



LLM for Data Management

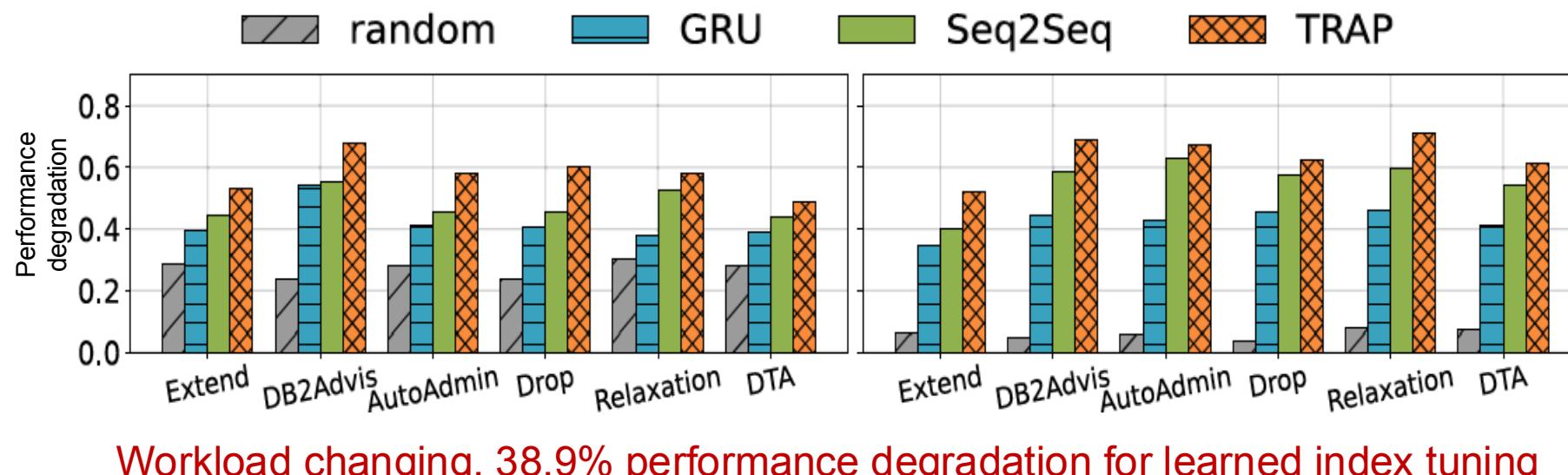
Up-To-Date Paper List

<https://github.com/code4DB/LLM4DB>

Challenges of Traditional AI4DB

□ Challenges in Traditional AI4DB

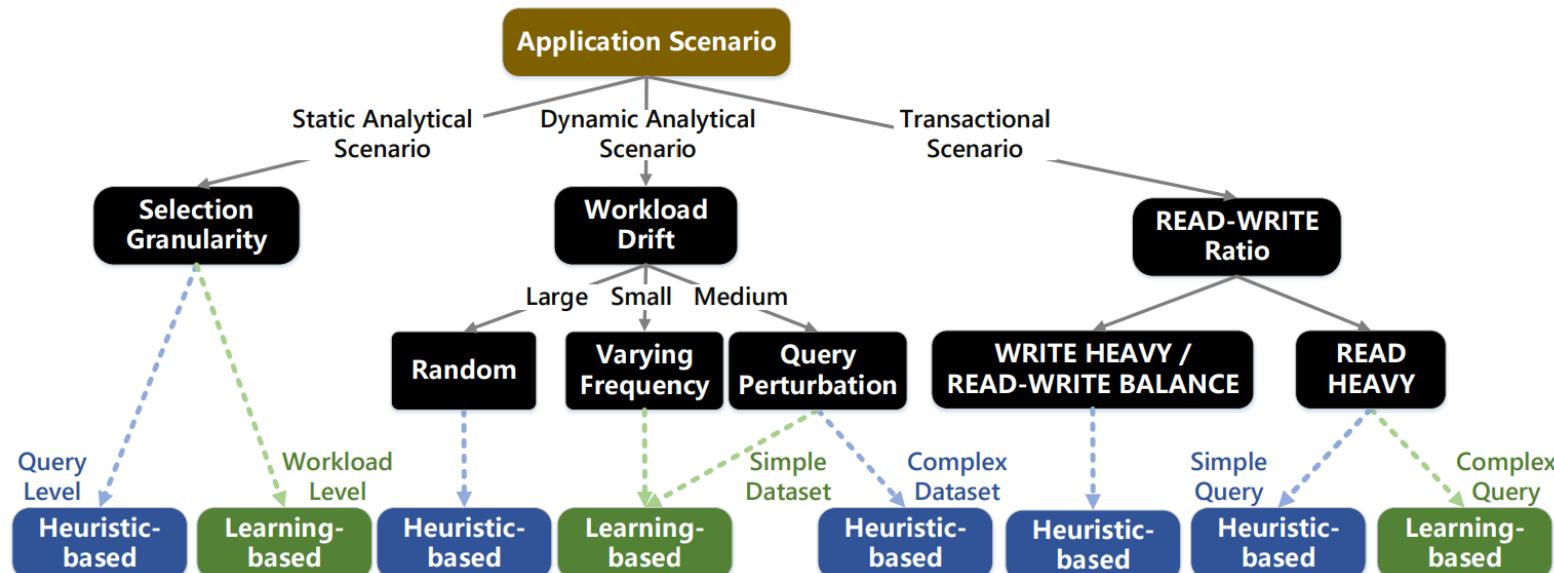
- Dynamic changing **schema**
 - Dynamic changing **data**
 - Dynamic changing **workload**
 - Dynamic changing **hardware**
- **High-quality** training data
 - **Poor** generalization
 - **Poor** interpretability



Challenges of Traditional AI4DB

□ Challenges in Traditional AI4DB

- Dynamic changing schema
 - Dynamic changing data
 - Dynamic changing workload
 - Dynamic changing hardware
- High-quality training data
 - Poor generalization
 - Poor interpretability

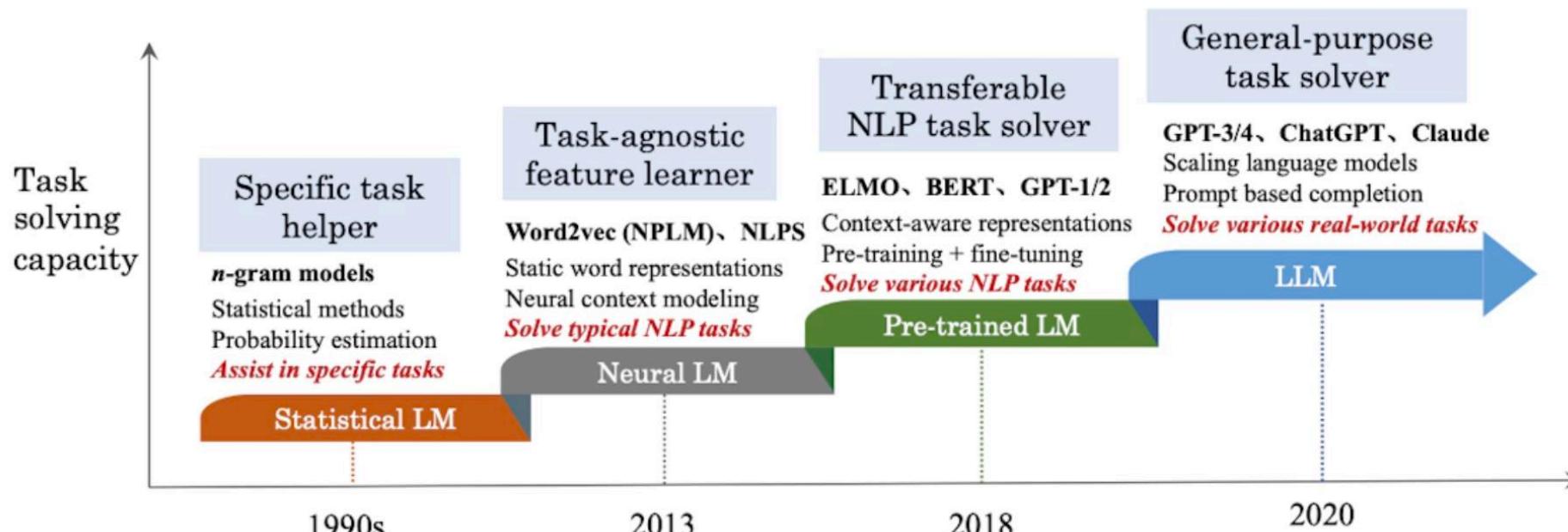


The Applicable Scenarios of Traditional Learned Methods are Limited!

Motivations of LLM for Data Management

□ Excellent performance and generalization capability

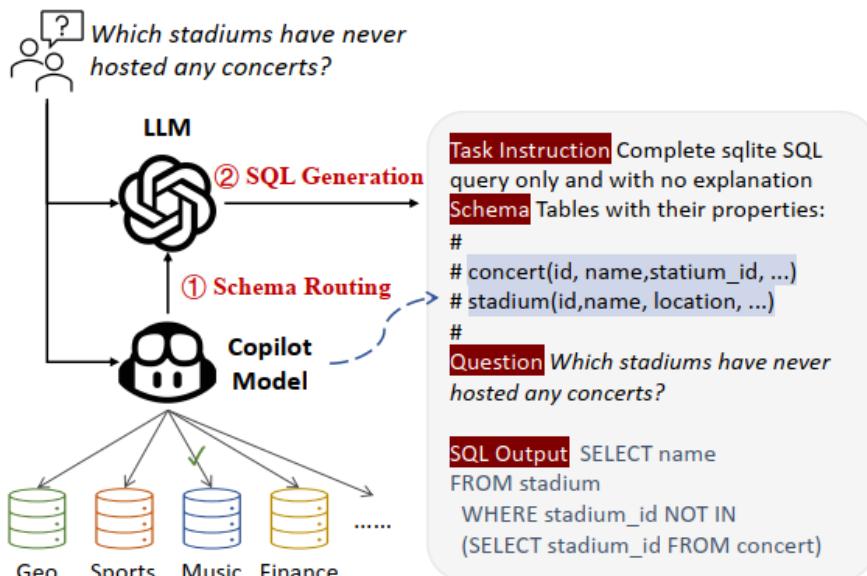
- Support various natural and programming languages
- Improved reasoning ability (v.s. traditional PLM)
- Solve various real-world tasks (e.g., coding, report writing)



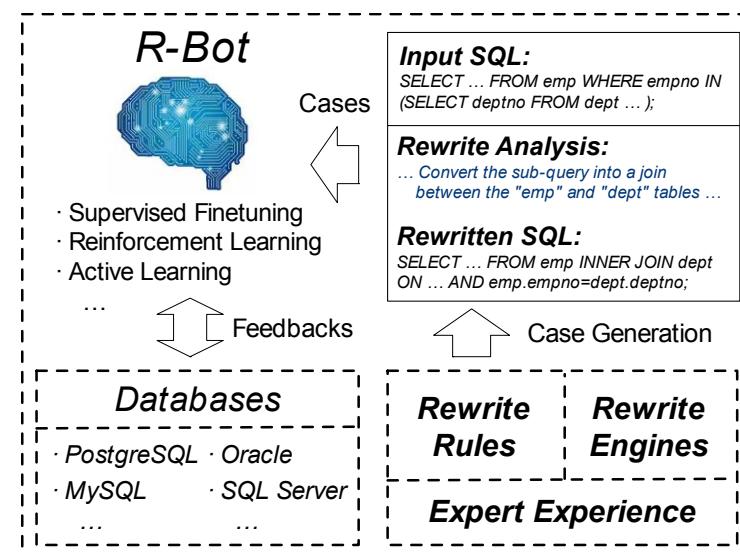
Motivations of LLM for Data Management

□ Opportunities of LLM for data management

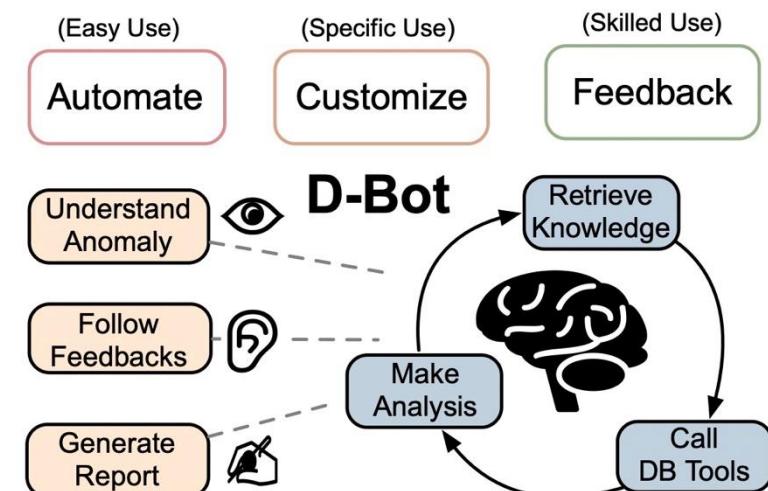
- Improved usability (e.g., Text2SQL)
- Improved performance (e.g., Query Rewrite)
- Improved maintainability (e.g., Diagnosis)
- Improved interpretability (e.g., Database Tuning)



Text2SQL

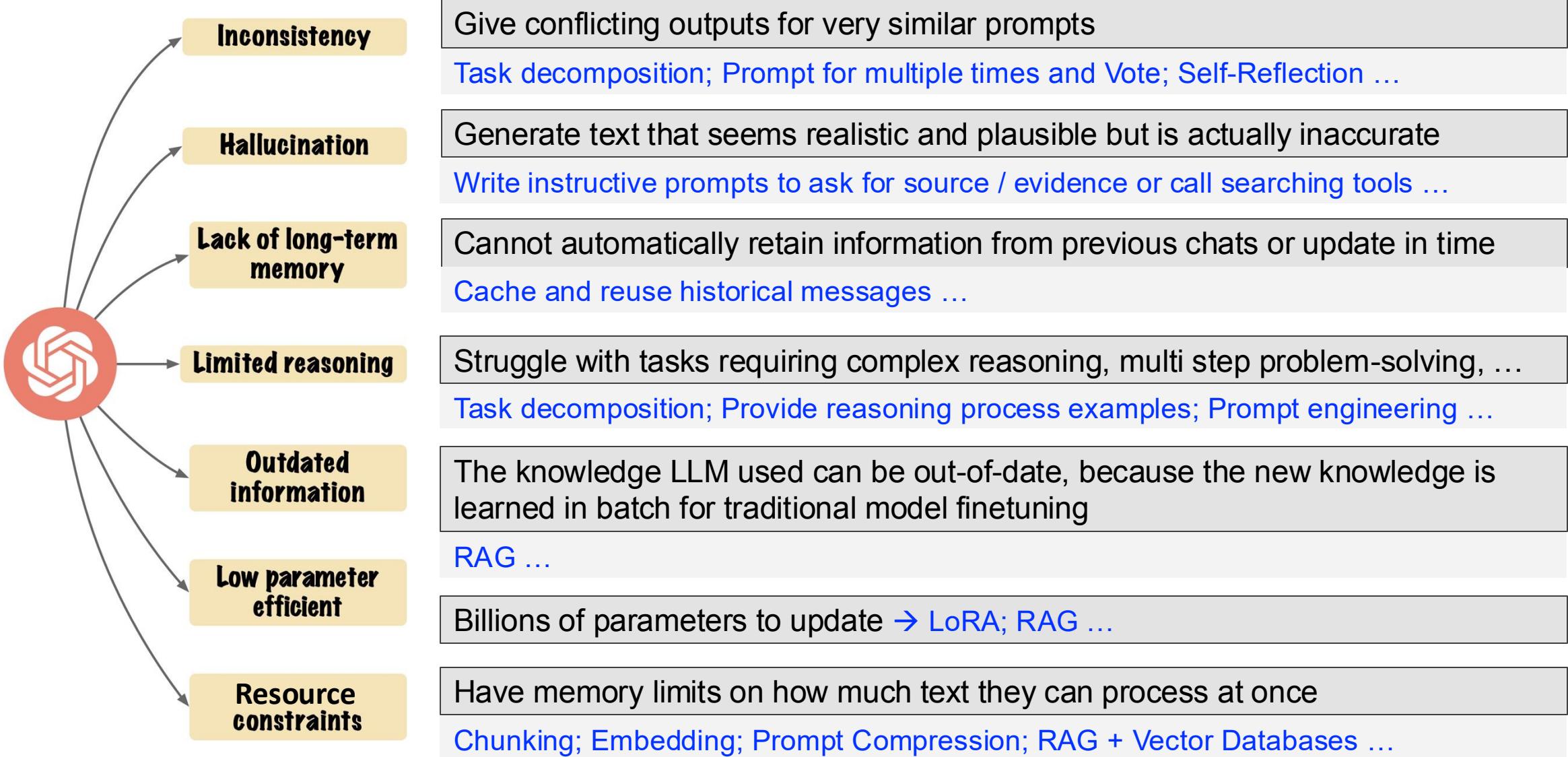


Query Rewrite



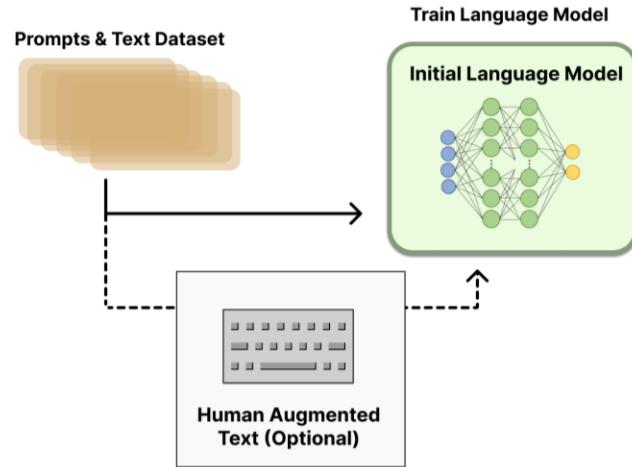
Diagnosis

Overview of LLM Challenges and Solutions



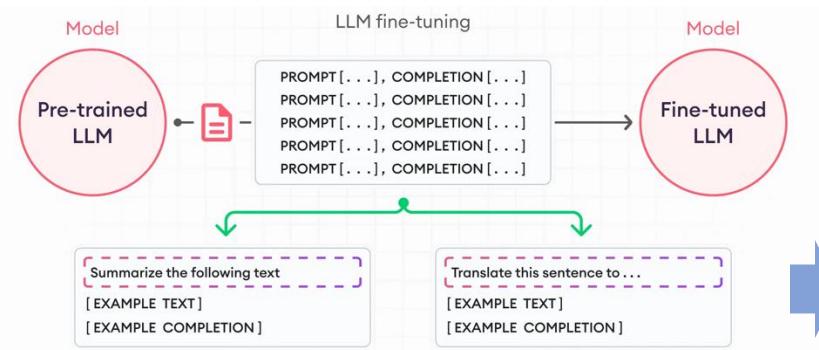
Different Stages of LLM

1. (Incremental) Pretraining



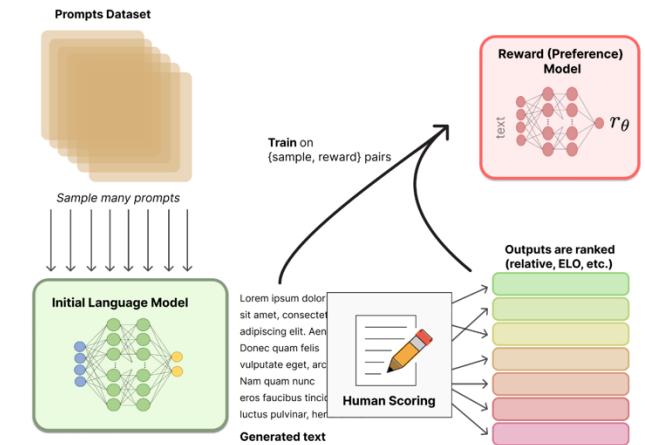
- Common Knowledge Acquisition
- Understanding Diverse Texts

2. (SFT) Finetuning



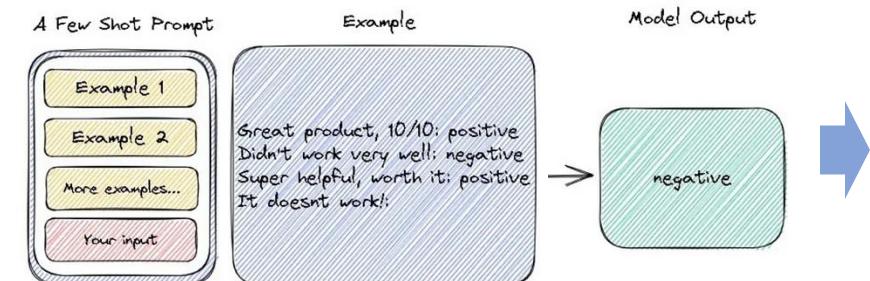
- Instruction Following
- Task Adaption like Translation/Q&A

3. (RLHF) Finetuning



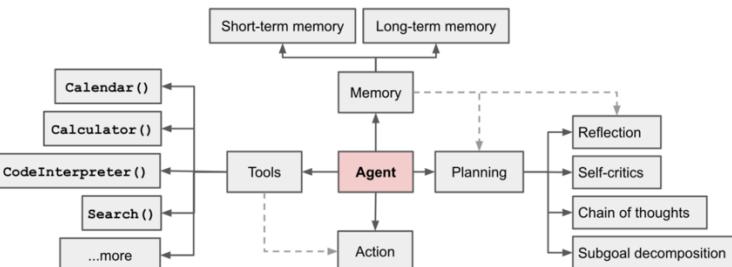
- Align with human preferences

4. Prompting



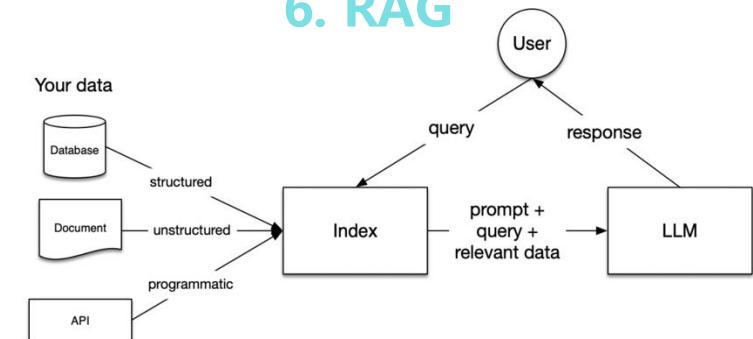
- Context Comprehension
- Learn from demo examples

5. Agent



- LLM system equipped with reasoning, tools, and memory

6. RAG



- External Knowledge Integration
- Contextual Relevance / QA Accuracy

LLM Pre-Training

□ Pretrain LLM as the foundation model for database

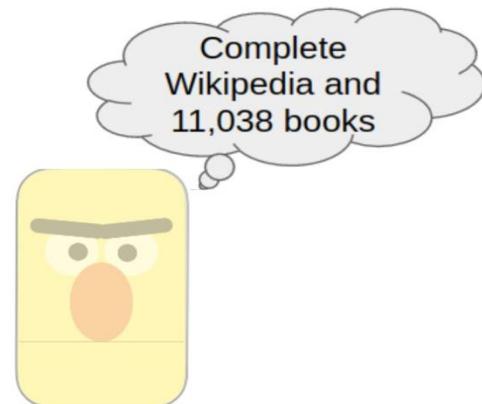
Tasks

Text corpus



(Self-supervised)
Training

Pretrained LM



Adaptation

Question Answering



Text Classification



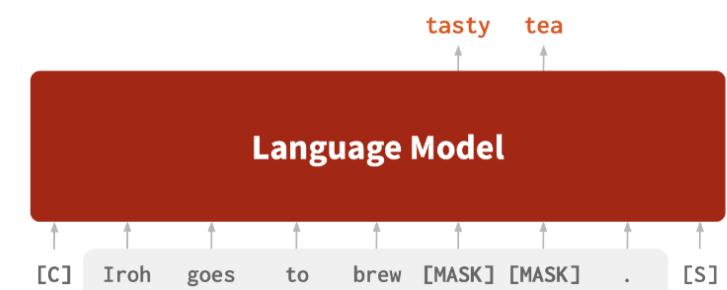
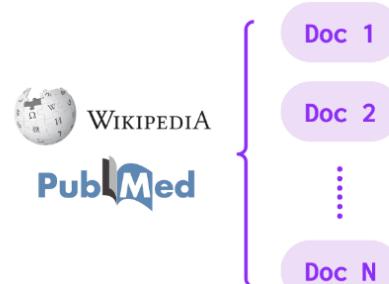
Information Retrieval



Transformer-based LLM:
Predict the next word
given a sequence of
previous text



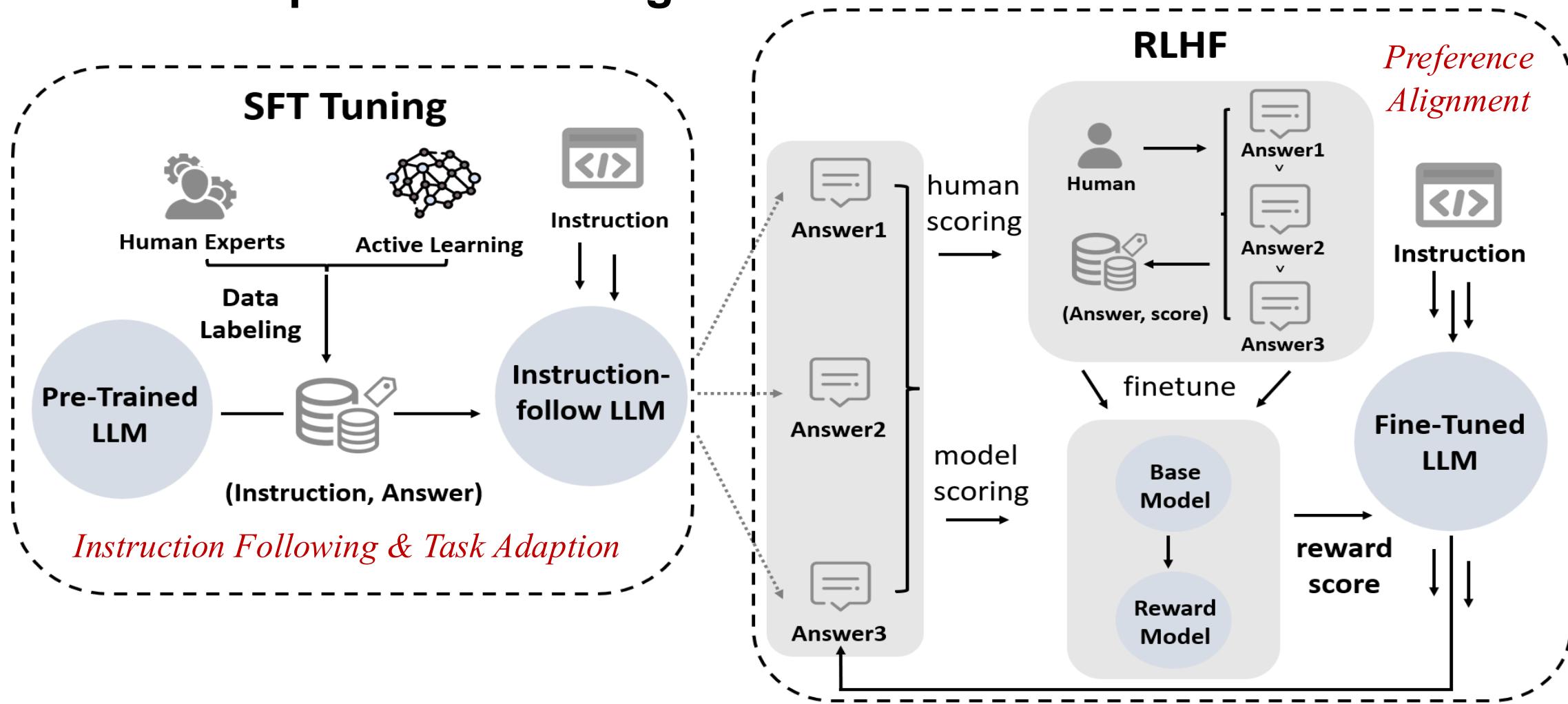
List of documents



Pretrain the LM

LLM Finetuning

- Finetune LLM over labeled dataset to learn instruction-following and task-specific knowledge



Prompt for LLM Inference

□ Input text for LLM to generate response or execute a task

- **Simple Prompt**

- (task) "Explain the theory of relational tables."

