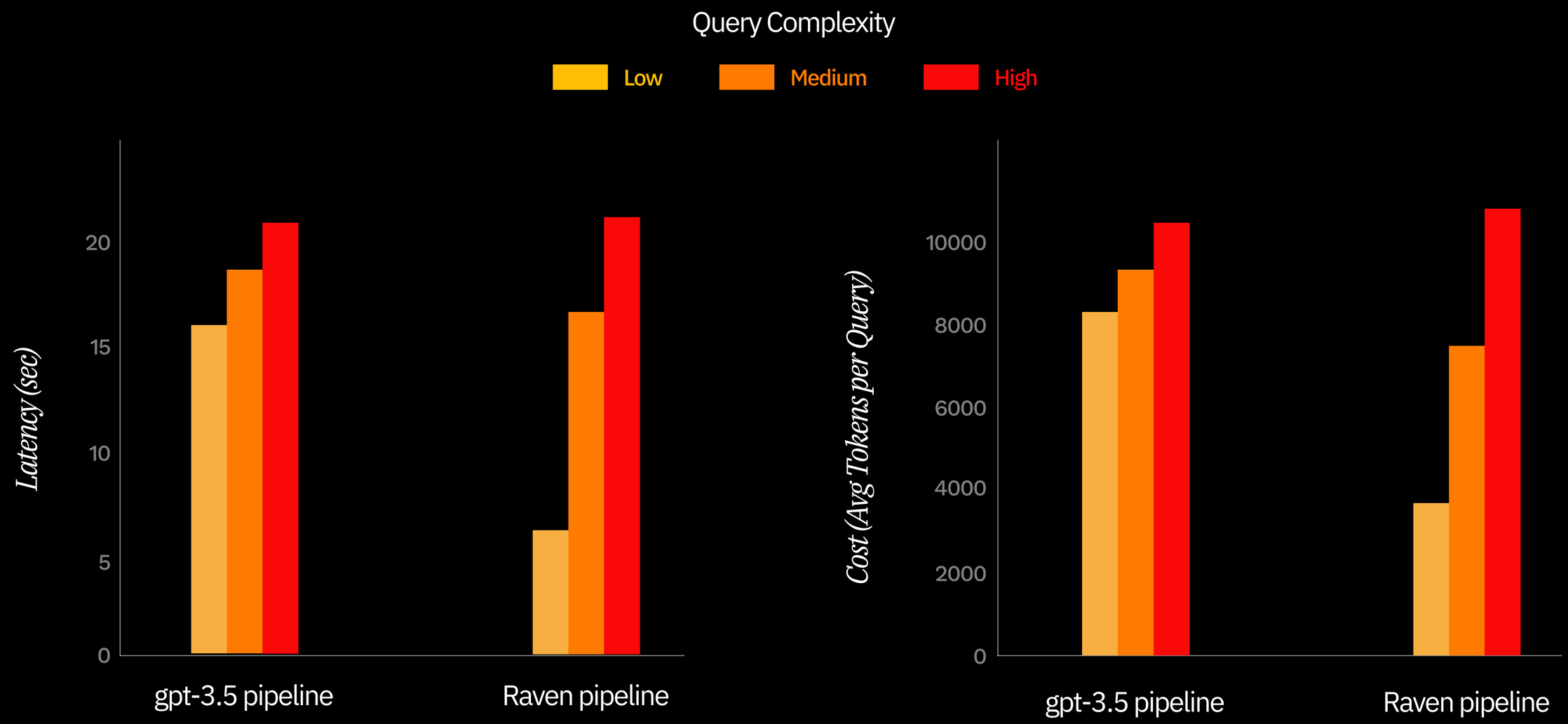
Evaluation Approach

Query Complexity



Query Complexity Distribution

Evaluation Approach



Agenda

01 Inference

02 Tool Management

03 Data Generation

04 Evaluation

05 Experimentation

Experimentation

Prompting Methods

01.
Advanced Prompting Techniques

02.
Emotional Prompting

03.
Chain of Verification

Output Structuring

01.
LMQL

02.
Function Calling

Prompting Methods

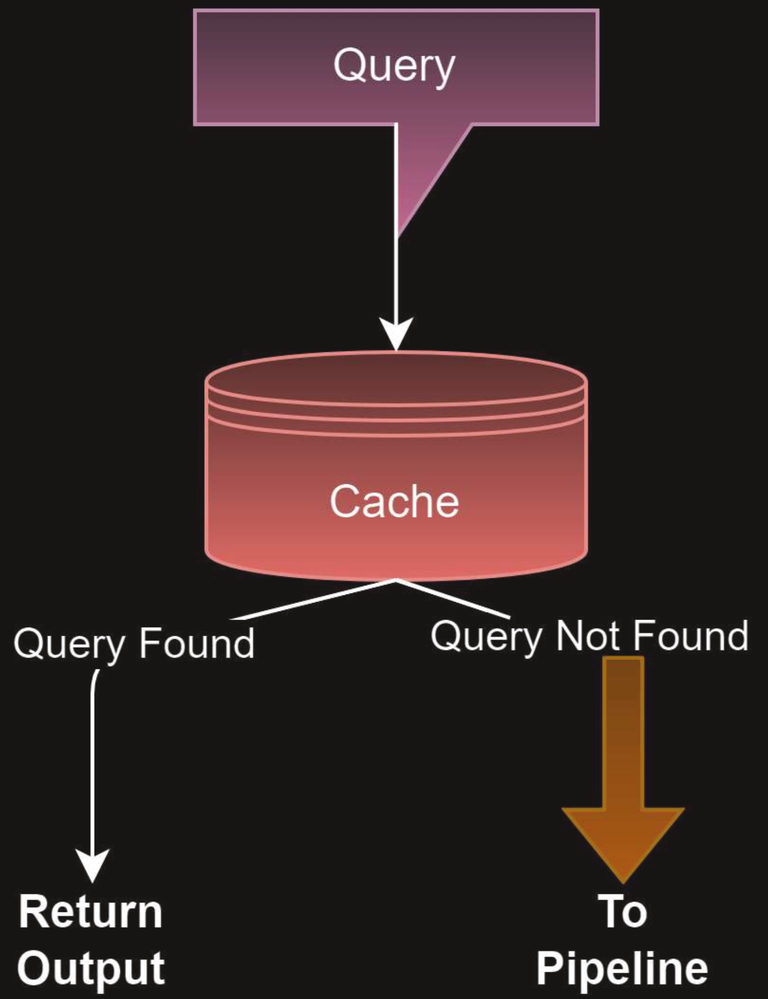
01.
NER

02.
Psuedo-Code Representation

02.
Query Refinement

Experimentation

Cache Inclusion



Experimentation

Fine Tuning

Downstream Tasks

Currently, we utilize LLM(GPT-3.5-Turbo) for simpler tasks like output structuring, classification and validations. Fine tuning smaller LLMs, or even language models, can be one way to handle such tasks more efficiently.

Fine tuning requires appropriate and abundant dataset.

Once data is gathered, we can test various **PEFT** methods on the downstream tasks. This approach would maximize the efficiency of the whole pipeline, as expensive LLM’s job would be reduced to only reasoning behind tool applications.

Thank You

Team 15