- **Contextual Prompt**

- (context) "Undergraudate students are studying database concepts for the first time and is curious about fundamental theories."
 - (task) "Explain the theory of relational table in a way beginners can understand."

- **Contextual Prompt with Instructions**

- (context) " Undergraudate students are studying database ..."
 - (task) "Explain the theory of relational tables ..."
 - (instructions) "1. Make sure the explanation is clear and engaging for someone new to physics; \n\n 2. Limit the explanation to a few paragraphs with examples."

- **Contextual Prompt with Instructions + Demonstration Examples ...**

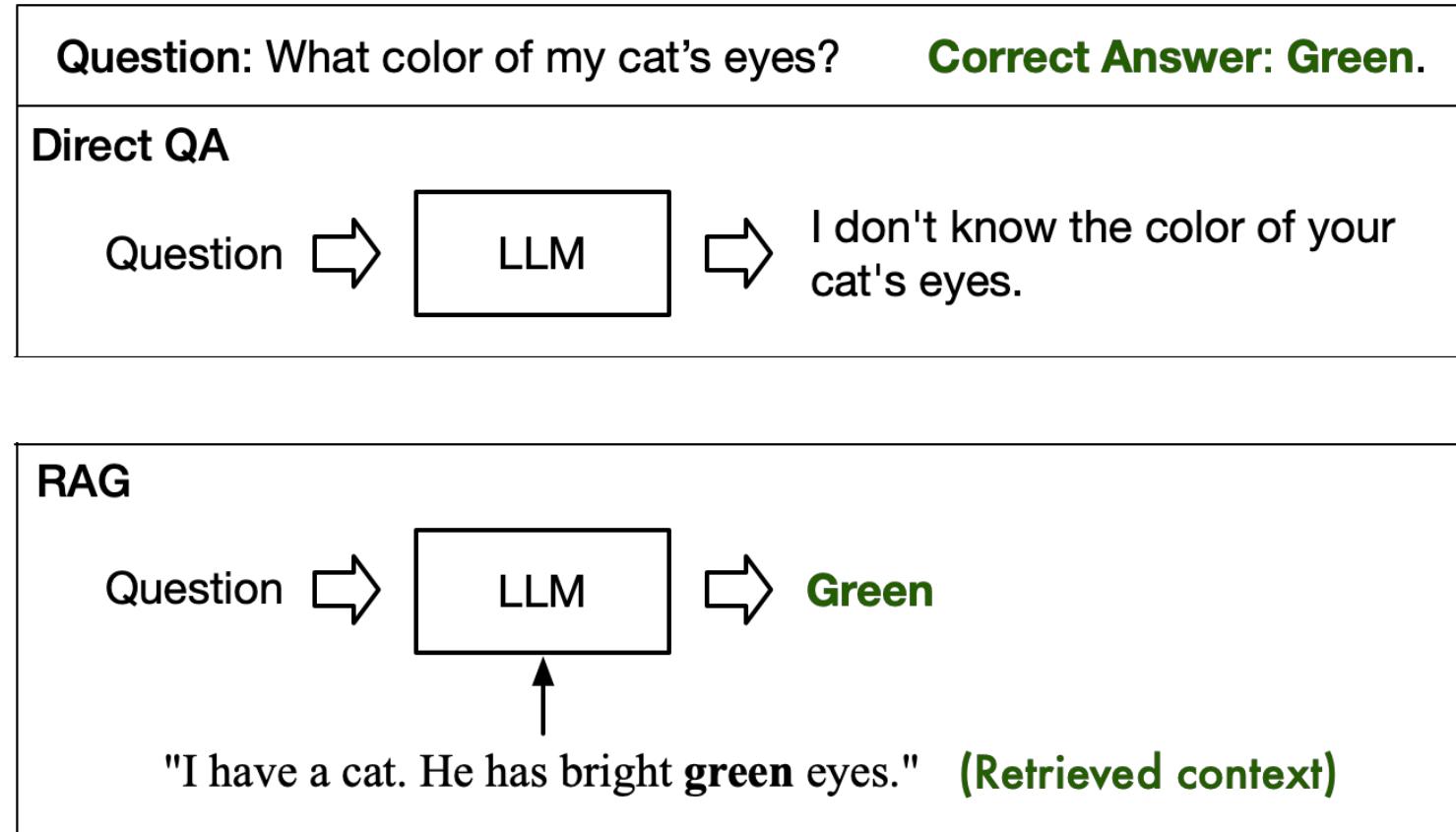
RAG for LLM Inference

□ Drawbacks of LLMs

- Hallucination
- Outdate information
- Low efficiency in LLM training
- Weak reasoning capability

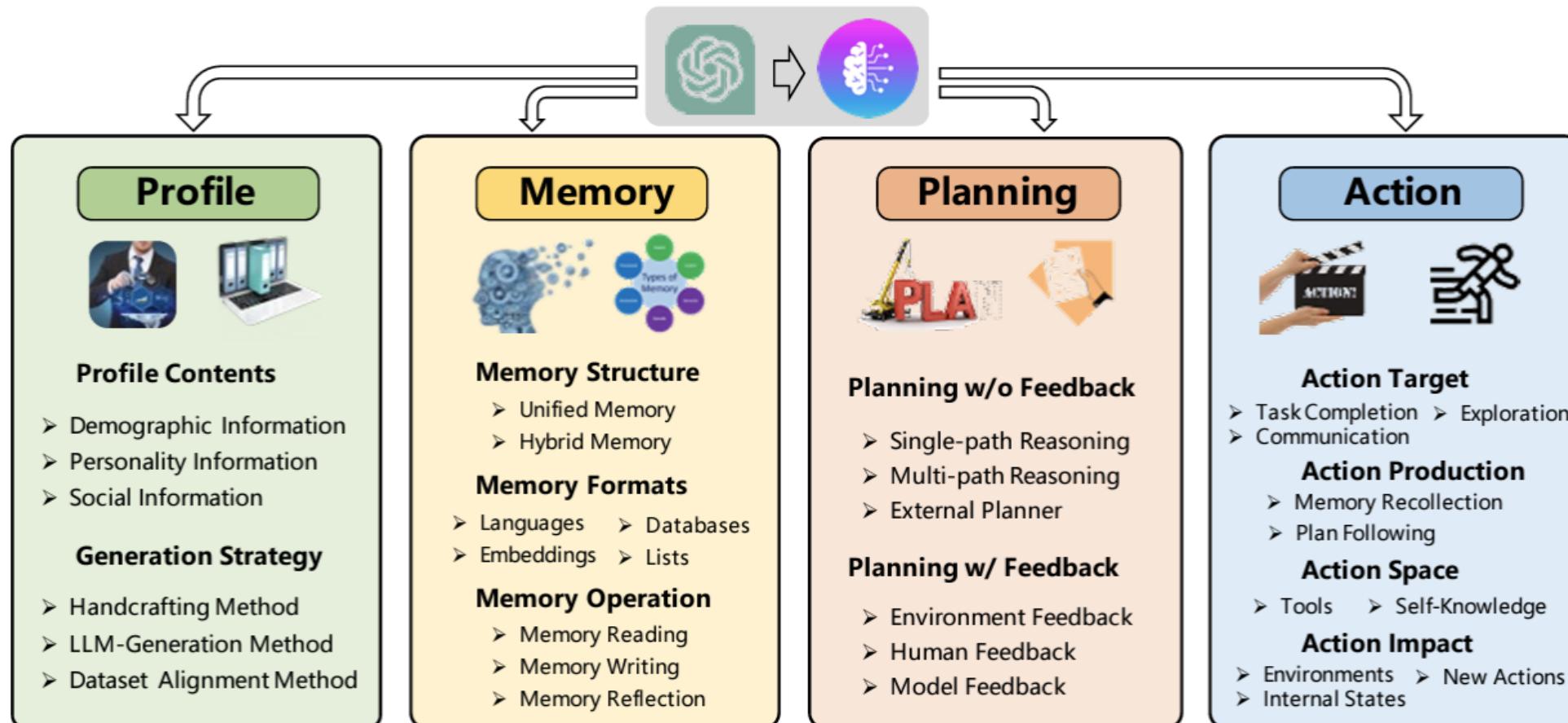
□ Practical Requirements

- Domain-Specific Accurate Q&A
- Frequent Data Update
- Explainability of Responses
- Controllable Cost
- Data Privacy Protection



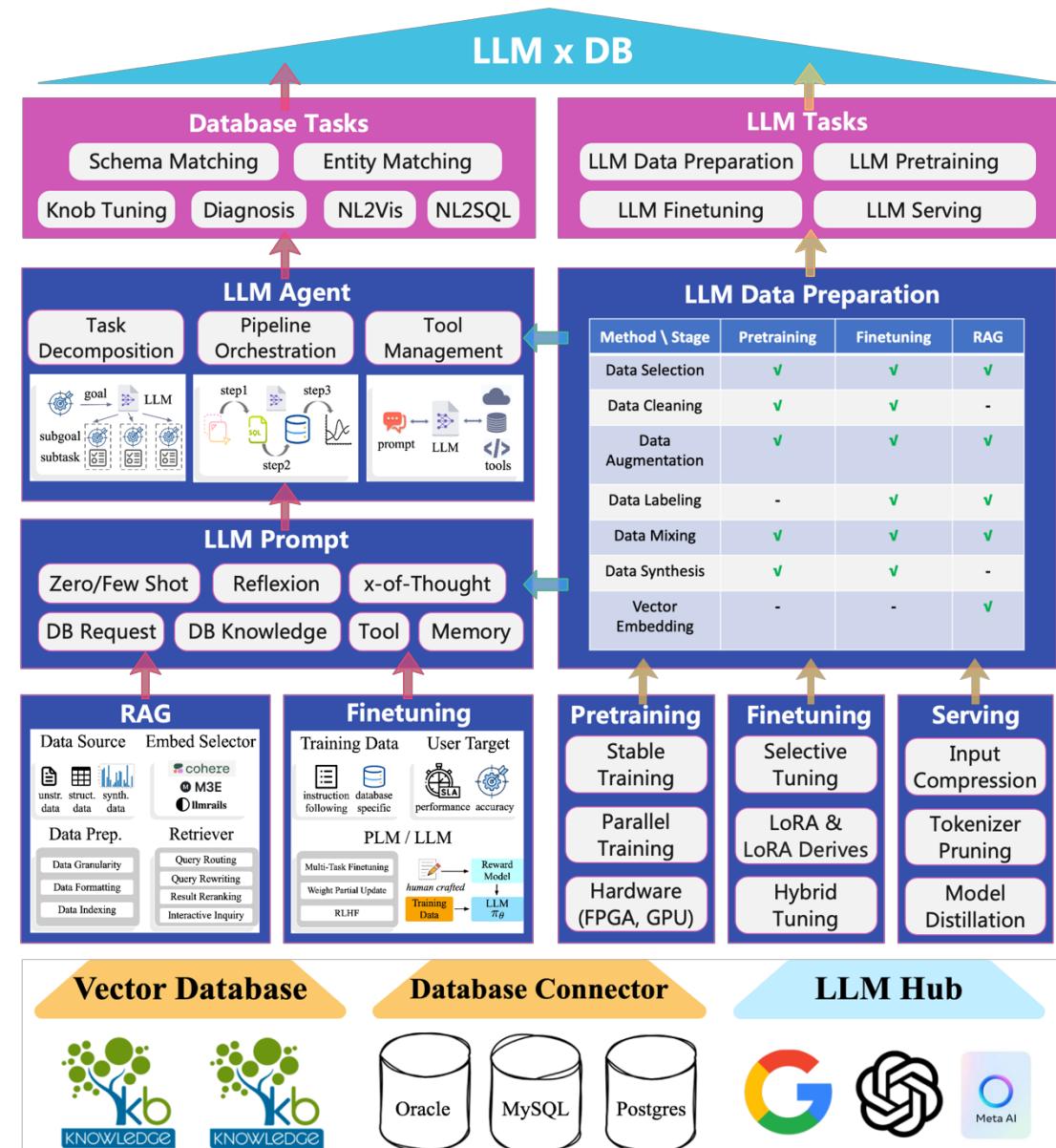
LLM Based Autonomous Agent

- LLM Agent: **Perceiving** the surrounding environment, **planning**, **executing** actions to complete tasks, and **memorize** past executions



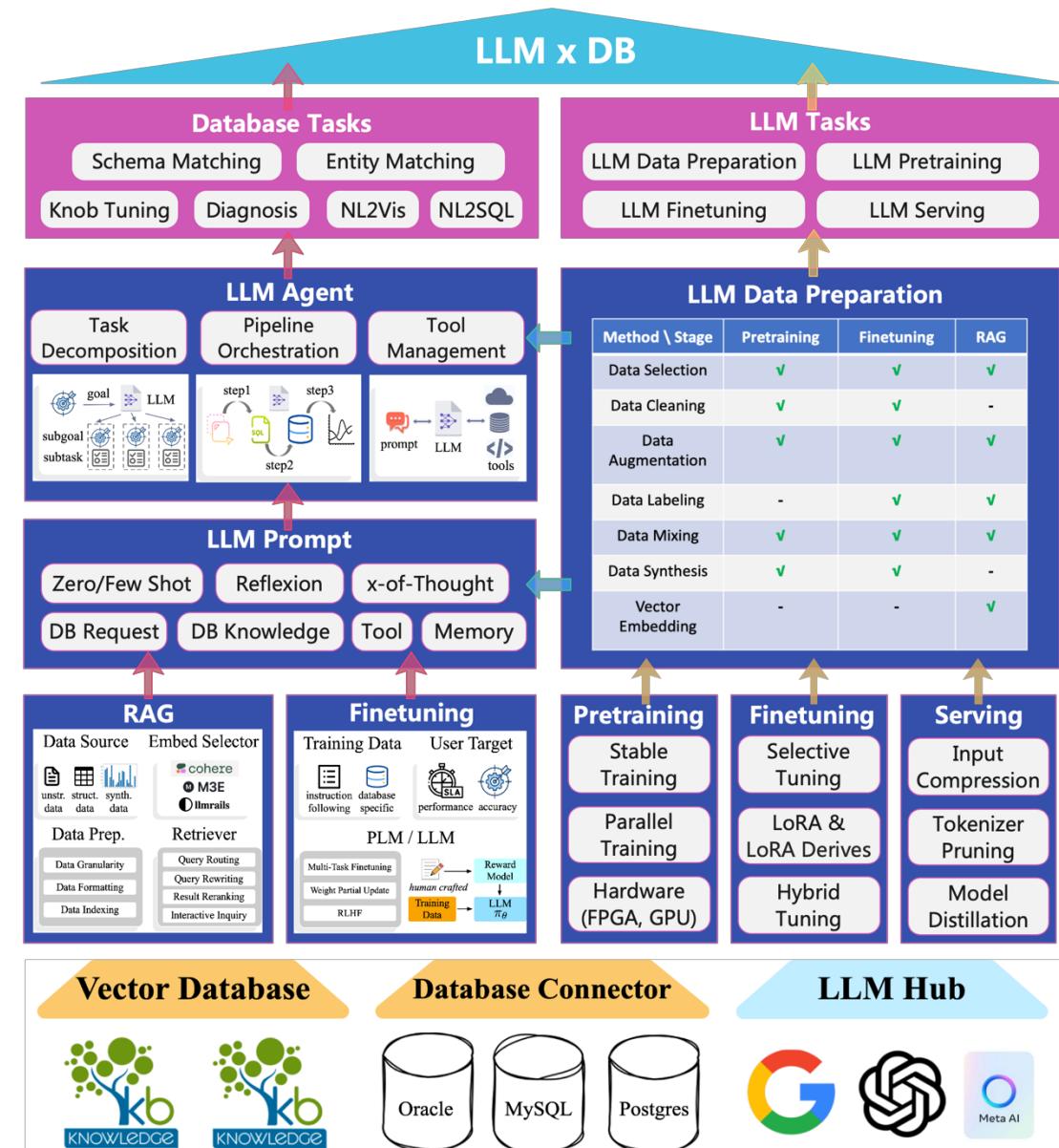
Outline of LLMxDB

- Data Management tasks
- LLM Prompt for Data Management
 - Instruction Prompting
 - Few-Shot Prompting
- LLM Agent for Data Management
 - Agent Models + Memory
 - Reasoning / Planning Strategies
 - Tool Management & Learning
- RAG for Data Management
 - Semantic Segmentation
 - Result Retrieval
 - Result Reranking
- Finetuning for Data Management
 - SFT Dataset Generation
- Data Management for LLM
 - LLM Data Preparation



Outline of LLMxDB

- Data Management tasks
- LLM Prompt for Data Management
 - Instruction Prompting
 - Few-Shot Prompting
- LLM Agent for Data Management
 - Agent Models + Memory
 - Reasoning / Planning Strategies
 - Tool Management & Learning
- RAG for Data Management
 - Semantic Segmentation
 - Result Retrieval
 - Result Reranking
- Finetuning for Data Management
 - SFT Dataset Generation
- Data Management for LLM
 - LLM Data Preparation



Overview of LLM for Data Management Tasks

□ Data Management Tasks

➤ Data Preprocessing

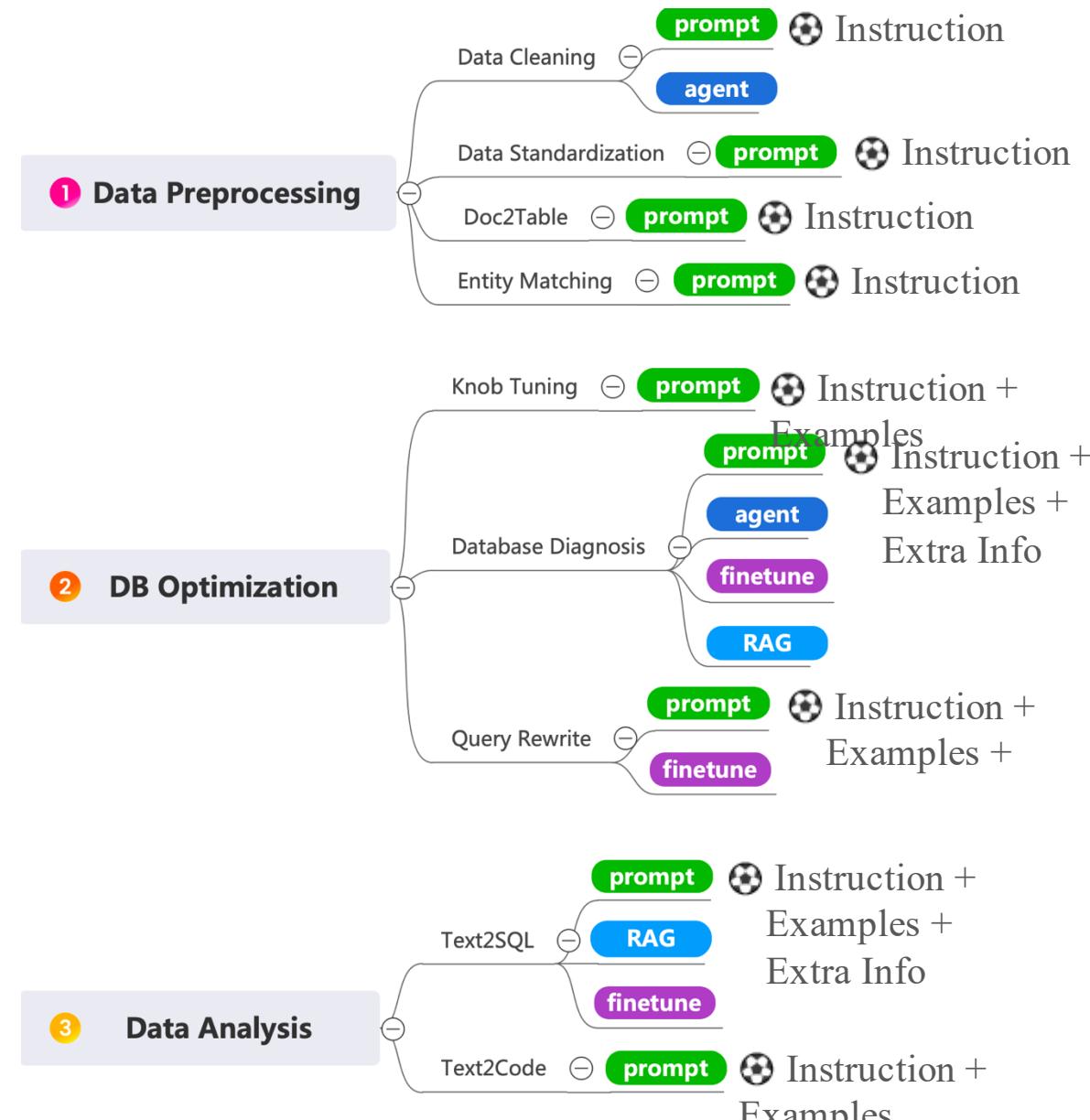
- Data cleaning
- Data Standardization
- Doc2Table
- Entity matching

➤ Database Optimization

- Knob tuning
- Database diagnosis
- Query rewrite

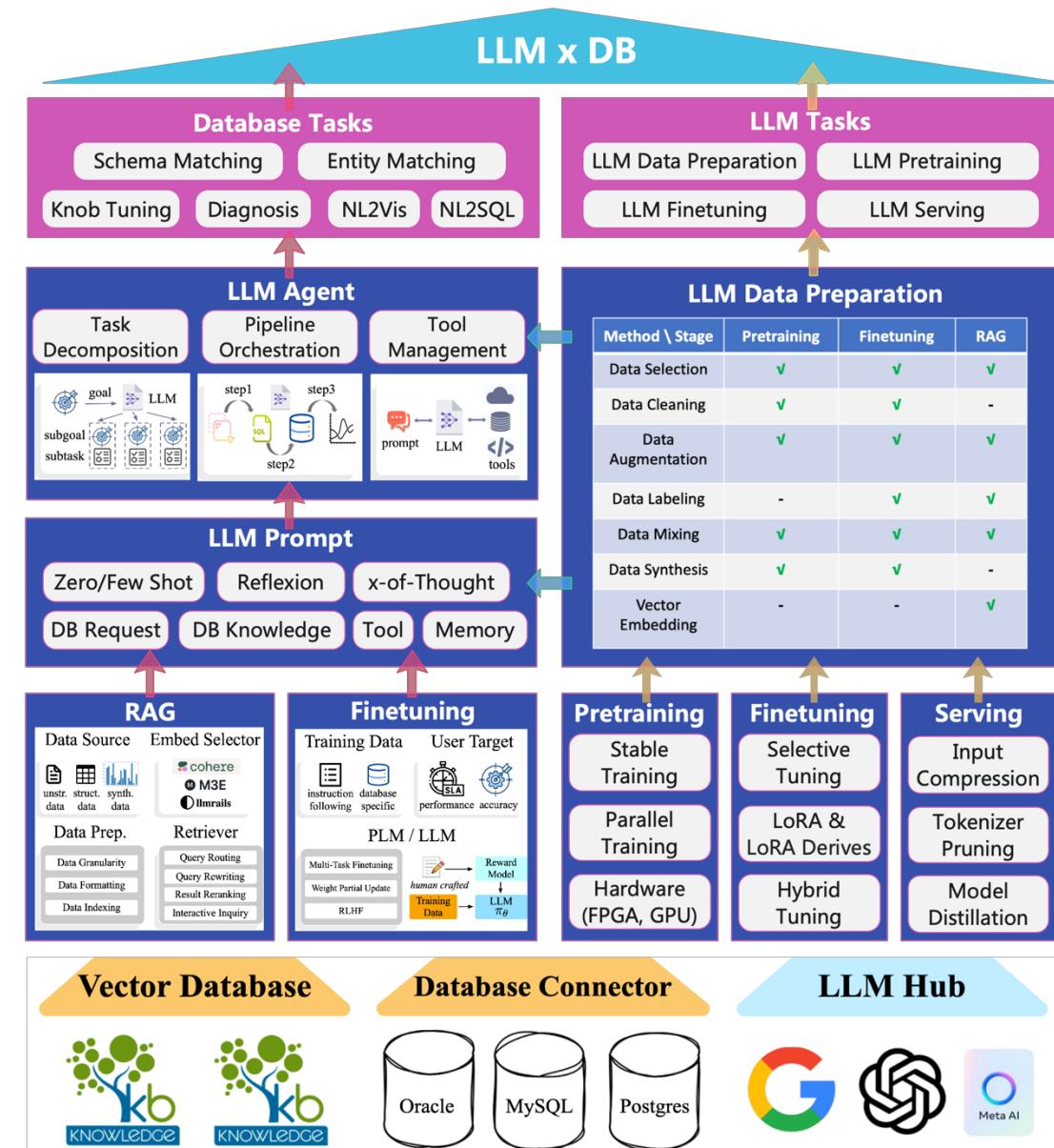
➤ Data Analysis

- Text2SQL
- Text2Code
- ...



Outline of LLMxDB

- Data Management tasks
- LLM Prompt for Data Management
 - Instruction Prompting
 - Few-Shot Prompting
- LLM Agent for Data Management
 - Agent Models + Memory
 - Reasoning / Planning Strategies
 - Tool Management & Learning
- RAG for Data Management
 - Semantic Segmentation
 - Result Retrieval
 - Result Reranking
- Finetuning for Data Management
 - SFT Dataset Generation
- Data Management for LLM
 - LLM Data Preparation



Prompt Engineering (PE)

□ High-Quality Prompt can instruct LLM to optimize DB tasks without training

➤ Zero-shot Prompting

- Input LLM with a **task description**, without training over labeled data

• Instruction Prompting

- Input LLM with **explicit instructions** on approaching the task, e.g., detailing the format, tone, or type of output response

➤ Few-shot Prompting

- Provide LLM with a few **examples** of the task within the prompt to guide the model on how to reason and generate responses

Prompt of Query Rewrite

Task Description

Write an equivalent SQL query that can be executed on a Postgres database with decreased latency.

Instruction

1. Ensure output query is semantical-equivalent to the input query ...

Example Input

```
select ... from t1 where t1.a=(select avg(a) from t3 where t1.b=t3.b);
```

Example Output

```
select ... from t1 inner join (select avg(a) avg,t3.b from t3 group by t3.b) as t3 on (t1.a=avg and t1.b=t3.b);
```

Input

```
select t1.* from t1 where t1.col1>(
    select max(t2.col2) from t2 where t2.col1 in (
        select t1.col1 from t1 where t1.col1=t2.col1));
```

Output

```
select t1.* from t1 inner join (
    select max(t2.col2) max, t2.col1 from t2
    group by t2.col1) as t2 on (
        t1.col1=t2.col1)
where t1.col1>max;
```

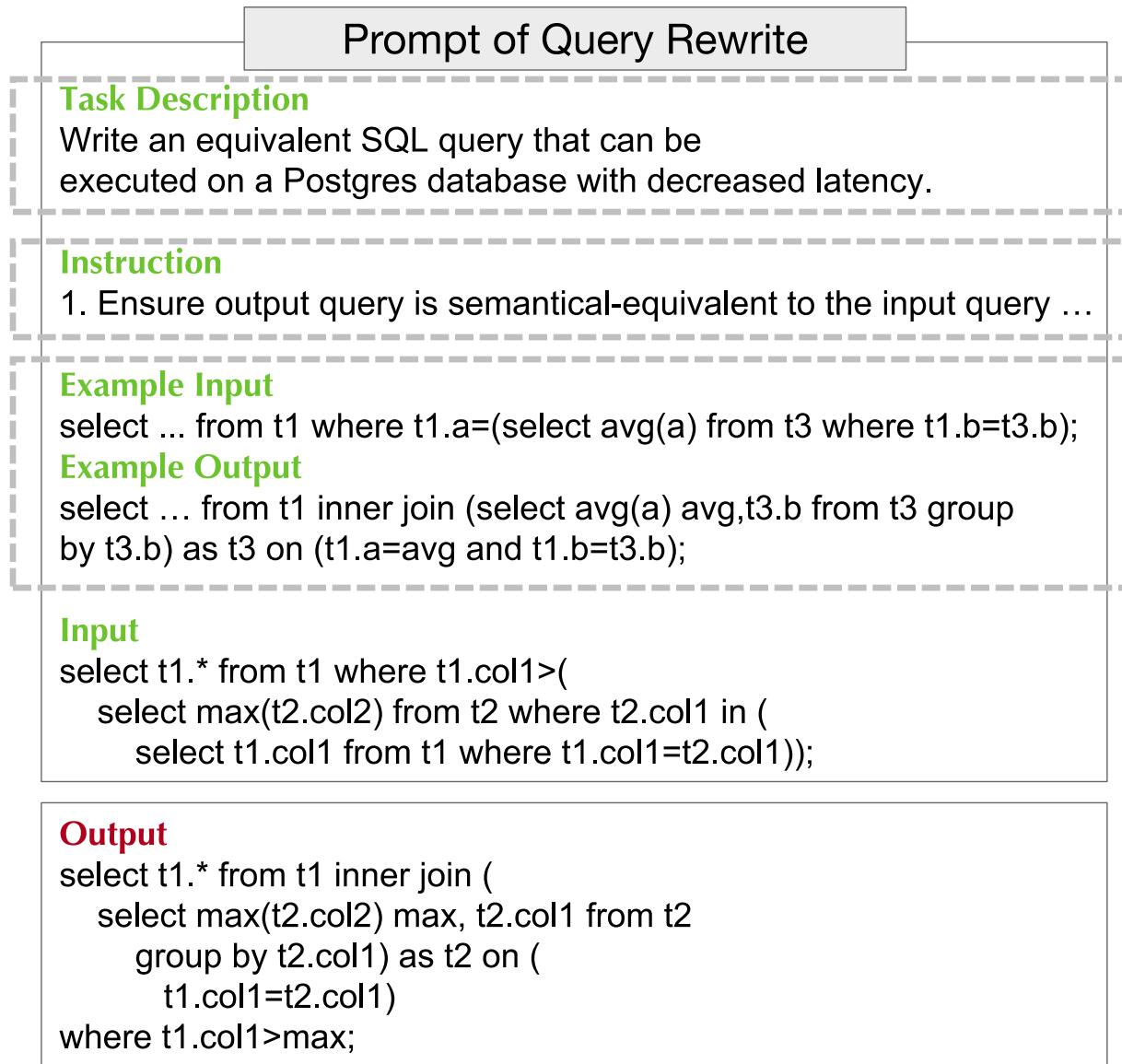
Challenges of PE for Data Management

□ Challenge 1: How to Automatically Generate Input Prompt?

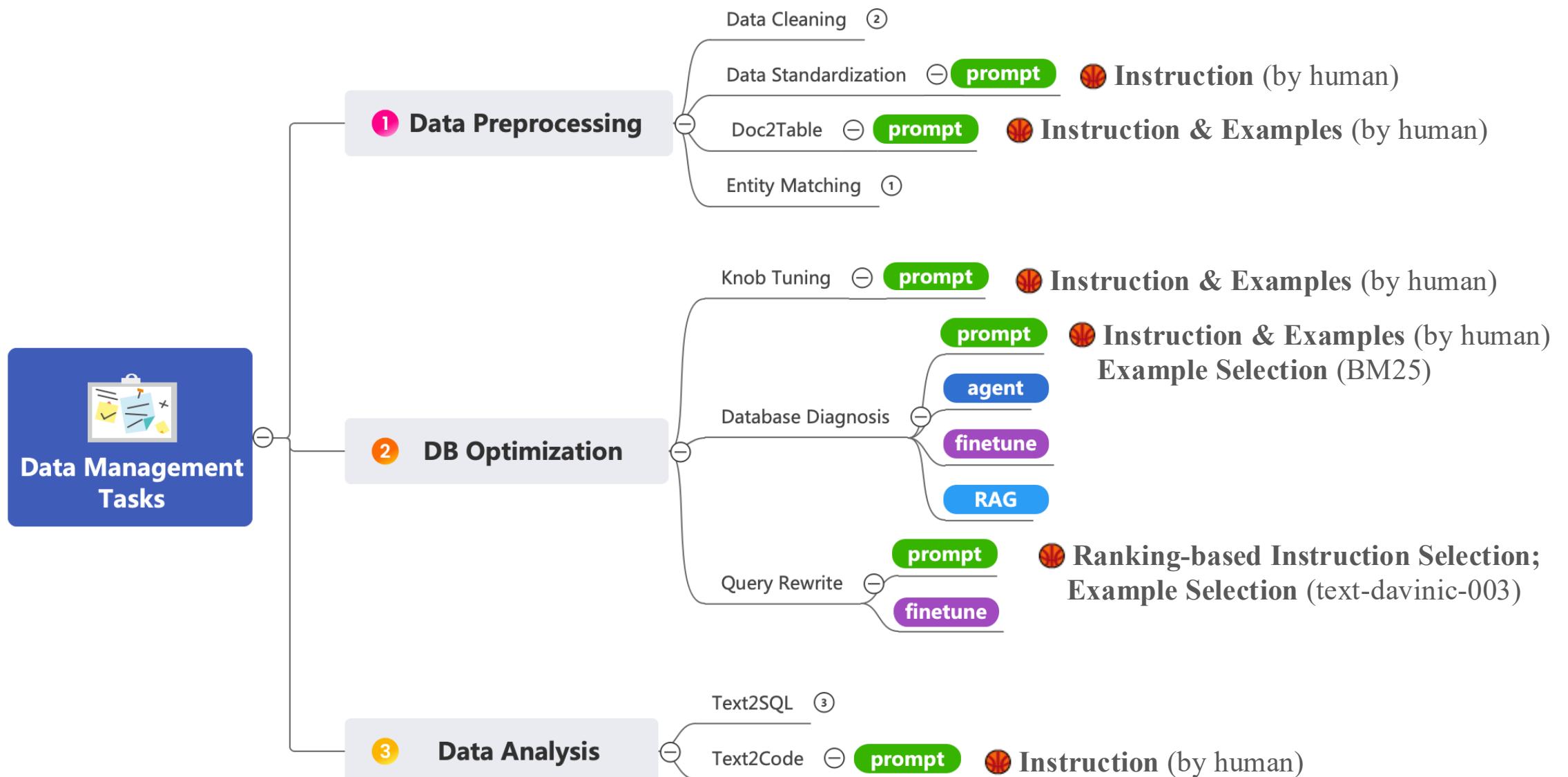
- Automatically generate appropriate (*instructions, demonstration examples*) from large space with limited prompt tokens (or interaction rounds with LLMs).

□ Challenge 2: How to Efficiently Interact with LLM Using Prompts?

- Iteratively select prompt for input request (e.g., retrieving suitable examples from candidate ones) is time consuming
- Long prompts often include more useful information, but require long inference time for LLM and hard to understand



Prompt Engineering Techniques for Data Management



Prompt Engineering for Data Standardization

□ Problems in Traditional Data Standardization

- Require substantial programming skills to write UDFs
- Require complex running environment to run the UDFs
- Much prior experience is hard to convert into UDFs

Decompose into Operations

- data structural consistency
- data type conversion
- data standardization
- data anomaly detection

□ Manually write NL prompts to avoid writing complex functions

```
# define an UDF
def user_define_function(inputs):
    # import the dependencies
    # implement the logic based on the use cases
    # process the inputs
    # return the processed data
    return processed_inputs

# SQL query
SELECT user_define_function(inputs)
```

```
# define a user-defined prompt template
prompt_template = "
<define the system prompt parameter.>
<define the prompt incorporating with the {inputs} and the
output.>"

def user_defined_function(inputs):
    # an LLM_call function communicates with LLM engine
    # and returns the processed result
    processed_inputs = llm_call(prompt_template.format(inputs))
    return processed_inputs
```

User-Defined Function



User-Defined LLM Prompt

Prompt Engineering for Data Preprocessing

□ Problems in Traditional Data Standardization

- Require substantial programming skills to write UDFs
- Require complex running environment to run the UDFs
- Prior experience cannot be utilized in UDFs

Decompose into Operations

- data structural consistency
- data type conversion
- data standardization
- data anomaly detection

□ Manually write NL prompts to avoid writing complex functions

- E.g., for date data structuralization,
- **Traditional UDFs:** Require enumeration of the date format / utilize different date processing packages

→ **Manually define the output format (YYYYMMDD) in the prompt and let LLMs handle the data processing**

1	item_id	user_id	user_rating	date
2	"101"	"201"	3	"20220305"
3	"102"	"201"	4	"2022/10/23"
4	"101"	"202"	5	"7th April 2021"
5	"101"	"203"	2	"Feb 03 2020"

Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** Insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** Different programming languages and SQL syntax

```
SELECT ...  
WHERE ...  
AND (  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=1  
    )  
    OR  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=2  
    )  
)
```

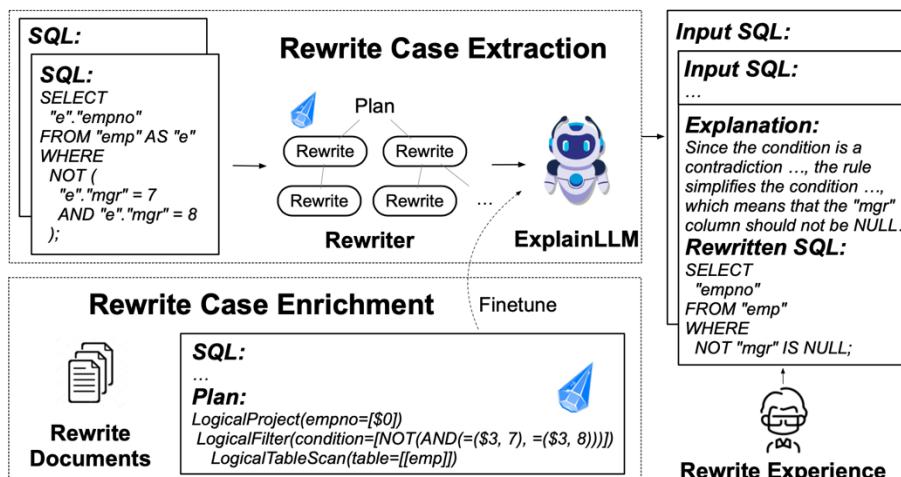
Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** Insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** Different programming languages and SQL syntax

□ 1. Prompt-based Rewrite Case Generation

- Utilize LLM to generate well-explained rewrite cases, i.e., *(original query, rewritten query, rewrite rule, rewrite explanation)*



```
SELECT ...  
WHERE ...  
AND (  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=1  
    )  
    OR  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=2  
    )  
)
```

$p_{plan} = \dots$ The rewriter first translates input SQL into equivalent input plan. Second, it uses the given rewrite rule to transform the input plan into rewritten plan. Third, it translates the rewritten plan into equivalent rewritten SQL. ... You should not mention the input plan and rewritten plan in your explanation, as if the rewriter directly transforms the input SQL into the rewritten SQL. ...

Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

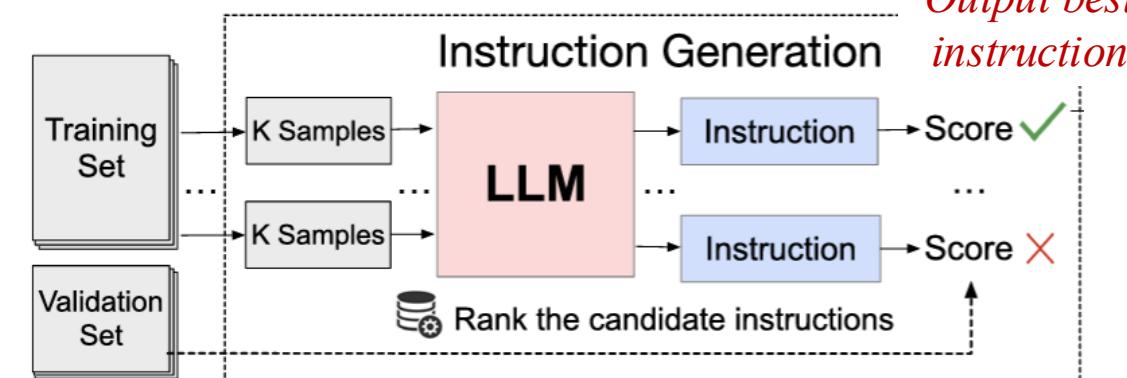
□ 2. Prompt-based Query Rewrite

- The quality of prompt impacts the performance of LLM on different rewrites → **Automatic Prompt Generation**

2.1 Instruction Generation

- Write prompt to instruct LLM to generate instruction candidates by examples (e.g., five pairs):
- I followed the instruction to rewrite the input SQL query to produce an equivalent SQL query...
- Based on the instruction, they produced the following input-output pairs: \n\n[example pairs] \n\nInstruction:"

```
SELECT ...  
WHERE ...  
AND (  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=1  
    )  
    OR  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=2  
    )
```



Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

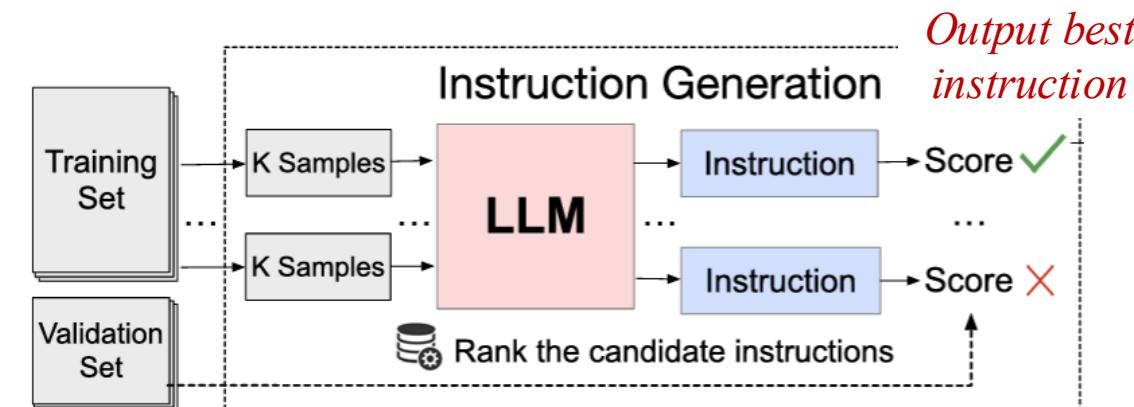
□ 2. Prompt-based Query Rewrite

- The quality of prompt impacts the performance of LLM on different rewrites → **Automatic Prompt Generation**

2.1 Instruction Generation

- Rank the generated instruction candidates based on their benefit for validation set (e.g., the cost reduction after rewriting with the instructed llm)

```
SELECT ...  
WHERE ...  
AND (  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=1  
    )  
    OR  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=2  
    )
```



Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

□ 2. Prompt-based Query Rewrite

- The quality of prompt impacts the performance of LLM on different rewrites → **Automatic Prompt Generation**

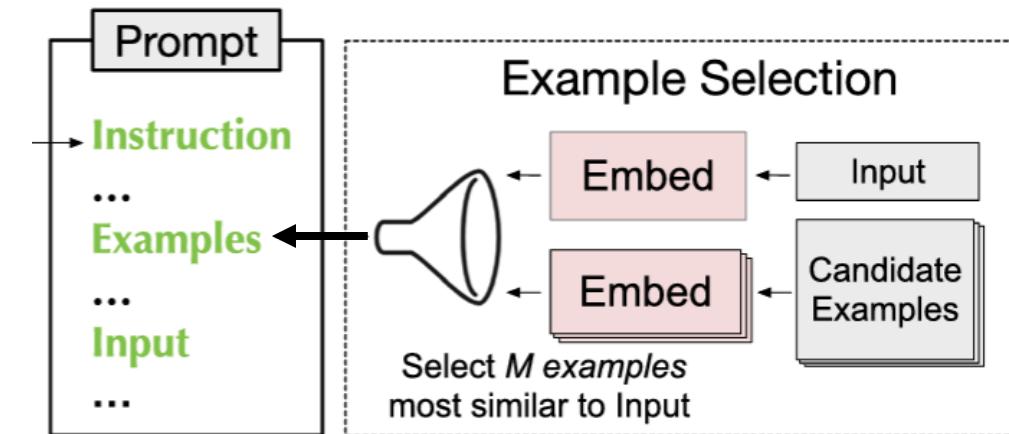
2.1 Instruction Generation

- Rank the generated instruction candidates based on their benefit for validation set (e.g., the cost reduction after rewriting with the instructed llm)

2.2 Demonstration Example Generation

- Match the current query q with a few candidate rewrites whose input queries are similar to q

```
SELECT ...  
WHERE ...  
AND (  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=1  
    )  
    OR  
    EXISTS (  
        SELECT a2 FROM t2  
        WHERE t2.b2=2  
    )
```



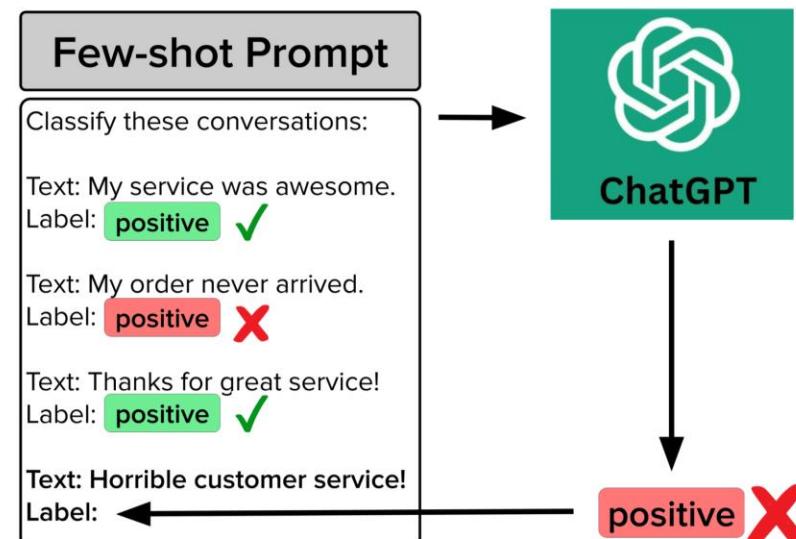
Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

□ 3. Efficiency Issues

- Search-based example matching is time-consuming
→ Prompt / Finetune a model to **identify the most suitable demo examples**



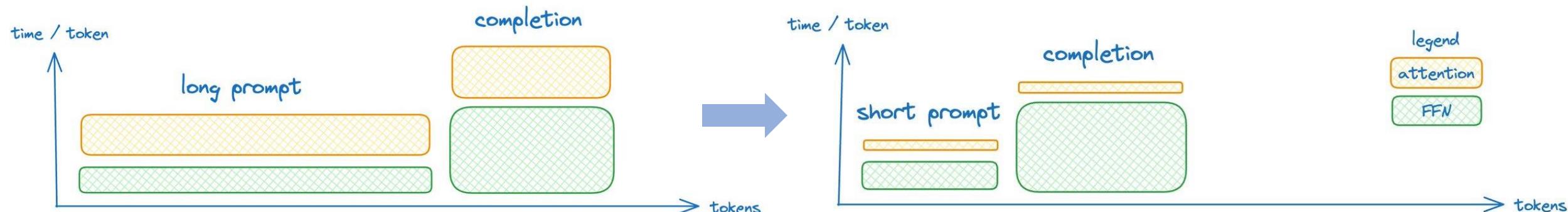
Prompt Engineering for Query Rewrite

□ Problems of Rule-driven Rewriters

- Inadequacy of rules: insufficient for handling complex query transformations (e.g., merging sub-queries)
- Cross-system migration: different programming languages and SQL syntax

□ 3. Efficiency Issues

- Excessively long prompt can slow down the LLM inference
→ Strike a balance between **prompt length and LLM performance** (e.g., generate summary for the detailed description of rewrite rules)



Prompt Engineering for Table Data Extraction

- LLM cannot directly extract structured table from long-context and semi-structured documents
- Core Idea
 - For long document, feed sampled documents to the LLM, and **prompt it to generate useful information** that can form a structured table (e.g., writing code to extract the values of important attributes)

Input

Data lake: A collection of semi-structured documents (e.g. HTML, TXT, XML)



EVAPORATE-CODE+ (Doc2Table)



Output

Tables: A structured view of the data in the input documents.



name	draft year	position
Jayson Tatum	2017	Power Forward
Anthony Davis	2012	Center
Kevin Durant	2007	Small Forward
Steph Curry	2009	Point Guard

Prompt Engineering for Table Data Extraction

- LLM cannot directly extract structured table from long-context and semi-structured documents
- Prompt-based Table Data Extraction
 - Schema Synthesis
 - With a sampling subset of documents, it **prompts LLM to extract attributes** based on their occurrence frequencies
 - Rerank the extracted attributes by adjusting their frequency weights with LLM
 - Function Synthesis
 - A heavy job to extract attribute values from every document → **Prompt LLM to write code** to extract the attribute values

*Function
Prompt*

Here is a file sample:
<title>U.S. GDP Rose 2.9% in the Fourth Quarter </title>
<meta itemProp="datePublished"
content="2023-01-26T10:30:00Z"/>
...

Question: Write a python function called "get_date_published_field" to extract the "datePublished" field from the text. Include any imports.



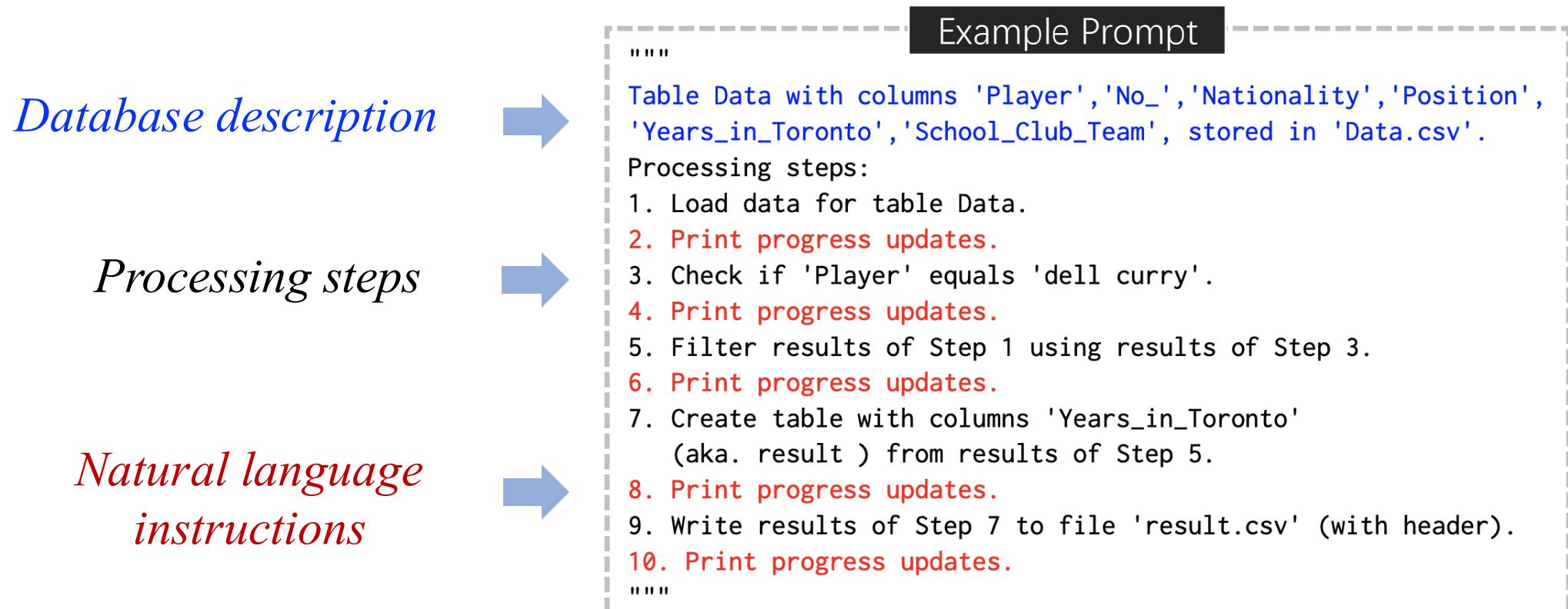
```
from bs4 import BeautifulSoup
def get_date_published_field(text: str):
    soup = BeautifulSoup(
        text, parser="html.parser"
    )
    date_published_field = soup.find(
        'meta', itemprop="datePublished"
    )
    return date_published_field['content']
```

Prompt Engineering for NL2Code

□ Motivation: Textual instructions → Execute by generating code

- Core Idea

- Design prompt to drive LLM to (1) **decompose the input query** into a series of simple processing steps and (2) translate the instructed steps into **executable code**



Prompt Engineering for Knob Tuning (Workload Generation)

□ Motivation

- ML-based methods require numerous workloads as training data

• Main Steps

- Workload generation: Use manual-crafted GPT-4 to generate diversified workloads for specific database schema and workload types

Workload Generation Prompt

You are a helpful DBA, tasked with generating challenging OLAP / OLTP workloads and fulfill my goals.

OLAP: Recall the complex queries in the TPC-H, TPC-DS, and JOB databases... Ensure that the queries do not involve write operations like 'insert', 'update', or 'delete'.

Note 1: The key attributes of OLAP queries are as follows:

OLTP: Recall the simple queries in the Sysbench, TPC-C and OLTPBench databases...

Generate several 'select', 'insert', 'update' or 'delete' queries...

Note 1: The key attributes of OLTP queries are as follows:

Goal: Executable Workloads

Here are the database schema followed by some column values that might assist you in generating predicates.....

```
CREATE TABLE public.events_relevant (
    event_id integer NOT NULL: "336", "444"...
    device_id text: "-918417362937292"...
); .....
```

Goal: Diverse Workloads

You should craft <number> highly intricate <type> queries, incorporating elements such as multi-table JOINs — with a minimum of <x> and up to <y> tables.

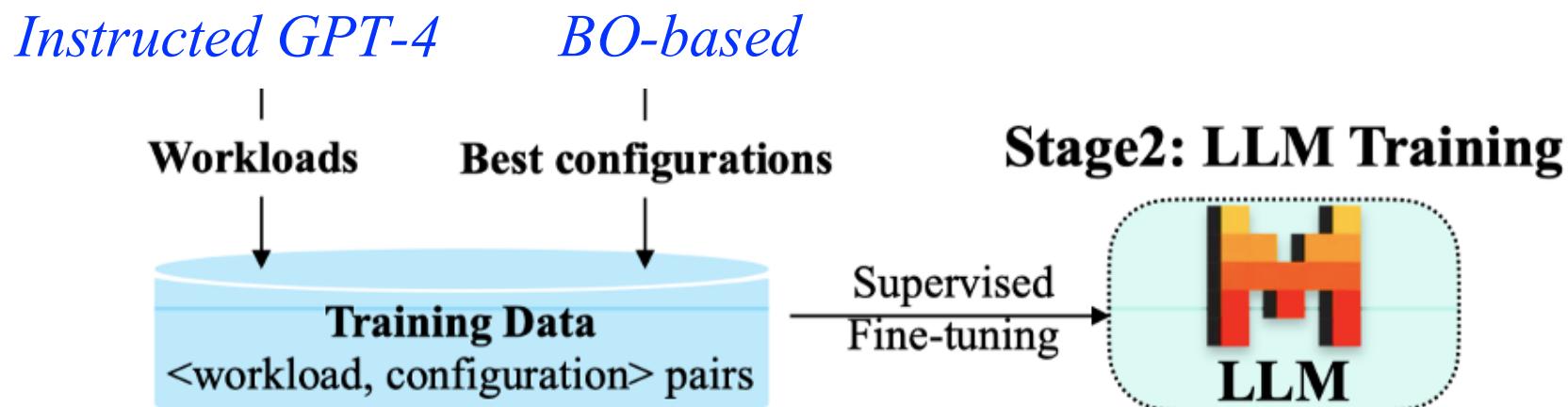
Prompt Engineering for Knob Tuning (Workload Generation)

□ Motivation

- ML-based methods require numerous workloads as training data

• Main Steps

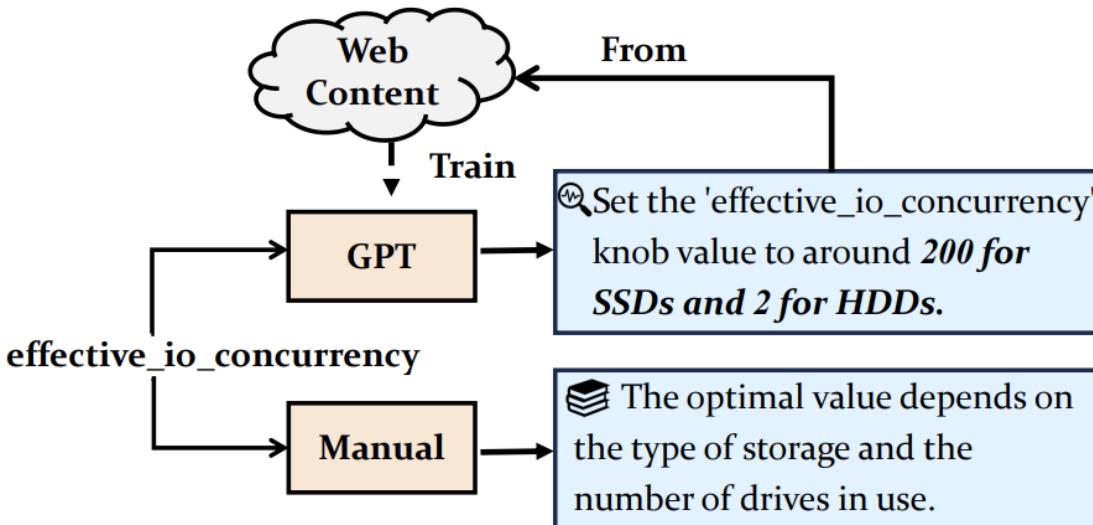
- LLM Finetuning: Train Mistral-7B with workloads labeled with configurations recommended by BO-based algorithm
 - **LLM Input:** Workload features, internal metrics, query plan;
 - **LLM Output:** Generate the configuration change based on previous configuration



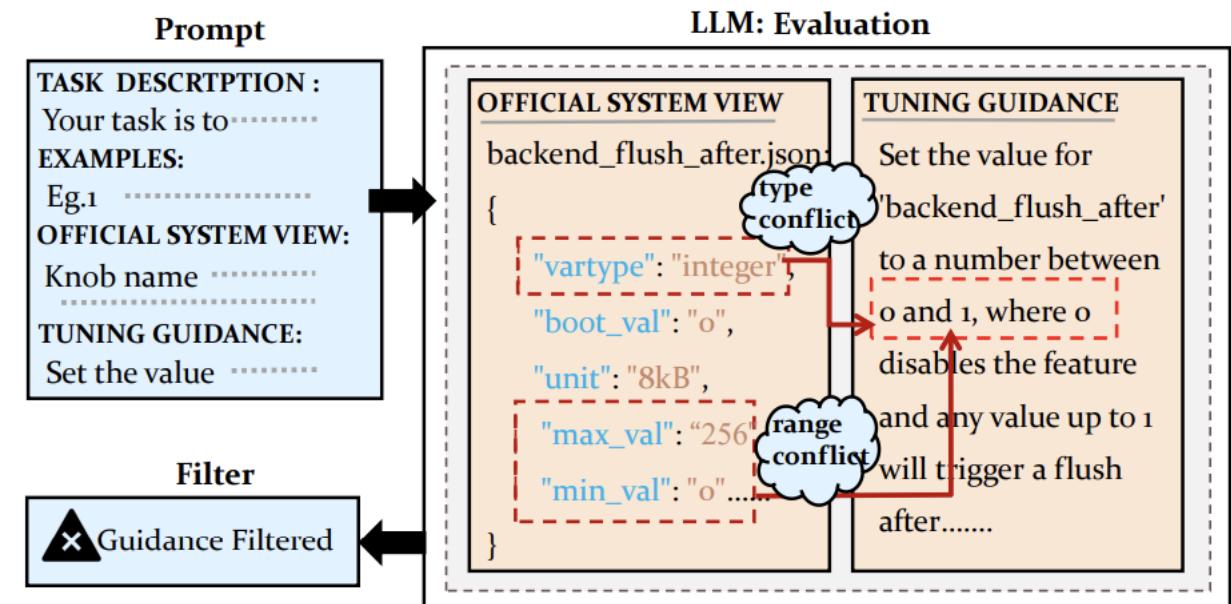
Prompt Engineering for Knob Tuning (Knowledge Extraction)

- Textual experience are not well utilized in knob tuning
→ Split knowledge extraction task into four main steps, and manually design prompts to guide LLM in each step

- 1. Extract knowledge from LLM:
retrieval knowledge by (1) directly asking GPT or (2) prompting LLM to summarize from documents



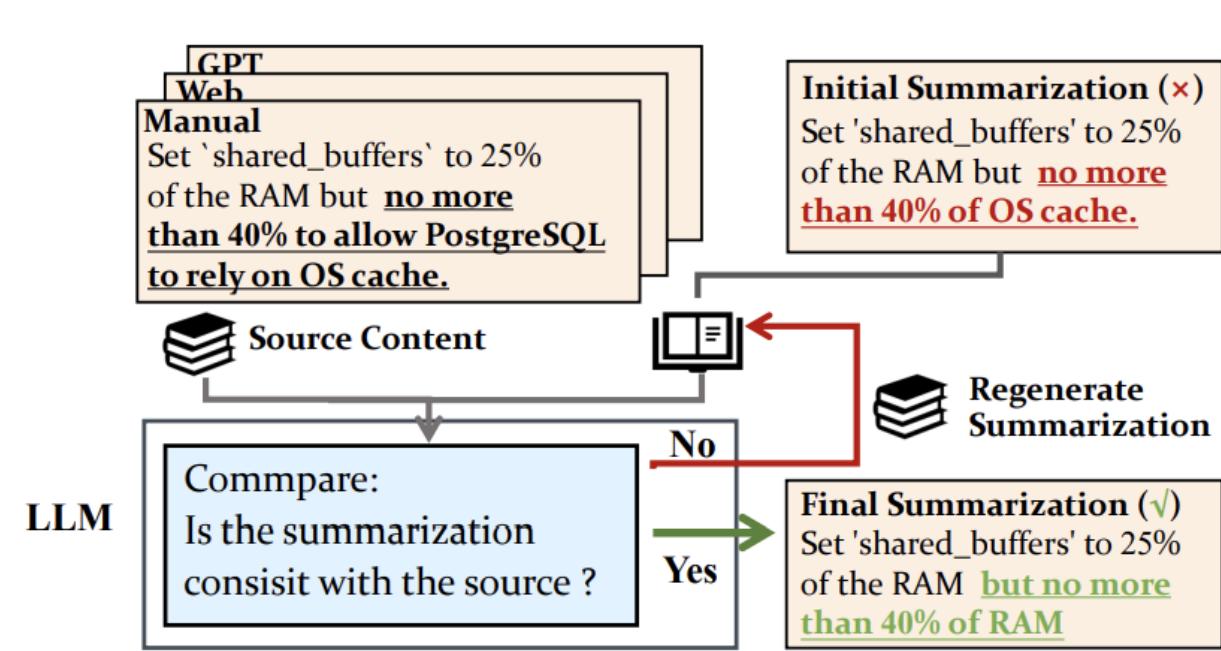
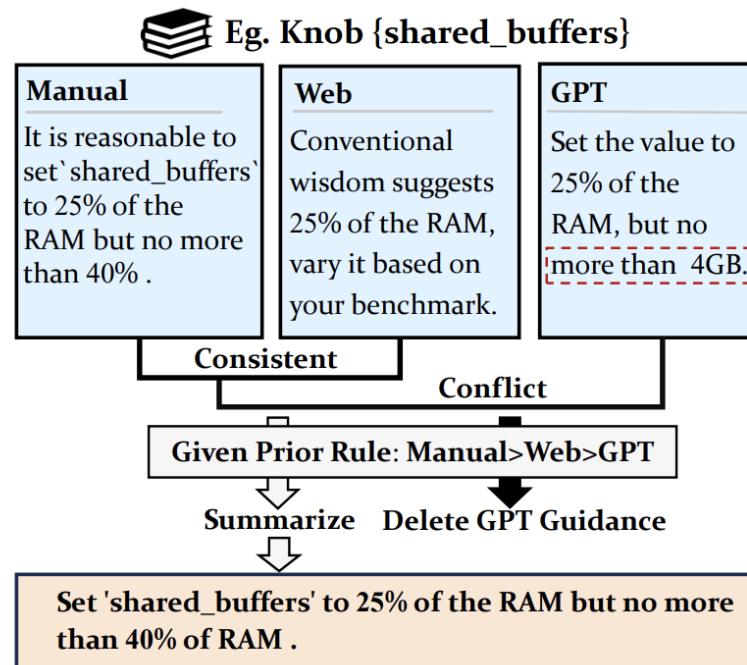
- 2. Filter noisy knowledge: Prompt LLM to evaluate whether the tuning knowledge conflicts with the system view



Prompt Engineering for Knob Tuning (Knowledge Extraction)

- Textual experience are not well utilized in knob tuning
→ Split knowledge extraction task into four main steps, and manually design prompts to guide LLM in each step

- 3. Summarize knowledge from various resources: Manual > Web > GPT
- 4. Check *factual inconsistency* with the document sources



LLM Prompt for Database Diagnosis

□ Improve diagnosis prompt with matched *Knowledge and Tools*

- 1. Anomaly Metric Description for triggered alerts

During the inspection, it was identified that from 13:20:49 to 13:37:49 on October 15, 2023, the load on the node 'ins:stdload1' was exceedingly high, reaching 160% of the standard capacity. This exceeded the threshold of 100%, thus triggering a warning alert. Although

- 2. Tool Matching with finetuned embed model

$$\text{Score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgDL}})}$$

emb(.): the embedding function of the fine-tuned Sentence-BERT model; s: context; t_j: tool API

- 3. Knowledge Matching by metric attributes

$$\text{sim}(s, t_j) = \frac{\text{emb}(s) \cdot \text{emb}(t_j)}{\|\text{emb}(s)\|_2 \|\text{emb}(t_j)\|_2},$$

- *D: A candidate knowledge chunk; q_i: abnormal metric*

Prompt Template

- *Expert Description: <Role> ; <Task> ; <Steps>*
- *The anomaly alerts are {Anomaly Info}*
- *In this step, you can use the following tools: {Tool APIs}*
- *The matched knowledge is: {Knowledge}*

(Demonstration Examples of available tool usage)

Example Knowledge Chunk D

```
"name": "large_data_insert",
"content": "Identify excessive inserted tuples in a table or
query operations.",
"metrics": ["inserts", "query", "index_schema"],
"steps": "For each inserted table, if the count of inserted
tuple is equal to or exceeds the {threshold}, it's
flagged as a potential root cause."
```

Summarization of Prompt-based Data Management

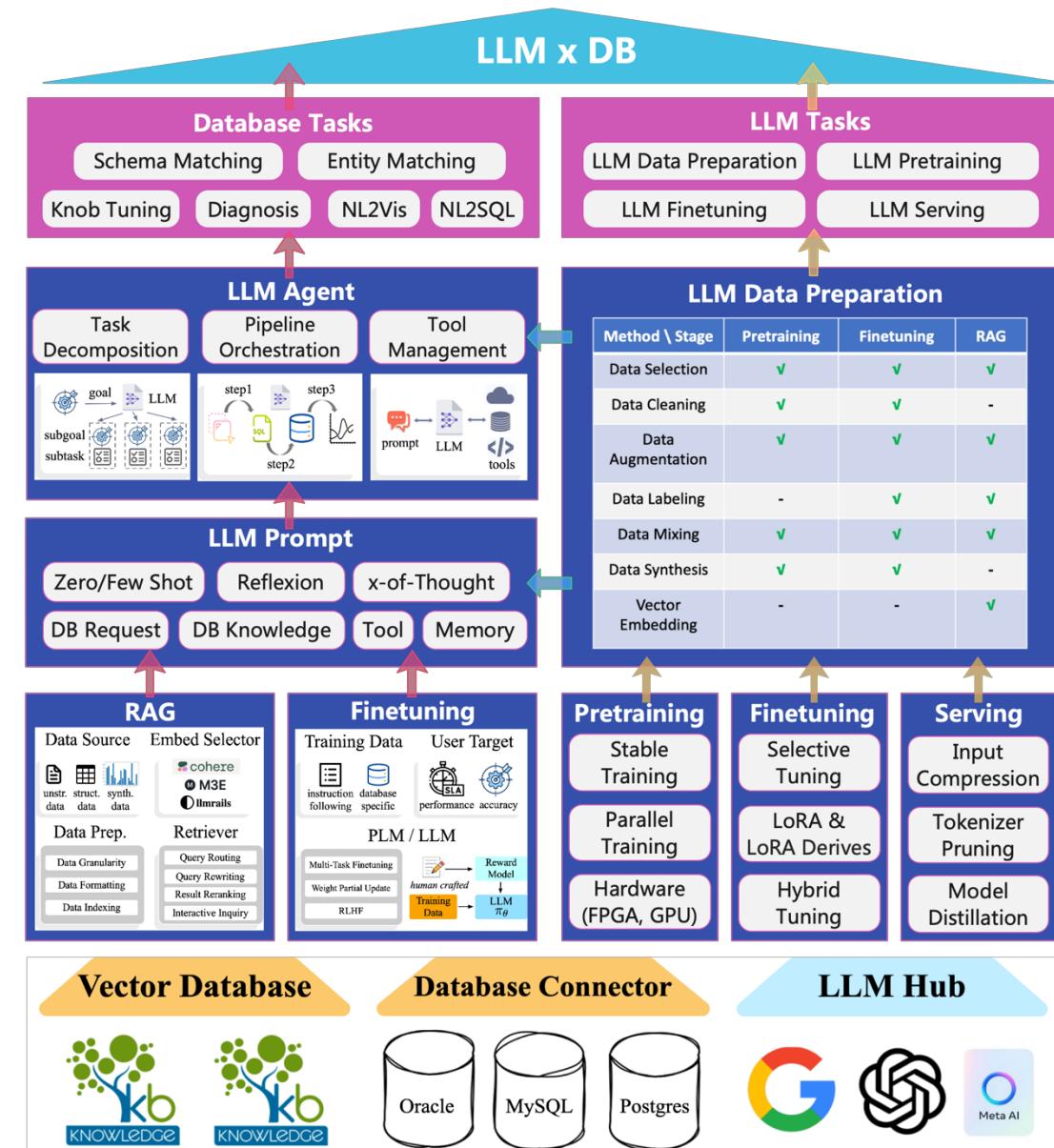
	LLM Job	Operations	Prompt (Content)	Prompt (Examples)	Efficiency Issue
Data Standardization	GPT: Modify Data format	Four Preprocessing Operations	Instruction	Manual	N/A
Query Rewrite	GPT: Query Rewrite	Typical Logical Transformations	Instruction + Example	Ranking based Selection	1. A small-sized model for Example Selection; 2. Prompt Compression; 3. Past rewrite Reuse
Table Data Extraction	GPT: Doc2table	Schema Extraction; Value Extraction	Instruction + Example	Manual	N/A
NL2Code	GPT: Text2code	Processing steps and comments	Instruction	Manual	N/A
Knob Tuning	GPT: Workload Generation; Mistral: Knob Tuning	Workload Generation + Knob Tuning	Instruction	Manual	N/A
Knob Tuning	GPT: Knowledge Extraction	Four steps for knowledge extraction	Instruction + Example	Manual	N/A
Database Diagnosis	GPT / Llama: Root Cause Analysis	Three analysis and tool calling steps	Instruction + Example + External Info	Matched by external info	N/A

Take-aways

- Existing prompts mostly depend on human experts to craft high-quality instructions and examples, which have the scalability problem
- Prompt examples can be generated from humans, real cases, LLMs (e.g., add explanations), and traditional algorithms (e.g., search-based for configs)
- Prompt example selection are critical to the success of the current task
- Prompt instruction format can affect LLM performance (especially weak LLMs)
- The order of instructions / different examples can affect LLM performance
- No one-size-fit-all-model prompt, which needs modifications for different LLMs
- Most existing works are only for offline tasks, which do not consider efficiency issues (e.g., low inference efficiency, unstable responses).
- Efficiency enhancement methods include prompt compression, model-based example selection, and batch inference

Outline of LLMxDB

- Data Management tasks
- LLM Prompt for Data Management
 - Instruction Prompting
 - Few-Shot Prompting
- LLM Agent for Data Management
 - Agent Models + Memory
 - Reasoning / Planning Strategies
 - Tool Management & Learning
- RAG for Data Management
 - Semantic Segmentation
 - Result Retrieval
 - Result Reranking
- Finetuning for Data Management
 - SFT Dataset Generation
- Data Management for LLM
 - LLM Data Preparation



LLM Agent

□ Human-Crafted Prompt for each task → Design Automatic Agents to automatically execute complex tasks with predefined prompt templates

- **LLM agents:** An LLM system that can automatically resolve a series of domain-specific tasks with minor human invocation



Advantages of LLM Agent

❑ Autonomy

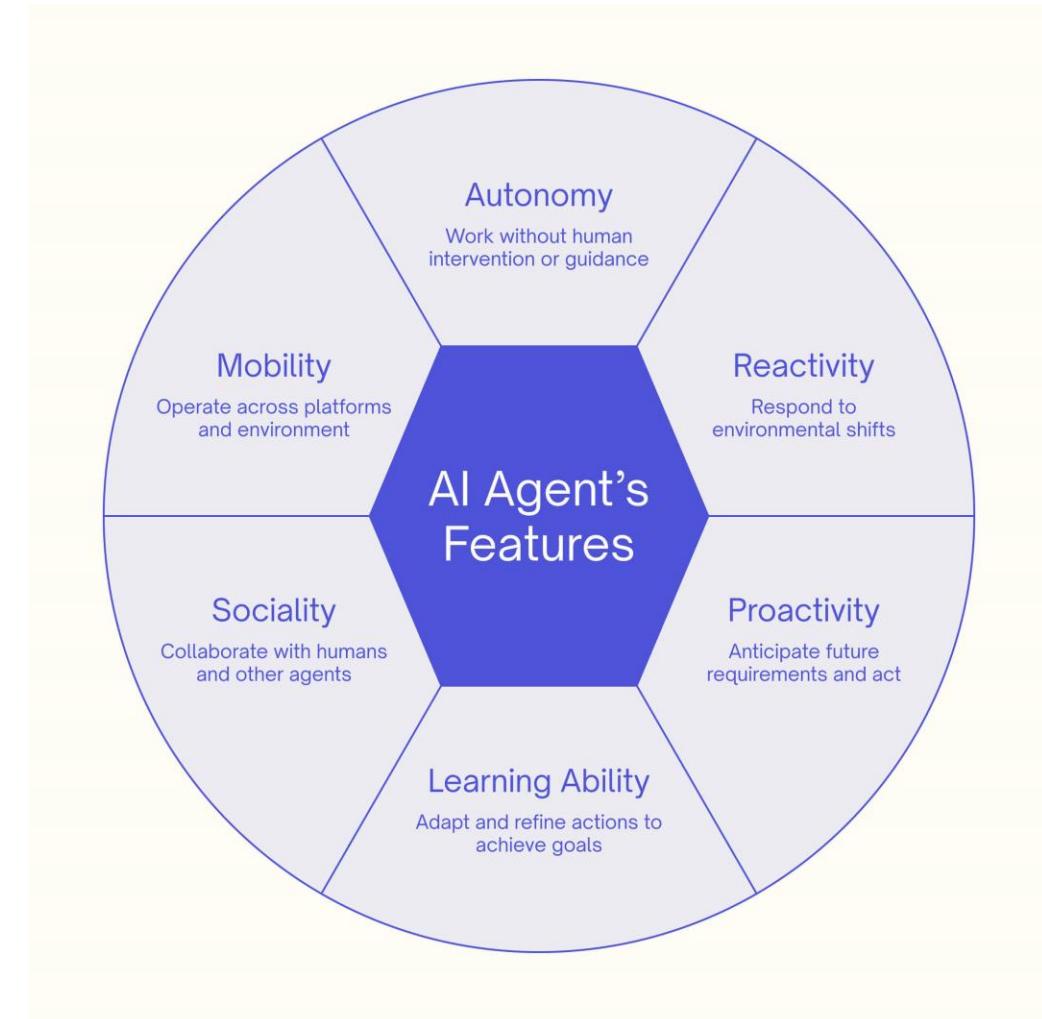
- Execute human instructions and complete tasks;
- Start and execute operations without human requirements

❑ Reactivity

- Respond to immediate changes and interact with its environment

❑ Pro-activeness

- Make plans, and take proactive measures in their actions to achieve specific goals



Advantages of LLM Agent

❑ Social (Multi-Agent) ability

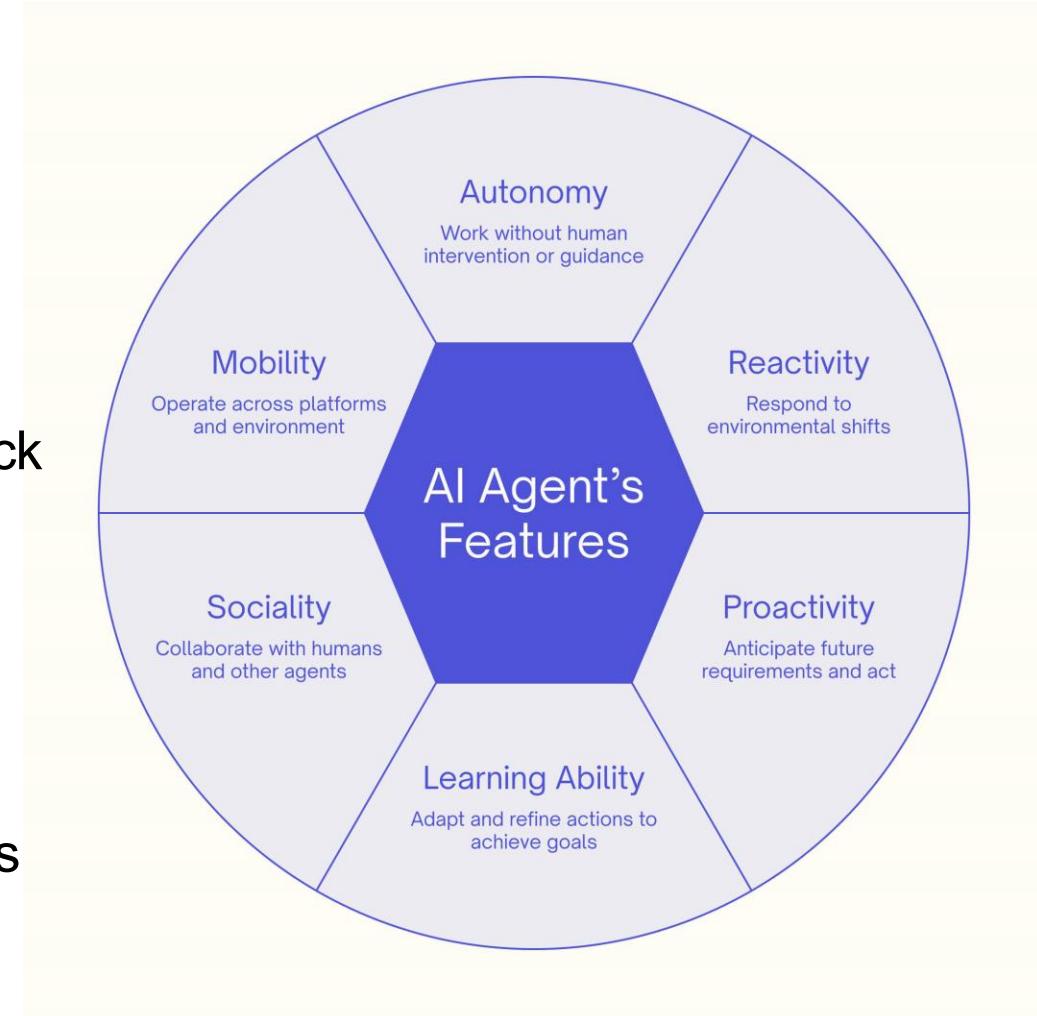
- Interact with other agents (including humans), generate and understand natural language

❑ Learning Ability

- Adaptively integrate new tools and refine the execution pipelines based on environment feedback
- Memorize experience in both external knowledge base and model parameters

❑ Mobility

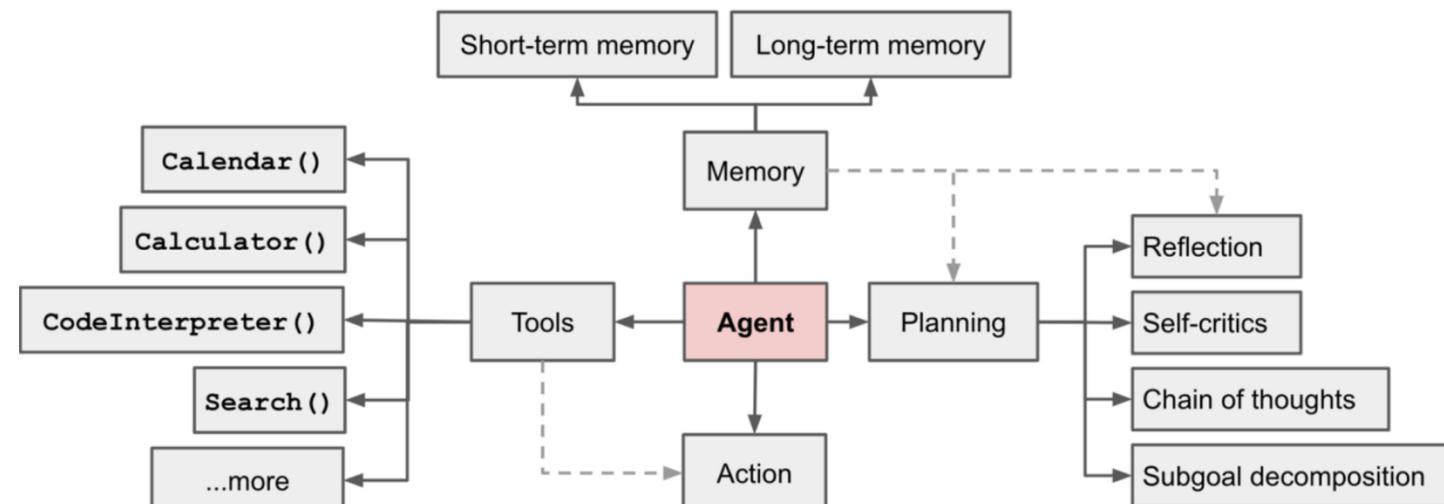
- Easy to generalize to new platforms (e.g., Postgres → MySQL) and scenarios (e.g., specifying new workloads) via prompt-level modification



Main Components of LLM Agent

□ Main components

- **Agent Model:** A powerful language model that can (1) conduct basic reasoning by following the instruction; (2) generate tool calling commands and understand tool outputs ...
- **Planning:** Decompose complex task and conduct step-by-step reasoning
- **Memory:** Store records of interactions from the agent's (long/short term) previous tasks
- **Tools:** Manage agent's calling of external API tools

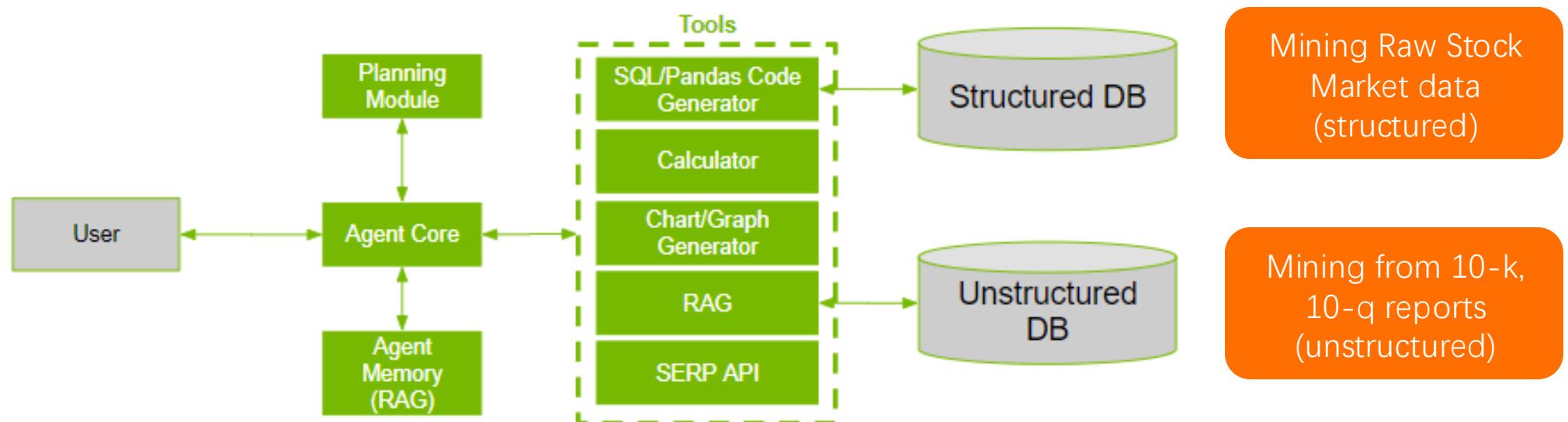


Example LLM Agent

□ Main components

- **Agent Model:** A powerful language model that can (1) conduct basic reasoning by following the instruction; (2) generate tool calling commands and understand tool outputs ...
- **Planning:** Decompose complex task and conduct step-by-step reasoning
- **Memory:** Store records of interactions from the agent's (long/short term) previous tasks
- **Tools:** Manage agent's calling of external API tools

*Example
Data Agent
By Nvidia*



Challenges & Techniques of Agent Components

□ Main components

- *Agent Model: How to (1) conduct basic reasoning by following the instruction; (2) generate tool calling commands and understand tool outputs ...*
 - The **selection / finetuning** of suitable agent model is vital to the performance
 - GPT-3.5 Turbo is effective as code interpreter, particularly in its ability to understand and translate complex task descriptions into functional code;
 - GPT-4 demonstrates weaker consistency compared with GPT-3.5

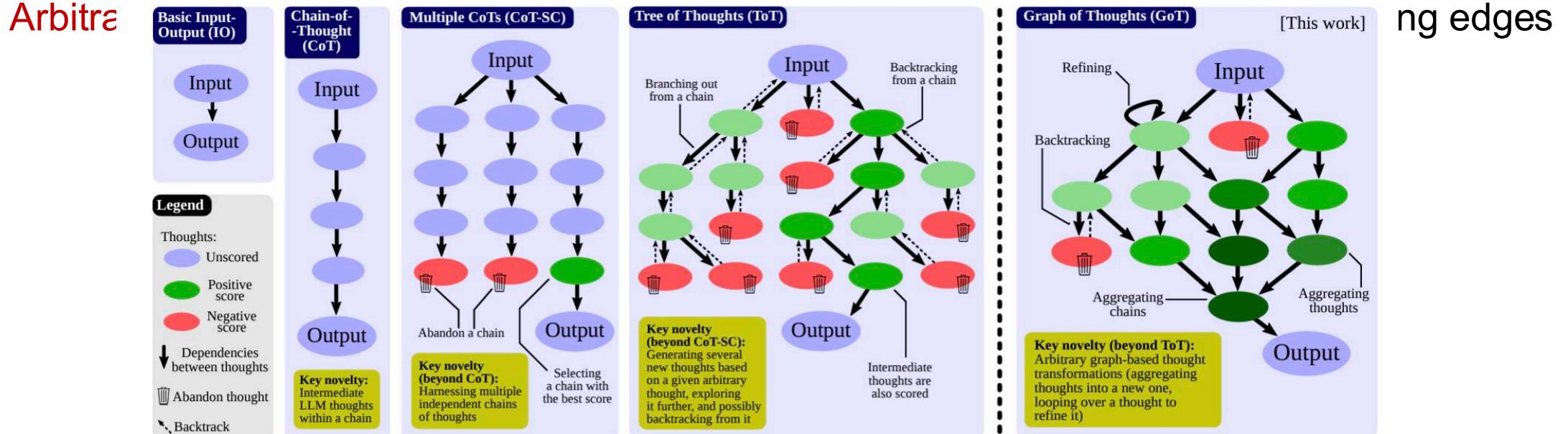
Table 1: Workflow of the proposed model: Application across diverse datasets and output formats

LLM Model	Product	Parameter	I/P Description	Accurate Result	Quality
GPT-4 Turbo	OpenAI	1.96 trillion	10	6	★★★★☆☆
GPT-4	OpenAI	1.76 trillion	10	5	★★★★☆☆
GPT-3.5 Turbo	OpenAI	154 billion	10	7	★★★★★☆
GPT-3.5	OpenAI	125 billion	10	5	★★★★☆☆
Google Bard	Google	1.56 trillion	10	4	★★☆☆☆☆
LLama	Meta	70 billion	10	2	★☆☆☆☆☆
Hugging Face	Hugging Face	355M	10	2	★☆☆☆☆☆

Challenges & Techniques of Agent Components

□ Main components

- **Planning:** *How to decompose complex task and conduct step-by-step reasoning*
 - **Chain-of-Thought (CoT):** Explicitly break down the task into **a sequence of intermediate steps**
 - **Tree-of-Thought (ToT):** Decompose the thought process into **multiple branches and sub-nodes**
 - **Graph-of-Thought (GoT):** LLM thoughts as vertices, edges as thought dependencies → Arbitra



Challenges & Techniques of Agent Components

□ Main components

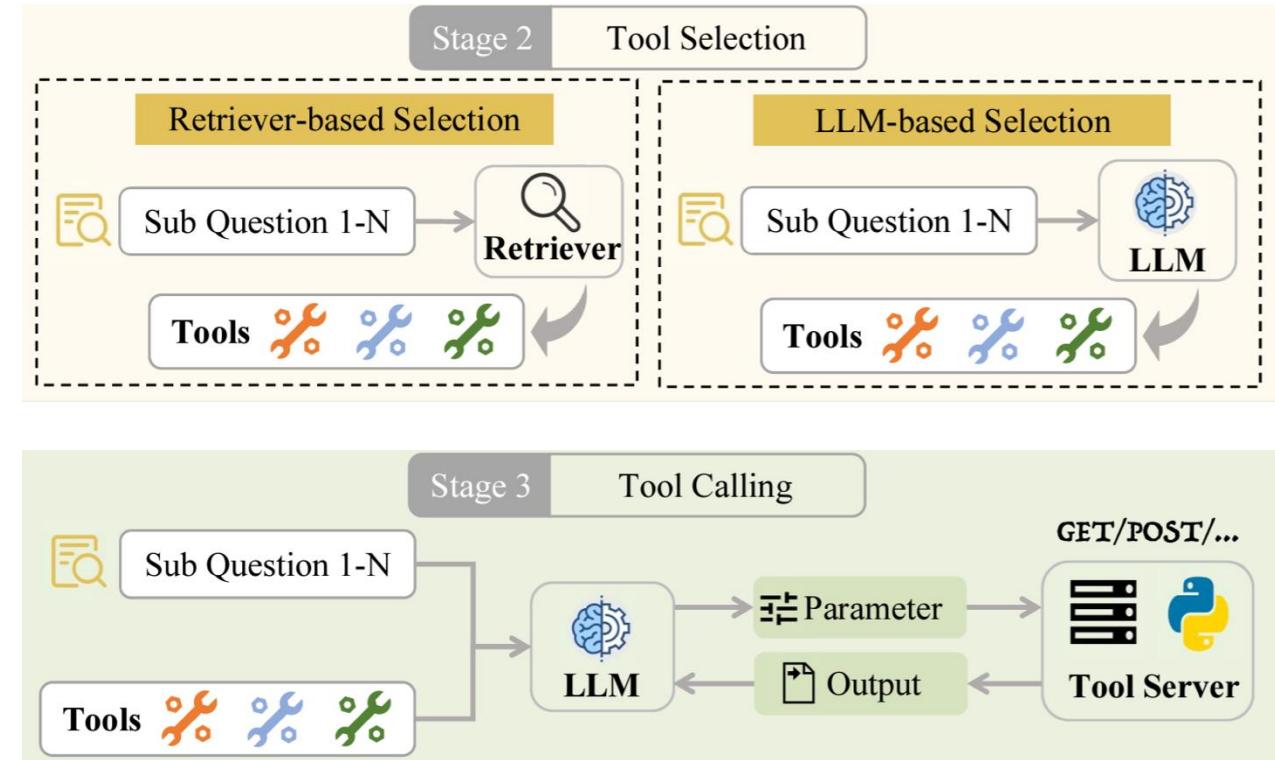
- **Memory:** *How to store historical messages for caching or effective task-solving*
 - **Short-Term Memory:** Memory information is directly written into prompt (e.g., for maintaining the internal state during executing a task)
 - **Challenge:** Due to the limitation of context limit, it's hard to put all memories into prompt
 - **Long-Term + Short-Term Memory:** Long-term memory for stable knowledge (e.g., past behaviors and thoughts that can be retrieved by current situation); Short-term memory for flexible planning (e.g., recent feedback)
 - **Challenge:** How to enhance the memory storage capability and retrieval accuracy
 - **Memory formats:** Different memory formats possess distinct strengths

Memory Format	Advantage	Case
Natural language	Flexible; Rich semantics;	Reflexion: Stores experiential feedback in natural language within a sliding window.
Embeddings	Benefit retrieval	ChatDev: Encode dialogue history into vectors for reuse
Databases	Support Complex Query	DB-GPT: Agents are fine-tuned to understand and execute SQL queries , enabling them to interact with databases

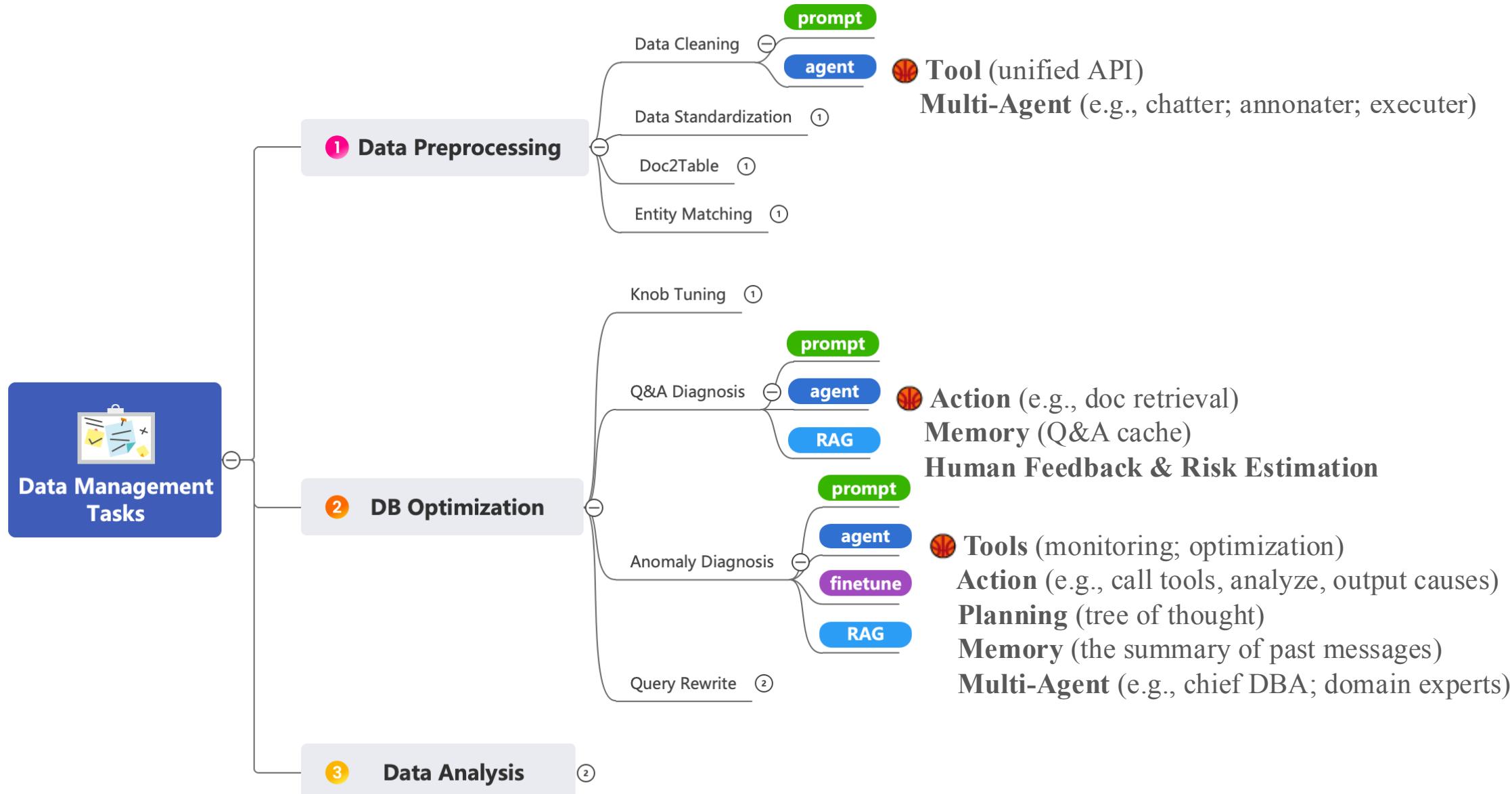
Challenges & Techniques of Agent Components

□ Main components

- *Tools*: *How to prepare tool APIs for better tool calling and result understanding*
 - Vast number of Tools (length and latency issues)
 - → **Tool Selection**
 - Retriever-based tool selection
(match the embed of context and tools)
 - LLM-based tool selection
 - Adherence to API parameter and formats
 - **Tool Calling**
 - LLM Finetuning (with tool calling samples)
 - Output Constraints (e.g., utilize LSM to constrain the LLM output in json format)
 - ...



Prompt Engineering Techniques for Data Management



LLM Agent for Data Cleaning

- Problem: The complexity of tools like Pandas require great human efforts to write code for various column types

Input Table T		
Name	Admission Date	Address
Abby	Fri Jan 1st 10:36:28 2021	1234 west main heights LA 57033
Scott	1996.07.10 AD at 15:08:56	1111 S Figueroa St, LA, 90015

Standardized Table T'		
Name	Admission Date	Address
Abby	01/01/2021 10:36:28	1234 W. Main Hts., LA, 57033
Scott	01/15/2020 15:08:56	1111 S. Figueroa St., LA, 90015

Data Standardization: Unify the format of values within the same column (e.g., the “Admission Date” column)

```
1 def standardize_address(addr):  
2     # Extract street number and street name  
3     street = pd.Series(addr).str.extract(r'(\d+ [^,]+)').  
4         squeeze()  
5     # Extract state name  
6     state = "LA"  
7     # Extract zipcode  
8     zipcode = pd.Series(addr).str.extract(r'(\d{5})').  
9         squeeze()  
10    # Output standardized address  
11    return f"{street}, {state}, {zipcode}"
```

Case-by-Case Analysis and Coding ✖

LLM Agent for Data Cleaning

- Core Idea: Convert textual instructions into declarative API calls and automate data standardization with LLM agent

- Challenges

- (1) How to design declarative and unified tool APIs for data standardization?
- (2) How to optimize the interaction between data scientists and LLM agent?

- CleanAgent

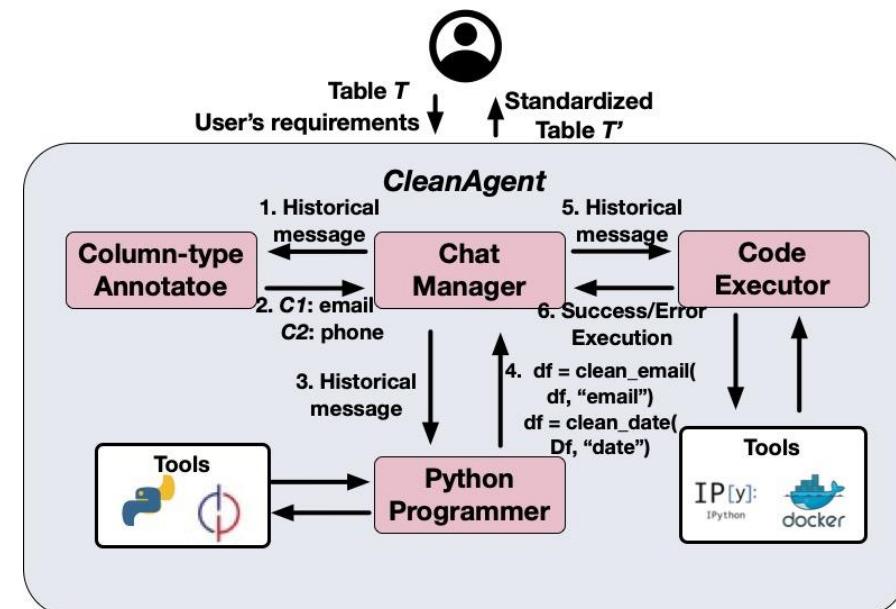
- Unified tool API

```
clean_type(df, column_name, target_format)
```

df: input dataframe (table)

column_name: the column needs to standardize

target_format: the target standardization format users specified

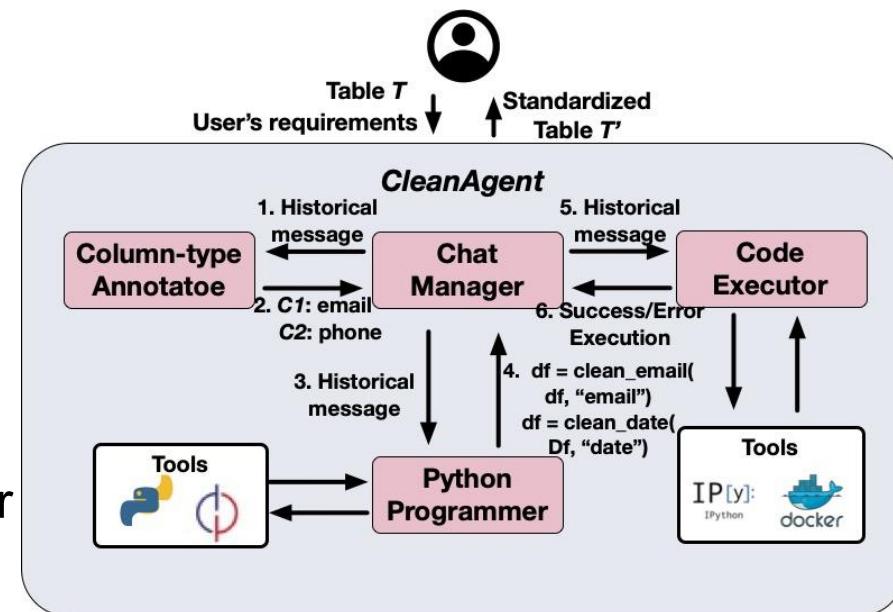


LLM Agent for Data Cleaning

- Core Idea: Convert textual instructions into declarative API calls and automate data standardization with LLM agent
- Challenges
 - (1) How to design declarative and unified tool APIs for data standardization?
 - (2) How to optimize the interaction between data scientists and LLM agent?

□ CleanAgent

- Multi-Agent (ChatGPT as the model)
 - Chat Manager: Store historical message
 - Column Annotator: Annotate the type for each table column
 - Python Programmer: Generate code with candidate APIs
 - Code Executor: Execute code and feed result to Chat Manager



LLM Agent for Q&A Diagnosis

- **Four characters for DBAs to accurately answer debugging questions**
- **Trust:** *Where is the knowledge come from (citations)?*
- **Impact:** *What would happen if I deploy these recommendations?*
- **Risk:** *How catastrophic is this advice (e.g., dropping tables)?*
- **Feedback:** *"this recommendation is useless. Keep it in mind for future"*

Q. My Aurora PostgreSQL database is facing performance issue and I see a large number of sessions waiting for CPU and Lock. What's the issue and how I can fix it?

GPT4: To address the issue, you can try the following steps:

1. Optimize your queries: Review your queries to ensure they are optimized for performance. This may include adding indexes, re-writing queries to eliminate redundant or expensive operations, and ensuring that you're not selecting unnecessary data.
2. Tune PostgreSQL settings: Adjust the PostgreSQL settings to better suit your workload. This may include modifying parameters such as `shared_buffers`, and `work_mem`.
3. Consider upgrading hardware: If your workload is consistently heavy and you are reaching the limits of your current hardware, consider upgrading to a more powerful server with more CPU cores and memory.
4. Implement caching: Consider implementing a caching layer such as Redis or Memcached to reduce the number of queries hitting the database and improve overall performance.
5. Analyze I/O performance: Monitor the I/O performance of your server and database to identify any bottlenecks. Consider using a tool like iostat to get detailed information about your server's I/O performance. You may need to tune the filesystem or RAID configuration to optimize performance.
6. Regularly maintain your database: Ensure that you are regularly performing maintenance tasks such as vacuuming and analyzing your database to optimize performance and prevent bloat.

Panda: Based on my analysis, it looks like your Aurora PostgreSQL database is facing performance issue due to high contention of resources, mainly Lock and CPU.

To further strengthen my analysis I looked at the SQL queries that ran in last 60 minutes, and found 2 dominating queries that had an increase in `shared_blk_written`, `total_time` while waiting for CPU, Lock:`transactionid`, and Lock:`tuple`. Looking at telemetry data, I found that there was increase in total CPU usage and a decline in CPU idle time. I also found memory metrics like `memory:pagetables` and `memory:buffers` increased while `memory:hrmss` decreased. Similarly, `disk:writekbps`, `disk:tps` and `disk:avgreqsz` showed a significant increase as compared to last 24 hours.

Upon investigating the two dominating queries, I found that one is a `SELECT` and the other `UPDATE` both performing a look up on 'aid' column in `pgbench_accounts` table. Hence, based on my analysis I would recommend the following:

Adding an index on 'aid' column in `pgbench_accounts` table can help speed up the retrieval of results for the `SELECT` query. It can also help the `UPDATE` query speed up finding relevant rows to update.

DBE Feedback: ❌

DBE Feedback: 🙌

LLM Agent for Q&A Diagnosis

□ LLM is insufficient to offer the four aspects of characters → Agent

- **Trust:** *Where is the knowledge come from (citations)?*
- **Impact:** *What would happen if I deploy these recommendations?*
- **Risk:** *How catastrophic is this advice (e.g., dropping tables)?*
- **Feedback:** *"this recommendation is useless. Keep it in mind for future"*

Query: For my Aurora MySQL database, I see a lot of sessions waiting for cpu and io/redo_log_flush. What's the issue, and how can I fix it?

Panda: Based on the task information and reference documents, the likely cause of the performance issue is an increase in wait events related to cpu, redo log flush. [This is typically caused by excessive commits and write I/O operations \[s₁\]](#).

To fix this issue, the recommended action is to [group your write operations together to reduce the number of commits \[s₂\]](#). Instead of inserting records row by row or in small batches, you should consider [larger batch sizes \[s₃\]](#) or using bulk insert operations. By reducing the number of commits you can minimize the wait events related to write I/O operations and, mutex lock contention thereby improving the performance of your Aurora MySQL database.

Sources:

[s₁]: Amazon Aurora User Guide for Aurora Pg.901, L.26.

[s₂]: Amazon Aurora User Guide for Aurora Pg.902, L.22.

[s₃]: Amazon Aurora User Guide for Aurora Pg.903, L.44.

GPT-4: If you are experiencing a high number of sessions waiting for 'redo log flush', and 'CPU' in your Aurora MySQL database, it could indicate potential performance issues:

Optimize your database design and queries to reduce the number and duration of transactions. Ensure your transactions are committing or rolling back promptly. Consider breaking down large transactions into smaller ones to reduce contention. Monitor and tune the innodb_buffer_pool_size and innodb_buffer_pool_instances configuration parameters to optimize memory usage.

Monitor your disk I/O performance to identify any bottlenecks. Consider optimizing your queries and transaction management to minimize the amount of data written to the redo log. Ensure that your database server has sufficient disk throughput and IOPS for your workload. Adjust the innodb_log_file_size configuration parameter to an appropriate value based on your workload.

DBE Feedback: GPT-4 is incorrect. Aurora doesn't use innodb_log_file_size. Also, "breaking down large transactions into smaller ones to reduce contention" is the opposite of what should be done here. Panda clearly outperforms GPT-4 in this scenario.

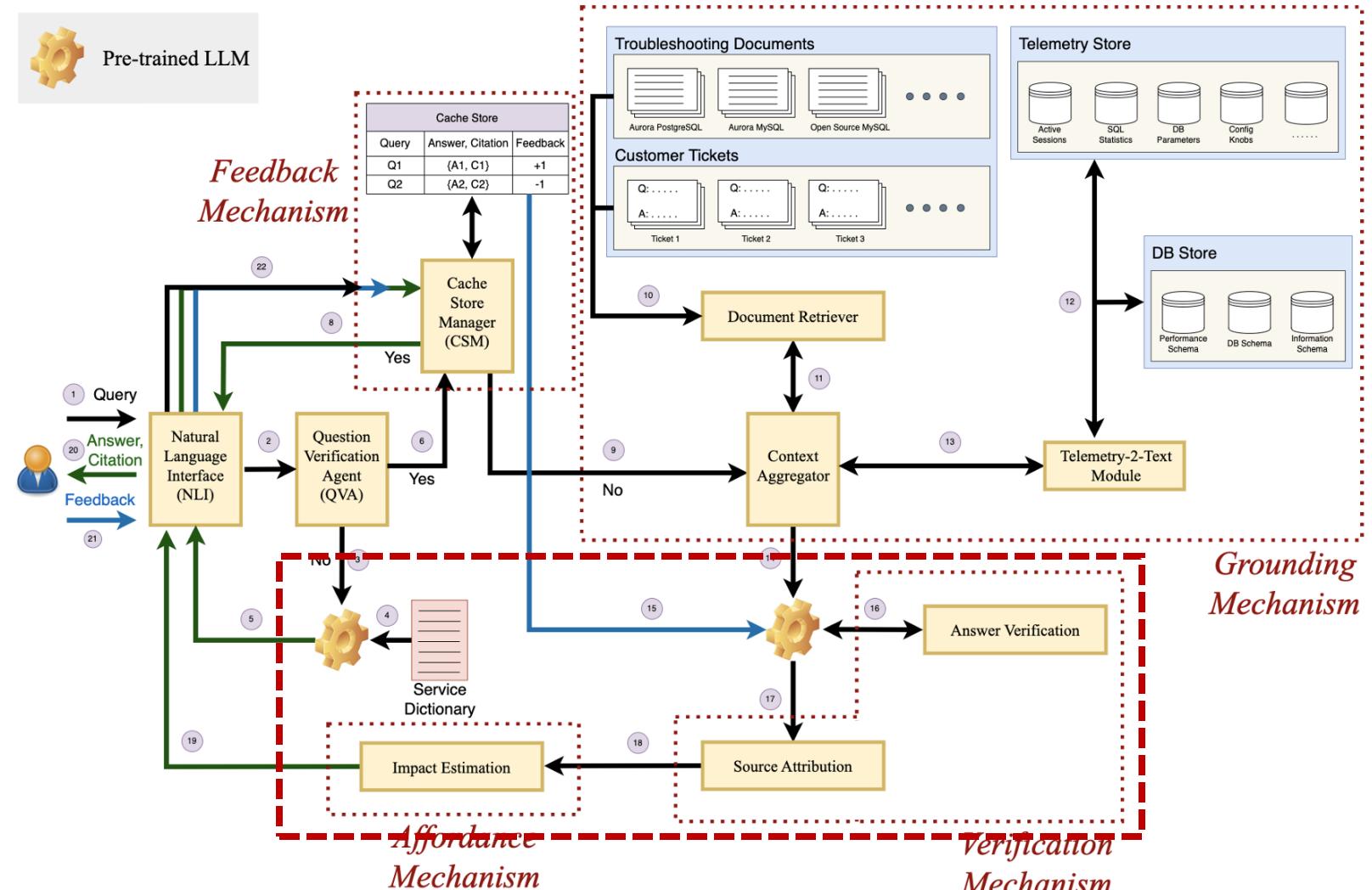
LLM Agent for Q&A Diagnosis

□ Panda: Performance Diagnosis Agent

Address accuracy and security concerns with LLM+database+human

1. **Executable Action.** Prior to evaluation, it is vital for the LLM agent to generate executable actions.

2. **Human Feedback Management.** There is a need to effectively utilize both human and database feedback to address similar issues in databases.

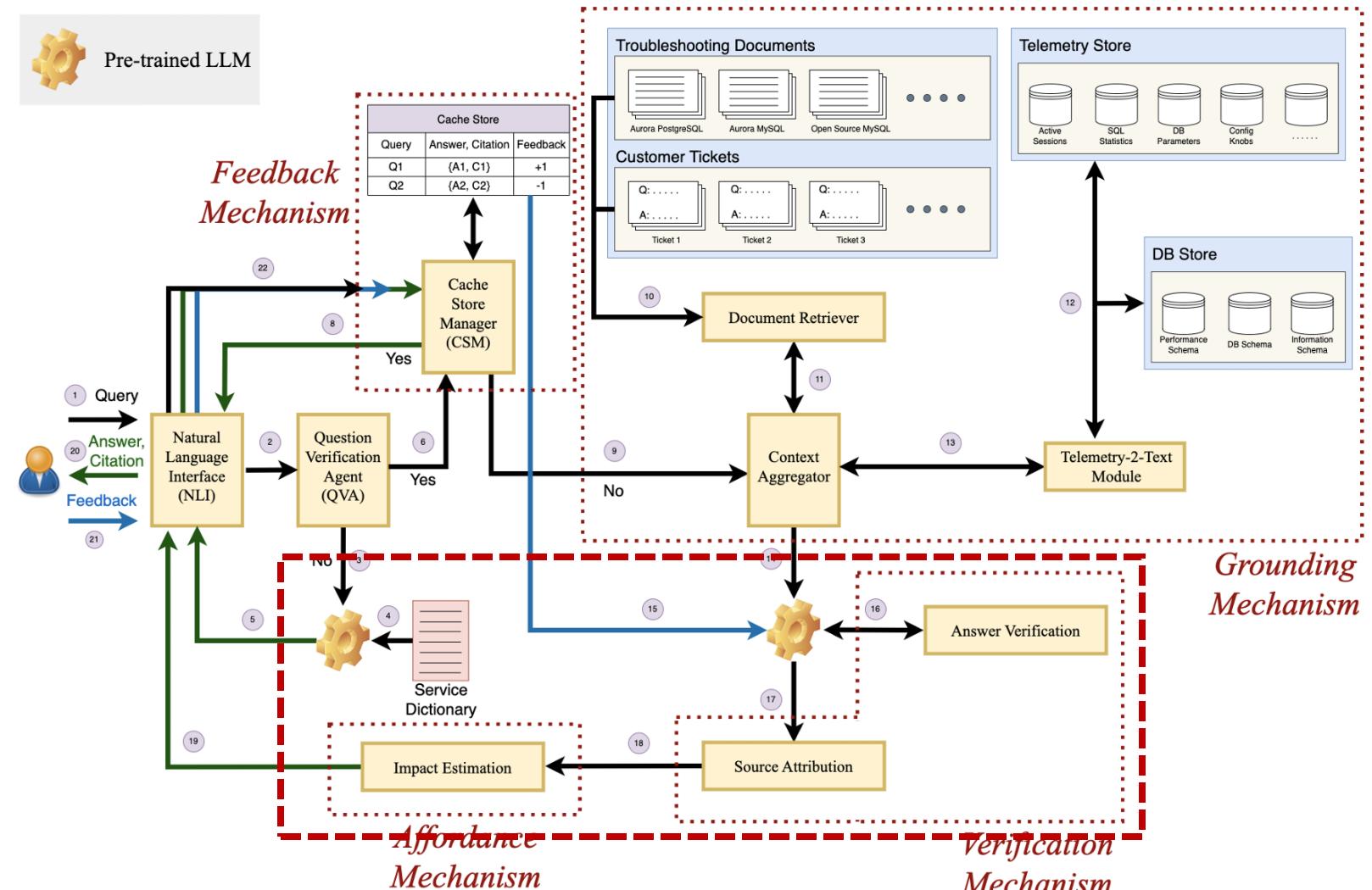


LLM Agent for Q&A Diagnosis

□ Panda: Performance Diagnosis Agent

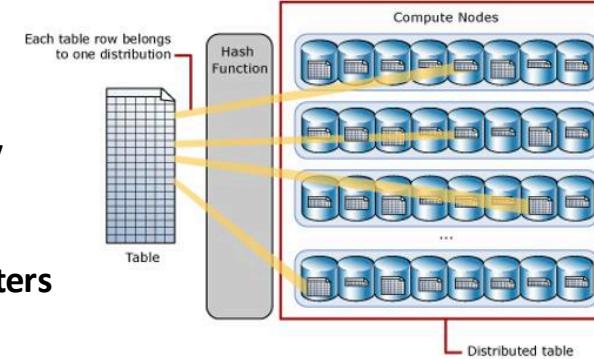
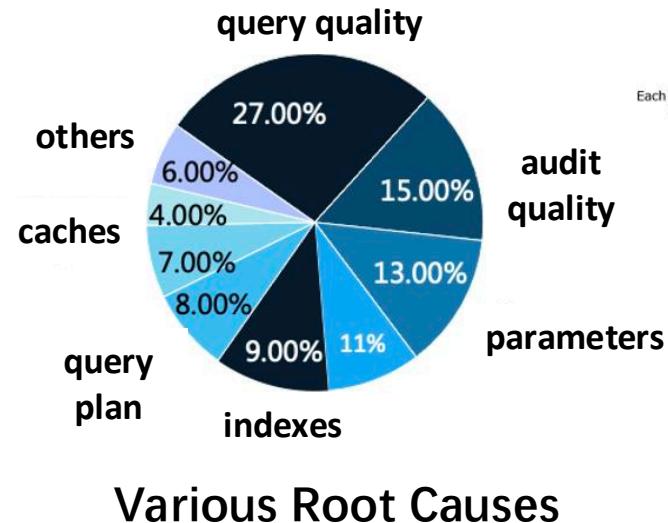
Address accuracy and security concerns with LLM+database+human

3. **Risk Estimation.** Involve a wide range of metrics at both fine and coarse (e.g., query or workload latency) levels.

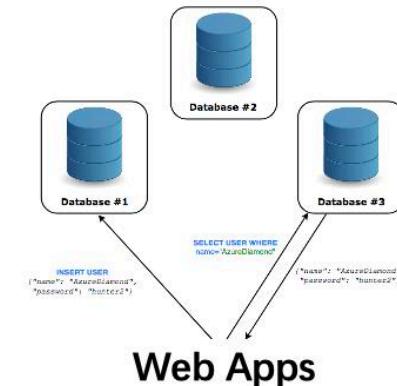


LLM Agent for Anomaly-based diagnosis

- **Numerous Documents:** A single database product provides over **10,000+ pages** of materials (e.g., operational guides and instructions). It's tricky for junior DBAs to fully grasp all of this.
- **Significant Operational Pressure:** The number of cloud database instances is massive, but there's a shortage of DBAs.
- **Complex Issues:** Many urgent problems are interconnected, making it hard to respond quickly, resulting in economic losses.



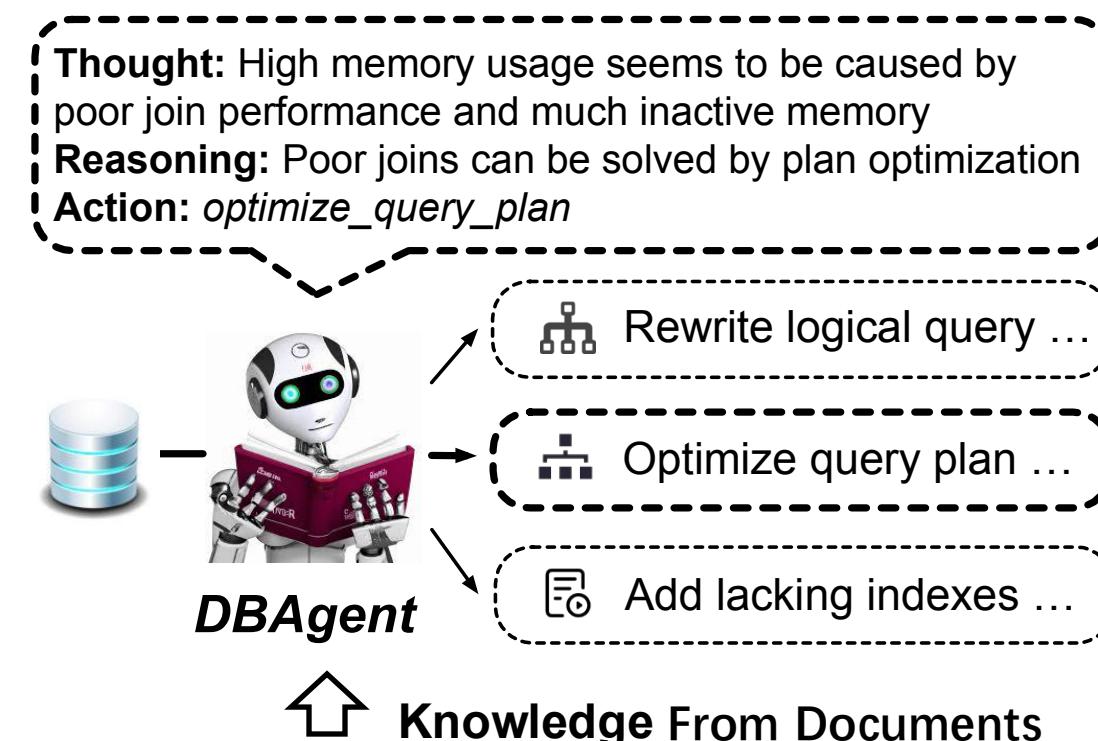
1M+ Cloud Instances



Correlated Online Issues

LLM Agent for Anomaly-based diagnosis

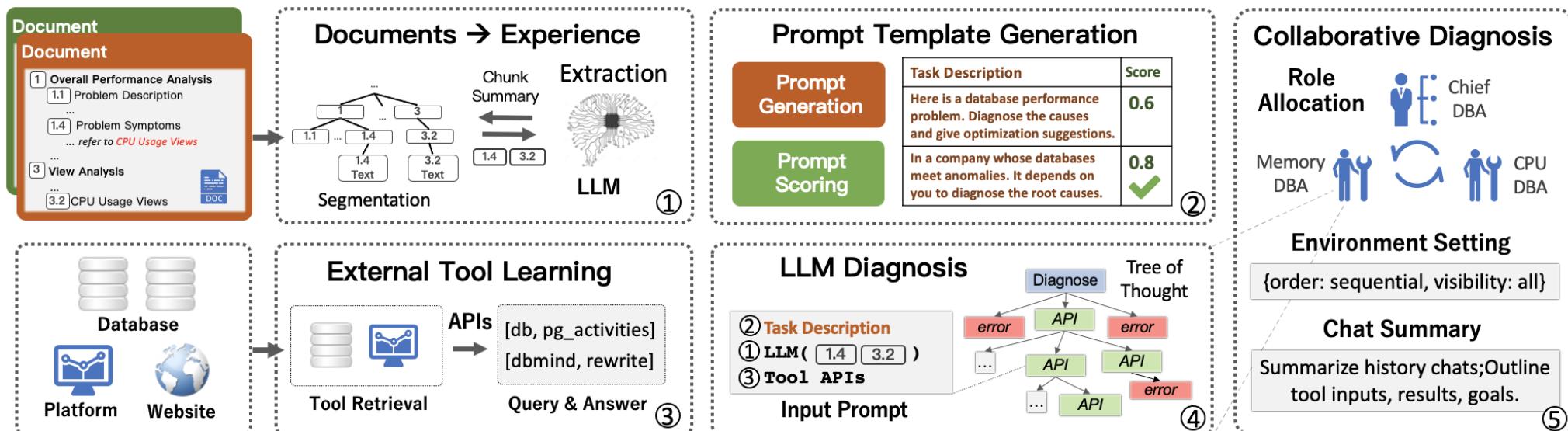
- **Goal:** Utilize LLM as an “intelligent DBA assistant”, learning from human operational experience and diagnose root causes.
- **Benefits:** Save manpower; Enhance proficiency in using maintenance tools; Improve the ability to generalize operational knowledge.



Other methods like
expert systems are
hard to reuse
similar cases

LLM Agent for Anomaly-based diagnosis

- **Tool API:** Monitoring Tools (e.g., logs, views, metrics); Optimization Tools (e.g., index)
- **External knowledge Extraction:** Segmentation of text blocks; generation of summary indexes; extraction of formalized knowledge.
- **Planning:** Improving tool usage accuracy through tree-search-based method.
- **Multi-Agent optimization:** Chief DBA (diagnostic scheduling, finding summarization); Domain Experts (e.g., resource anomaly, query optimization), Chat Summarizers; Users (providing effective feedback).



Agent-Based Data Management

Methods	Model	Memory	Tool	Planning	Extra Knowledge	Multi-Agent
Data Cleaning	GPT	Historical messages	Declarative APIs	Iterative	N/A	N/A
Q&A Diagnosis	GPT-3 Codex	Historical answers	Monitoring Tools	Iterative	Documents; Customer Tickets	N/A
Anomaly Diagnosis	GPT-4 Llama	Results of historical actions	Monitoring Tools; Optimization Tools	Tree of Thought	Documents	✓

Overview of LLM4DB Framework

□ Data Management tasks

□ LLM Prompt for Data Management

- Instruction Prompting
- Few-Shot Prompting

□ LLM Agent for Data Management

- Agent Models + Memory
- Reasoning / Planning Strategies
- Tool Management & Learning

□ RAG for Data Management

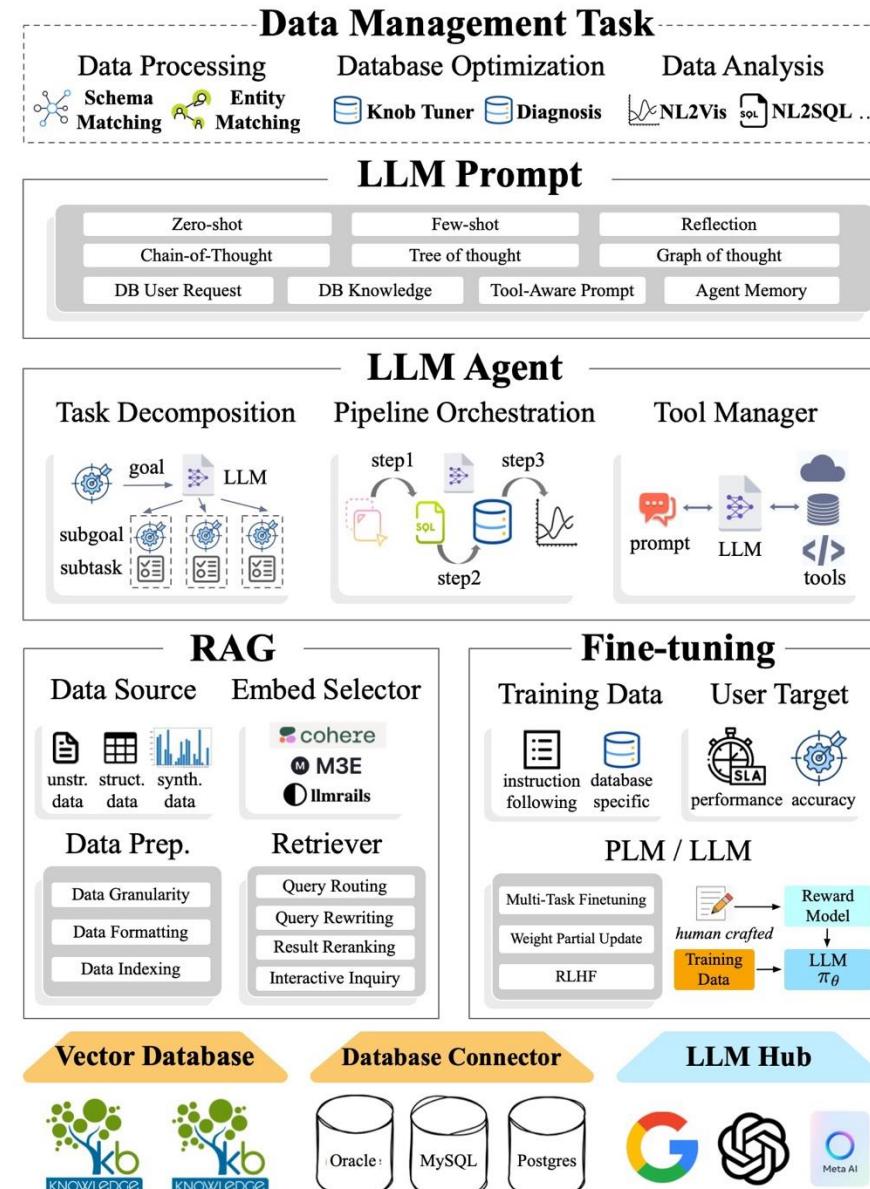
- Semantic Segmentation
- Result Retrieval
- Result Reranking

□ Finetuning for Data Management

- SFT Dataset Generation

□ Data Management for LLM

- LLM Data Preparation



Motivation of Retrieval Augmented Generation

Drawbacks of LLMs

- Hallucination
- Outdate information
- Low efficiency in parameterized knowledge
- Weak reasoning capability

Practical Requirements

- Domain-Specific Accurate Q&A
- Frequent Data Update
- Explainability of Generated Answer
- Controllable Cost
- Data Privacy Protection

Question: What color of my cat's eyes?

Correct Answer: Green.

Direct QA

Question ↗

LLM

↗

I don't know the color
of your cat's eyes.

RAG

Question ↗

LLM

↗

Green

"I have a cat. He has bright green eyes." **(Retrieved context)**

A motivative example.

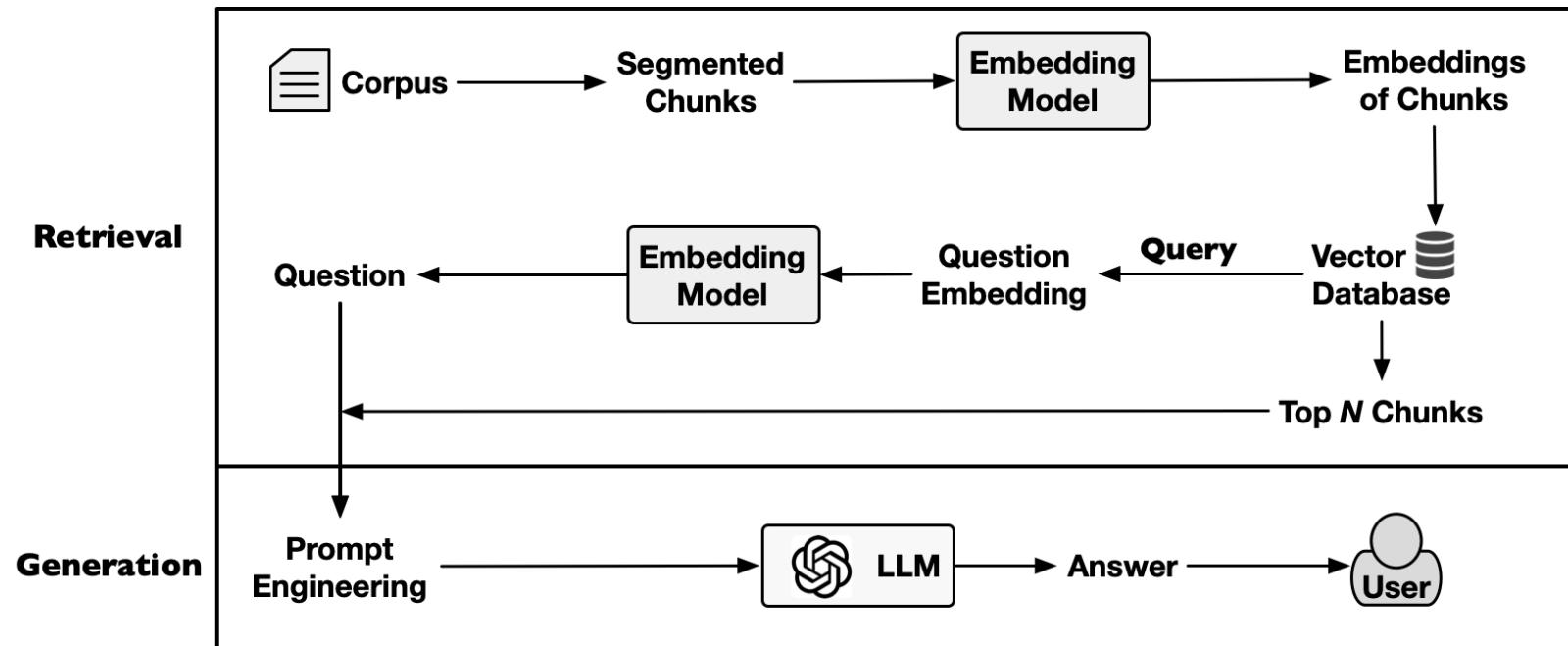
Retrieval and Generation Pipeline

Retrieval phase:

- The given corpus will be segmented into many chunks.
- For a question, a retriever identifies and selects the top K most related chunks as context.

Generation phase:

- The question, alongside the context, will be inputted into a LLM to generate an answer.



Limitations of RAG

Retrieval limitation:

- Chunk selection based on embedding similarity may lead to misleading or irrelevant chunks and missing crucial information.
- Methods:
 - *Semantic segmentation*
 - *Reranking*
 - *Retrieval techniques*

Generation limitation:

- LLMs may produce content that is not supported by the retrieved context.
- Methods:
 - *Prompt engineering*
 - *Using LLMs with high proficiency levels*

Semantic segmentation

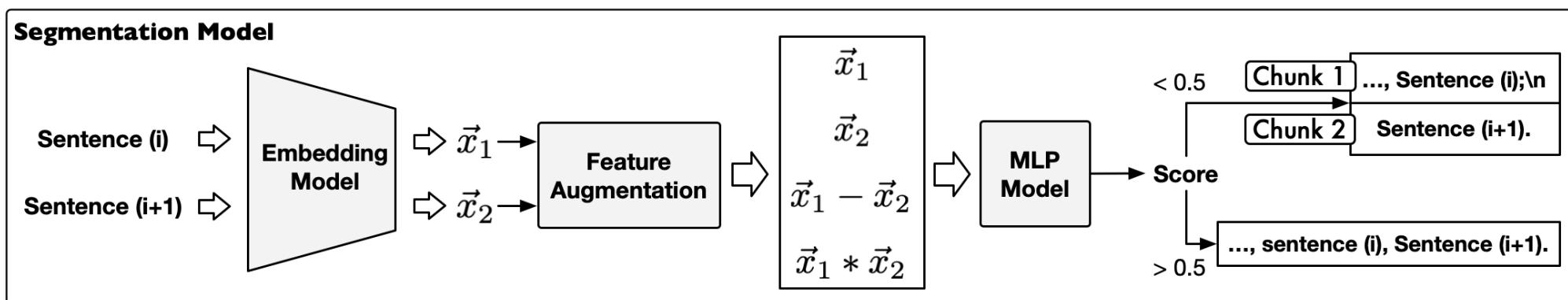
Motivation: Segment the corpus into semantically coherent chunks, ensuring that the retrieved information is semantically complete and relevant.

Training:

- Collecting *sentence pairs* with *labels*. (*label=1 means they are semantically relevant.*)
- Fitting the mapping between sentence pairs and labels using a model (Embedding model + MLP).

Inference:

- Each two adjacent sentences in the corpus is judged by the segmentation model.



Reranking Technique

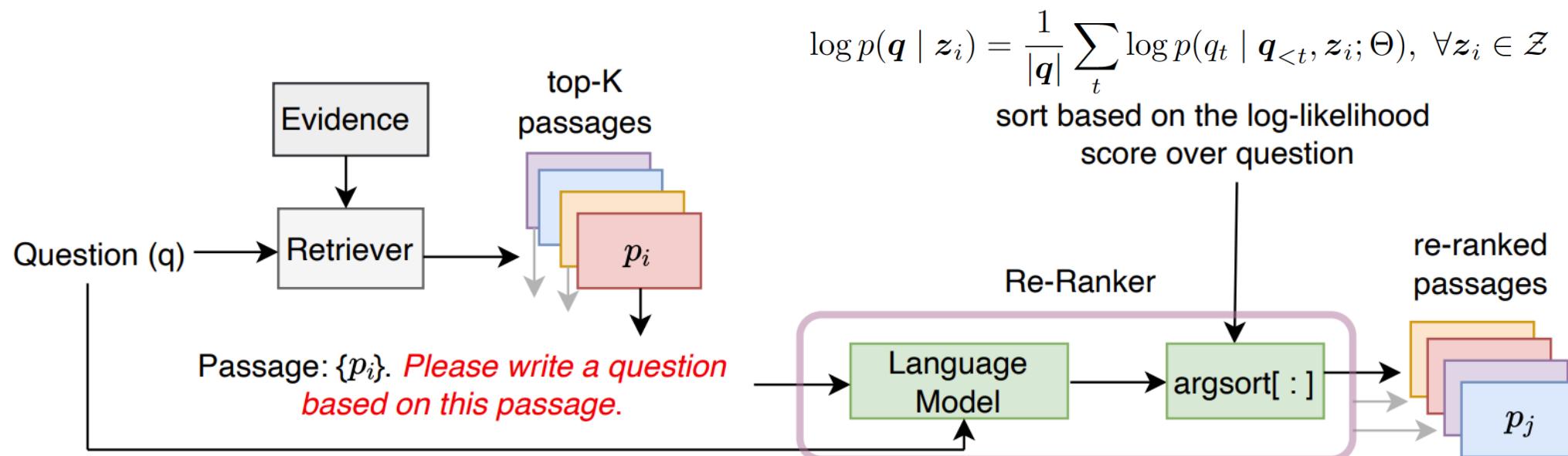
Reranking: Reorder chunks according to their relevance to the question, and then select the top K based on specific metric (e.g. relevance).

- Enhancer: Boost the relevance and quality of context.
- Filter: Filter out misleading or irrelevant chunks.

Example:

UPR | EMNLP 2022

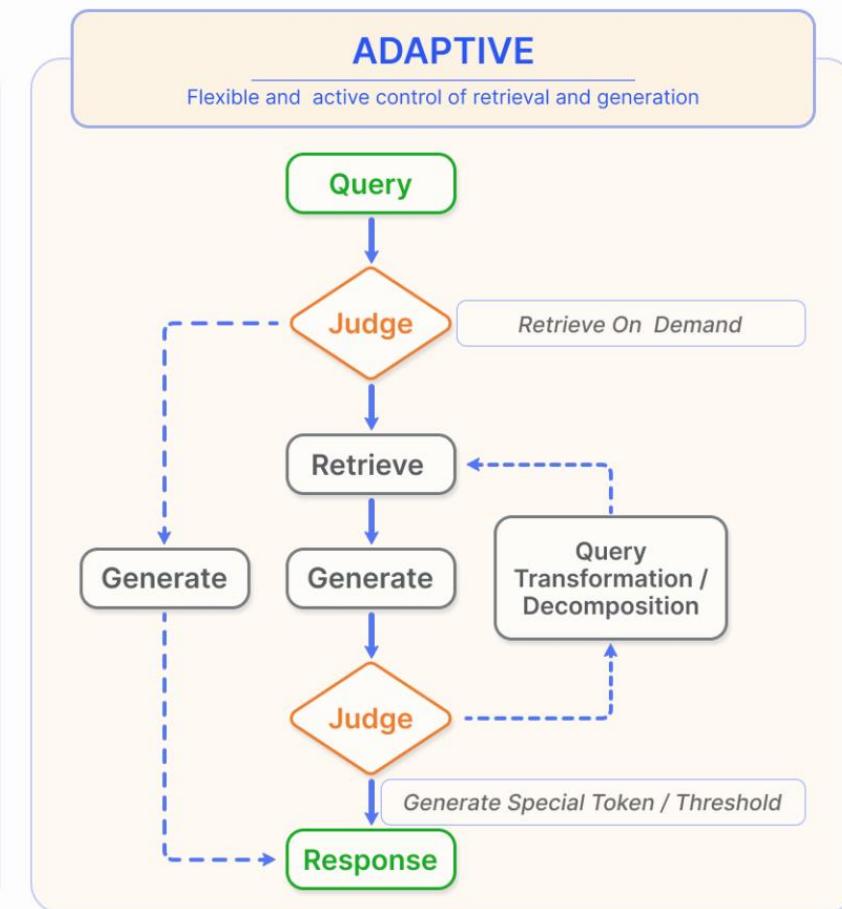
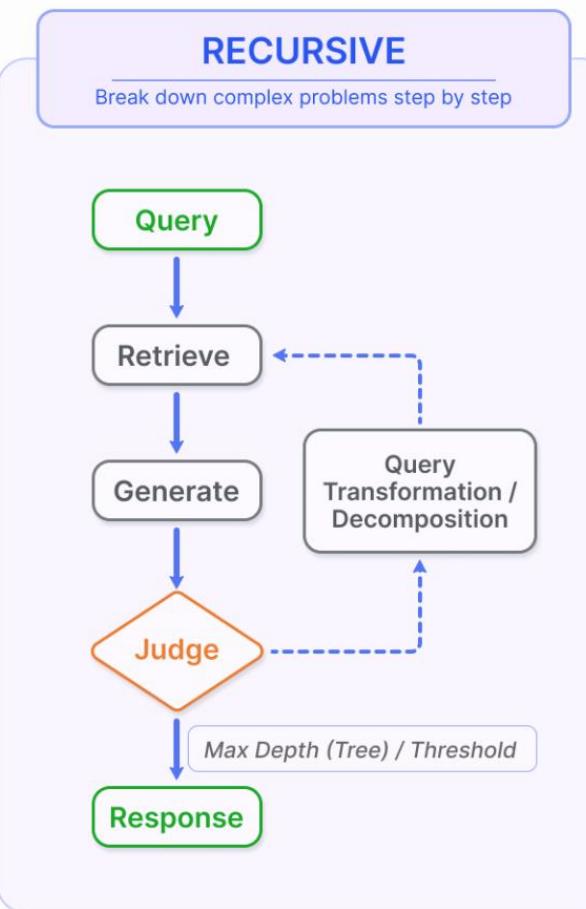
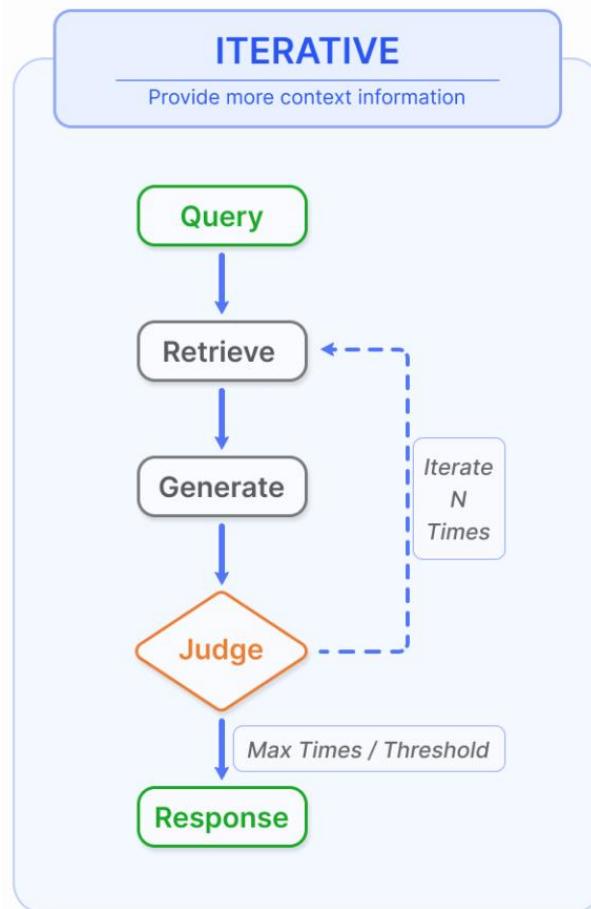
Using a PLM to compute the probability of the input question conditioned on a retrieved passage.



Retrieval Technique

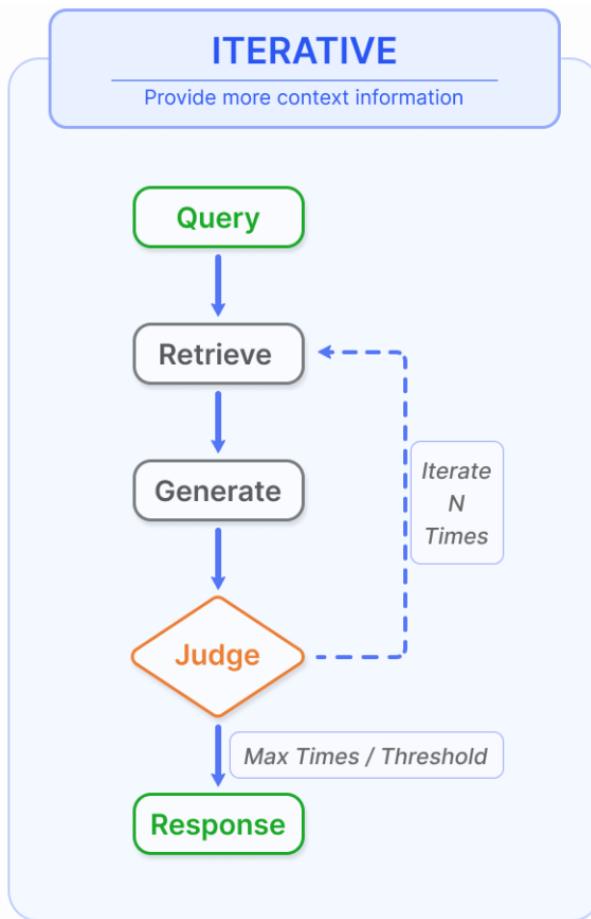
Retrieval augmentation

- Iterative Retrieval
- Recursive Retrieval
- Adaptive Retrieval



Retrieval Technique

- **Iterative Retrieval:** Repeatedly search based on the initial query and the generation result
Advantage: Enhance robustness of subsequent answer generation.



ITER-RETEGEN | EMNLP 2023

ITER-RETEGEN will iterate for specific times, in each iteration, it uses **generated content and the initial query** to retrieve.

Question: q

What is the height of the player who won the 2015 AFL Rising Star award?

Retrieval: $q \rightarrow \mathcal{D}_q$

Title: 2015 AFL Rising Star Context: The NAB AFL Rising Star award is given annually to a stand out young player in the Australian Football League. **The award was won by Jesse Hogan of Melbourne**

Retrieval-Augmented Generation: $\mathcal{D}_q \parallel q \rightarrow y_1$

The 2015 AFL Rising Star award was won by Jesse Hogan of Melbourne. Jesse Hogan is a professional Australian rules footballer. He is 198 cm tall. So the answer is 198 cm

Iteration 1

Question: q

What is the height of the player who won the 2015 AFL Rising Star award?

Generation-Augmented Retrieval: $y_1 \parallel q \rightarrow \mathcal{D}_{y_1 \parallel q}$

Title: Jesse Hogan Context: Jesse Hogan ... playing for the Melbourne Football Club. A key forward, **Hogan is 1.95 m tall** ... made his AFL debut in the 2015 season and won the Ron Evans Medal as the AFL Rising Star

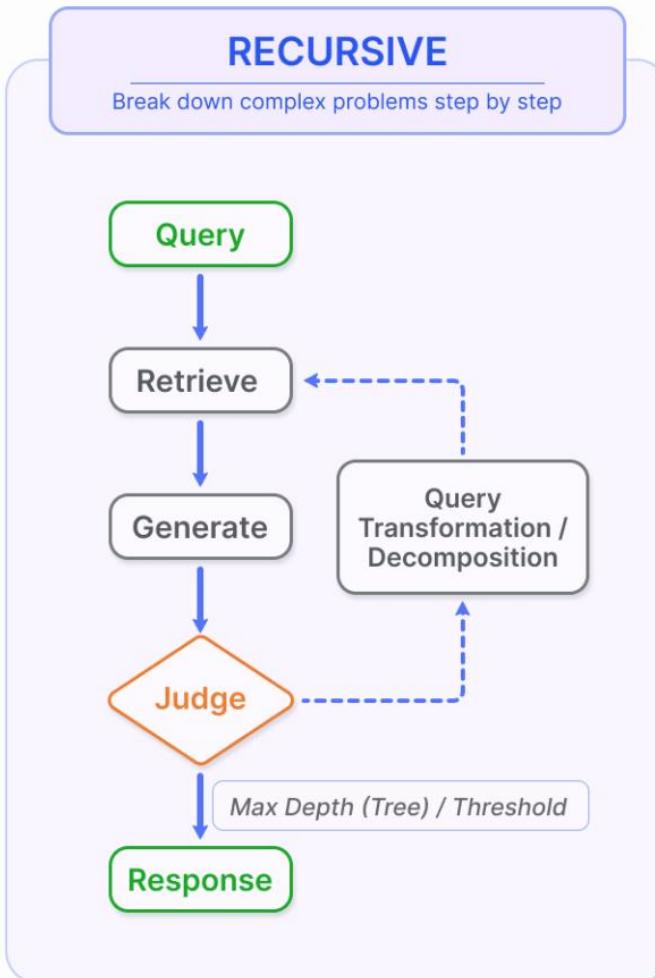
Retrieval-Augmented Generation: $\mathcal{D}_{y_1 \parallel q} \parallel q \rightarrow y_2$

The 2015 AFL Rising Star award was won by Jesse Hogan of Melbourne. Jesse Hogan is 1.95 m tall. So the answer is 1.95 m

Iteration 2

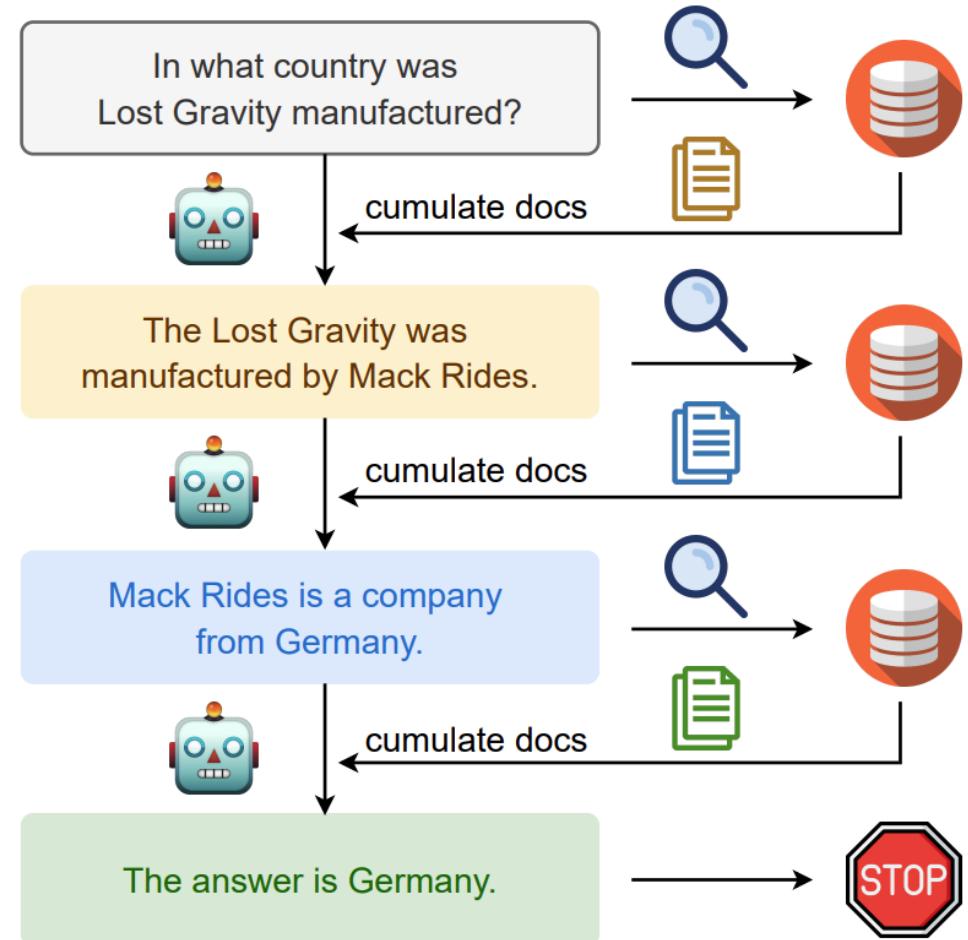
Retrieval Technique

- **Recursive Retrieval:** Iteratively refining search queries based on previous retrieval result.
Advantage: Improve the depth and relevance of search results.



Example: IRCoT will iteratively repeat two steps:

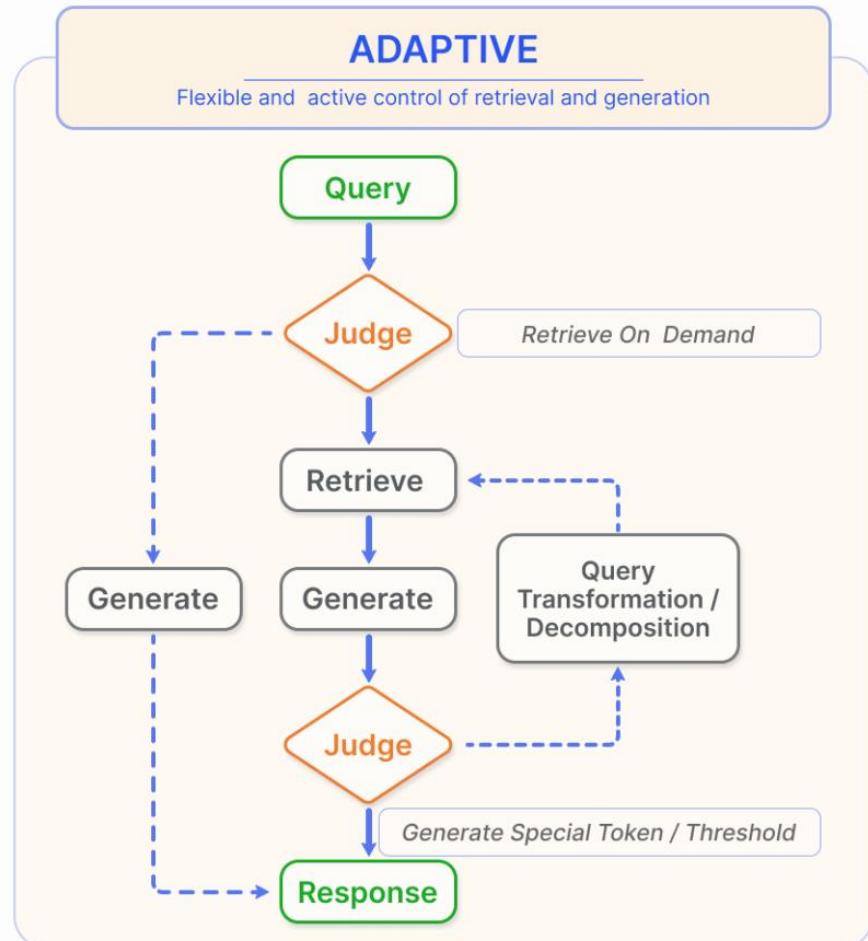
- **Generate CoT (chain-of-thought) question** based on retrieved corpus and question.
- **Retrieve** over the previous CoT question.



Retrieval Technique

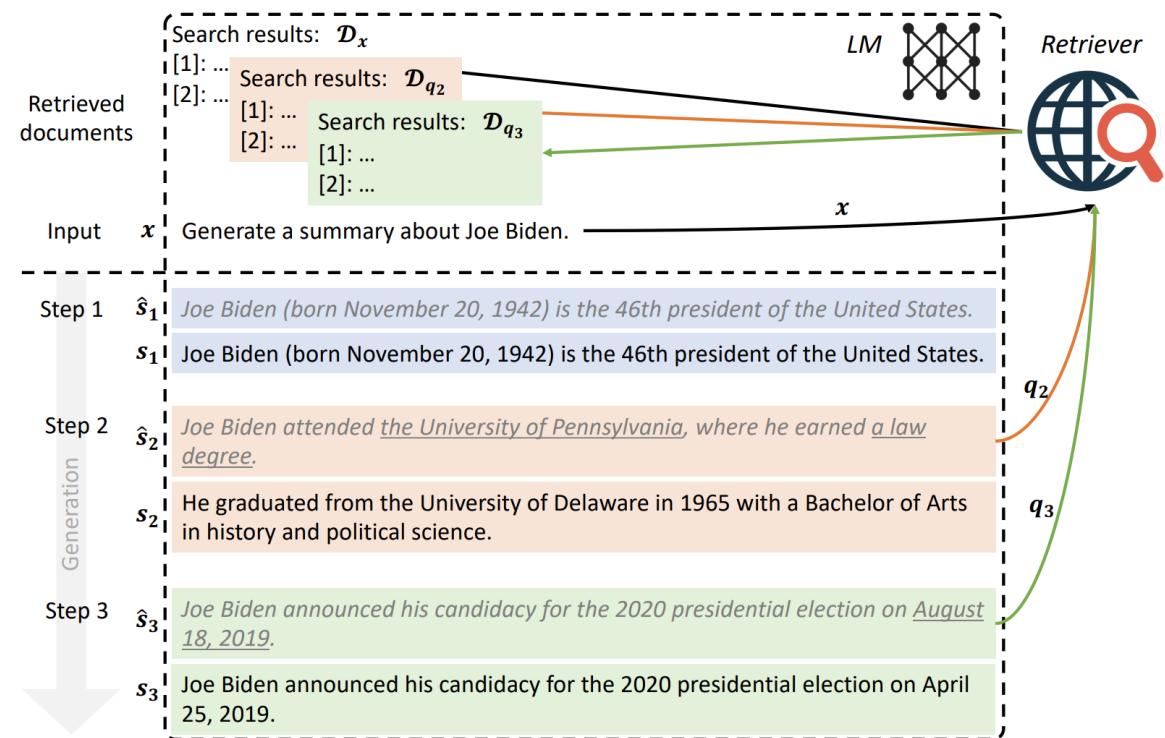
- **Adaptive Retrieval:** Enable LLMs to actively determine the moments for retrieval.

Advantage: Enhance the efficiency and relevance of the information sourced.



Example: FLARE

Retrieve when LLM generates tokens with probabilities below a set threshold.



Insights with experimental examples

- Noisy chunks retrieved considerably undermine the effectiveness of RAG systems.

Question: Who are genetically considered “kin”?

Option_1: [Full siblings]

Option_2: [All humans]

Option_3: [Adoptive children and full siblings]

Option_4: [Friends]

Noisy Chunks

Because nowadays, copies of these genes do reside in non-kin in your next-door neighbor and, for that matter, your worst enemy.

..., But in truth, you share virtually all your genes with any randomly selected homo sapien on any continent.

Genes that natural selection fully endorsed long ago--the basic genes for hunger, for lust, for familial love--are in everyone.

Scores of Chunks

Chunk 1	13.79
Target Chunk	13.58
Chunk 3	11.91
Chunk 4	11.55
Chunk 5	10.94
Chunk 6	7.815
Chunk 7	7.665
Chunk 8	5.490
Chunk 9	4.416
Chunk 10	1.304
Chunk 11	0.800
Chunk 12	0.255
Chunk 13	0.198
Chunk 14	0.093
Chunk 15	0.089

Get the correct answer [Option_1] when $2 \leq K \leq 10$.

Might get wrong answers when $11 \leq K \leq 13$.

Get the wrong answer [Option_2] When $K = 14$.

Insights with experimental examples

- Precise retrieval is a predominant part in RAG.
- The proficiency level of LLMs plays a crucial role in RAG.
- Embedding models, though useful, are not as important as LLMs.

COMPARISON ON THE QUALITY DATASET (USING GPT-4).

Metric Model	Accuracy in Test Set	Accuracy in Hard Set
GPT-4	77.2%	70.3%
RAPTOR+GPT-4	82.6%	76.2%
SAGE +GPT-4	90.10%	76.3%

ACCURACY ON QUALITY DATASET WITH DIFFERENT LLMs.

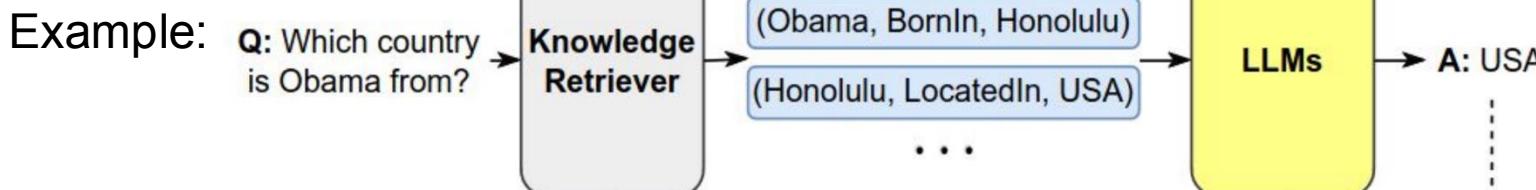
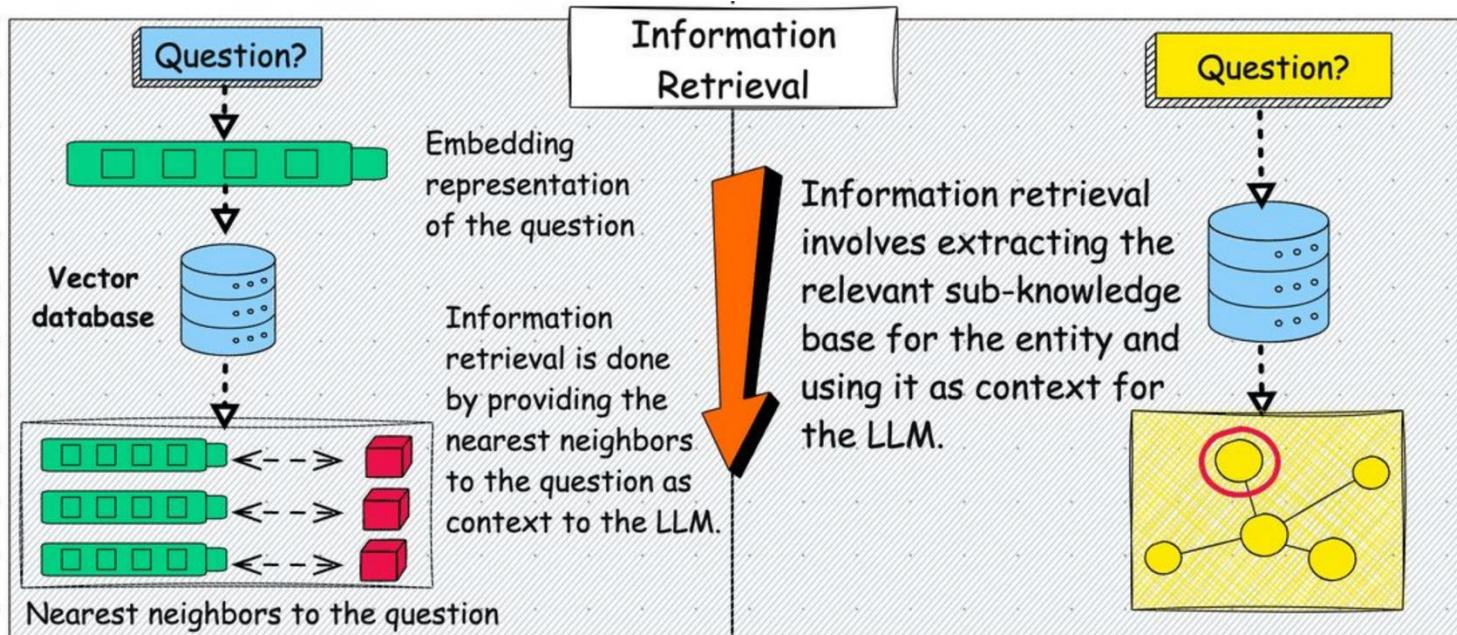
Metric Model	GPT-3.5 Accuracy	GPT-4o-mini Accuracy
BM25	62.70%	73.50%
DPR	60.4%	73.0%
SAGE	64.5%	77.1%

EFFECTIVENESS EVALUATION ON QUALITY AND QASPER DATASET
(USING GPT-4O-MINI).

Metric Model	Accuracy (QuALITY)	F1-Match (QASPER)
SBERT	72.48%	37.57%
BM25	72.18%	37.30%
DPR	72.38%	37.41%
OpenAI Embedding	75.32%	38.94%

Graph RAG

- Graph RAG includes a **graph database** as a source of the retrieval context sent to the LLM.
Retrieval Method: Entity Linking & NL-to-Graph-query.



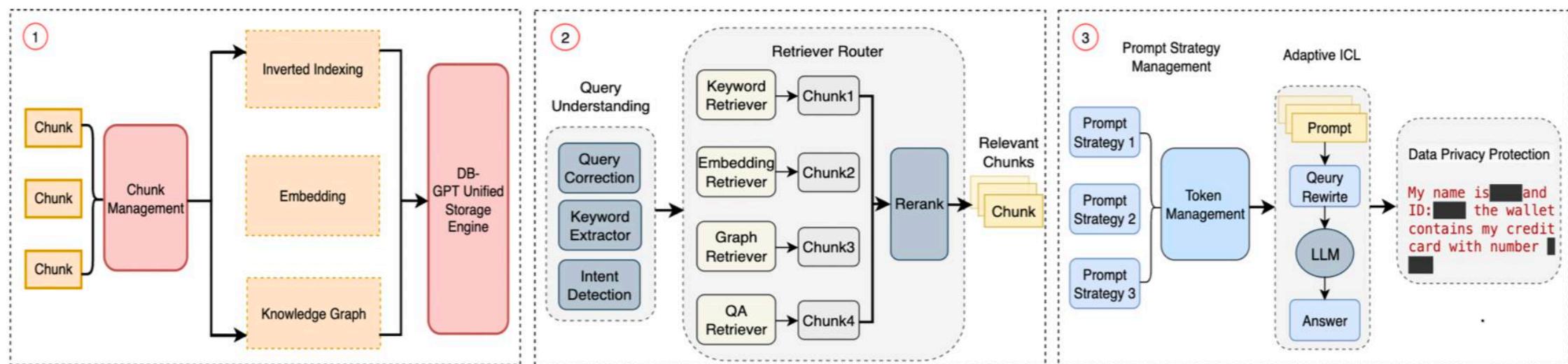
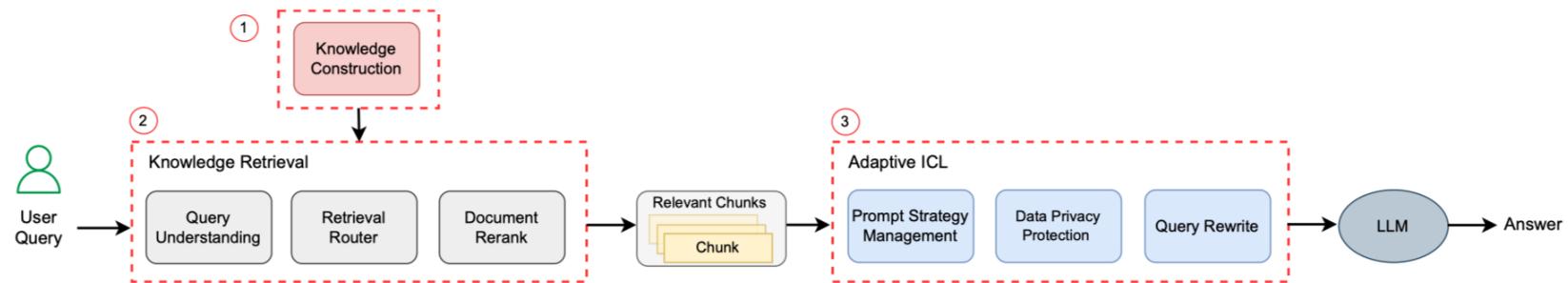
Advantages:

- Providing extra information like metadata for generation.
- Handling complex and nuanced queries.
- Supporting data update.

RAG for Text2SQL

Multi-source RAG for QA

- Knowledge construction
- Knowledge retrieval
- Learning to Embed and Search
- Adaptive ICL and Generation by LLM
 - ICL is a technique used to improve LLMs' performance in handling contextual information by incorporating additional context during the training or inference phase



Overview of LLM4DB Framework

□ Data Management tasks

□ LLM Prompt for Data Management

- Instruction Prompting
- Few-Shot Prompting

□ LLM Agent for Data Management

- Agent Models + Memory
- Reasoning / Planning Strategies
- Tool Management & Learning

□ RAG for Data Management

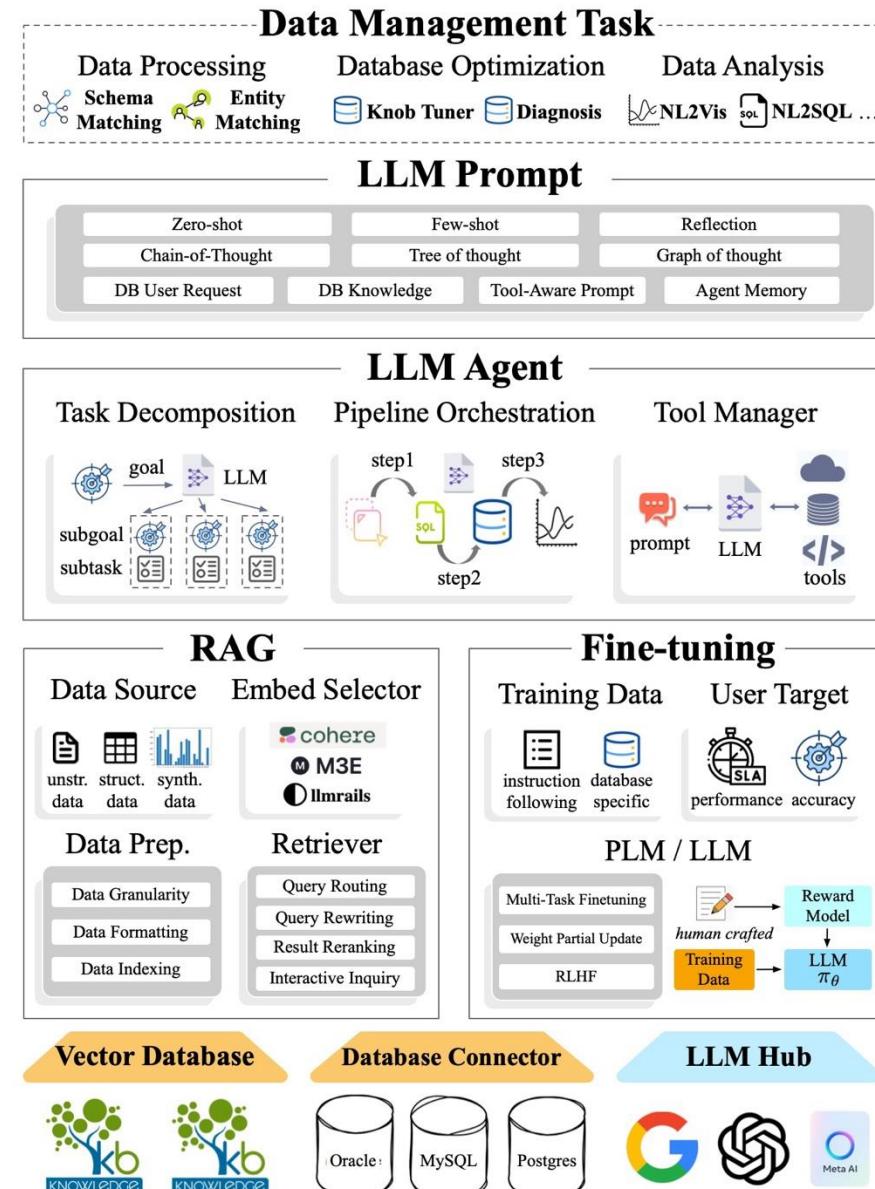
- Semantic Segmentation
- Result Retrieval
- Result Reranking

□ Finetuning for Data Management

- SFT Dataset Generation

□ Data Management for LLM

- LLM Data Preparation



LLM Finetuning Technique: Incremental Model Update

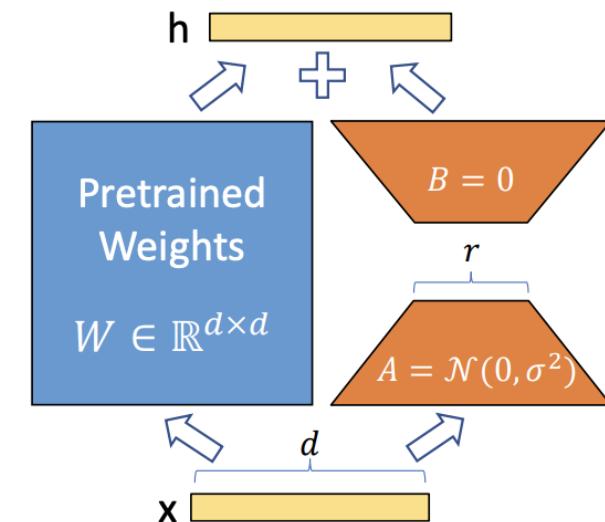
- Adjusting only a small number of parameters for adapting to new tasks → Low-Rank Adapter (LoRA)

$$W_{LoRA} = W_{orig} + \Delta W$$

- W : FC or attention layer in LLM
- B, A : Train the low-dimensional matrices $\rightarrow \Delta W = BA$
- ΔW : with much fewer parameters than W_{orig}

$$\begin{matrix} h = & \begin{matrix} \begin{matrix} 0.46 & 2.49 & -0.63 & -0.55 & -0.66 & \dots \end{matrix} \\ \begin{matrix} 1.74 & -0.81 & -1.38 & -0.67 & 0.2 & \dots \end{matrix} \\ \begin{matrix} -0.76 & -1.31 & -0.85 & 0.66 & -0.47 & \dots \end{matrix} \\ \begin{matrix} -1.31 & -1.39 & -1.41 & -1.64 & -0.79 & \dots \end{matrix} \\ \begin{matrix} -0.14 & 0.45 & -0.51 & 1.13 & 0.46 & \dots \end{matrix} \\ \dots & \dots & \dots & \dots & \dots & \dots \end{matrix} \end{matrix} + \begin{matrix} \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \dots & \dots \end{matrix} \cdot \begin{matrix} -0.7 & 0.74 & 0.06 & 1.11 & -0.46 & \dots \\ -0.96 & 0.71 & 0.42 & -0.4 & -0.88 & \dots \end{matrix} \end{matrix}$$

W **B** **A**



- Given basic dimension is 1024, and LoRA rank r as 4:
 - #-parameters of weight W is $1024 \times 1024 \approx 1M$
 - #-parameters of A, B are both $r \times 1024 \approx 4K$
- In this way, we only train 0.8% of the parameters to update LLM

LLM Finetuning Technique: Human Preference Alignment

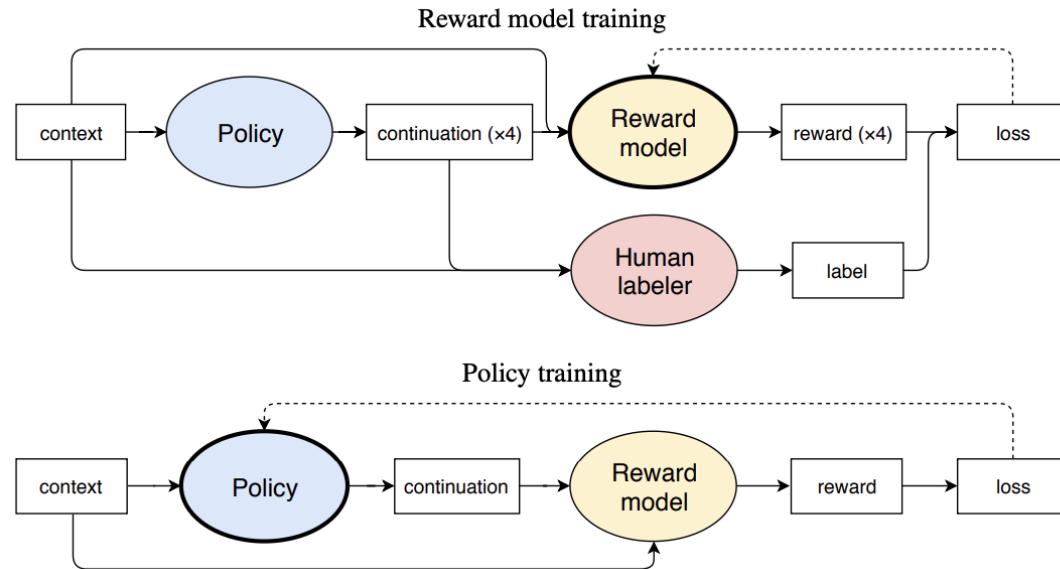
- Reinforcement learning from human / AI feedback (RLHF / RLAIF): Align an intelligent agent to human preferences by learning directly from human feedback

- Main steps:

- Collect demonstration data, and train a policy model (instruction tuning)
- Collect comparison data (labeled by humans / AI), and train a reward model
- Optimize a policy against the reward model using reinforcement learning

Title: Who is right here, me or my mother?
... I decided to get myself a better computer. At the same time, a friend of mine was getting rid of his computer ... I bought it for 5000kr and a monitor from somewhere else for approx 2000kr. Upon finding this out, my mother got really angry, and she has ordered me to return it to him. I didn't talk it over with her ... Anyway, she is saying that I legally can't make purchases like this **over 2000kr without telling her** because she is responsible ... Even though I used my money (mostly ... for the "mostly" part over: I was a little short on cash for the monitor, so 950kr of it was paid for with **a card the family is sort of sharing** ... She is also kinda paranoid about getting ripped-off ...

SFT Summary
I bought a gaming-PC without telling my mom, she is now mad and wants me to return it.



RLAIF Summary

I bought a used gaming-PC for **7000kr** without telling my mom. She is now demanding that I return it. I used my own money, but some of it was from **a card that the family is sharing**. She is also paranoid about getting ripped-off. **Who is right here?**

LLM Finetuning for Query Rewrite

□ Problems of Prompt-Only Rewriters

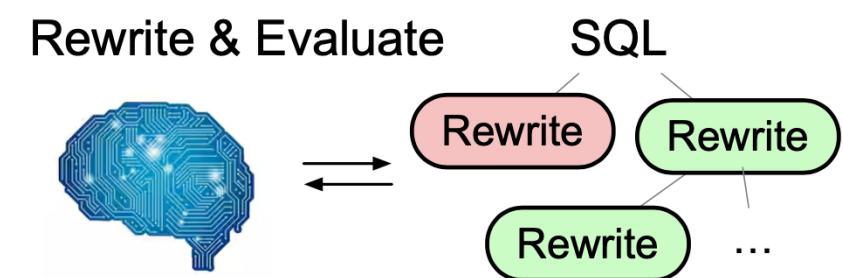
- Prompt engineering has been criticized for **limited knowledge capacity and unstable performance**

□ Finetuning for Query Rewrite

- Dataset Generation:** Given rewrite rules, generate new SQLs using the provided schemas and estimate whether RewriteLLM has effectively mastered the rule:

p_{aug} = ... Rule ... is ... For example, the given table schemas are: ... Some sample SQLs would be: ... From the example, first, you should summarize the conditions of the rule by analyzing why it can match the sample SQLs. Second, you should learn how to create SQLs that match these conditions. ...

```
SELECT ...
WHERE ...
AND (
  EXISTS (
    SELECT a2 FROM t2
    WHERE t2.b2=1
  )
  OR
  EXISTS (
    SELECT a2 FROM t2
    WHERE t2.b2=2
  )
)
```



LLM Finetuning for Query Rewrite

□ Problems of Prompt-Only Rewriters

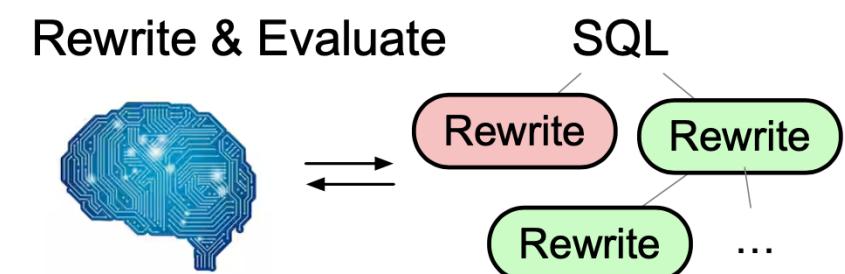
- Prompt engineering has been criticized for limited knowledge capacity and unstable performance

□ Finetuning for Query Rewrite

- Confidence Evaluation:** Denote the generated SQL $x[1 : t]$ and the conditional LLM possibility $\pi_\theta(\cdot)$, the perplexity is:

$$\exp - \frac{1}{t} \sum_i^t \log \pi_\theta(x_i | x_{<i})$$

```
SELECT ...
WHERE ...
AND (
  EXISTS (
    SELECT a2 FROM t2
    WHERE t2.b2=1
  )
  OR
  EXISTS (
    SELECT a2 FROM t2
    WHERE t2.b2=2
  )
)
```

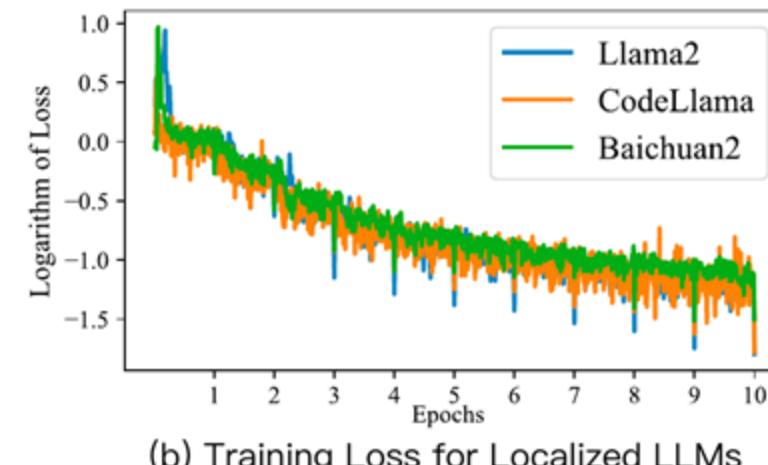


LLM Finetuning for Anomaly-Based Diagnosis

□ Finetune Localized LLM with the samples labeled by GPT-4

- S1. For the given anomalies, we **collect prompts used by D-Bot (GPT-4)** in the diagnosis process (e.g., *1000+ prompts*).
- S2. Consider the differences between LLMs (e.g., prompt preferences, max length limits), we **adjust the prompt format** to ensure compatibility with the target LLM (e.g., *modifying separators to those commonly used in the Llama series, such as '[INST]' and '[/INST]'.*)
- S3. We fine-tune the LLM on the dataset (e.g., 10 epochs), observe the convergence of the loss function, and **evaluate the accuracy and readability of responses** on the validation set, selecting the best fine-tuning checkpoint as the result LLM.

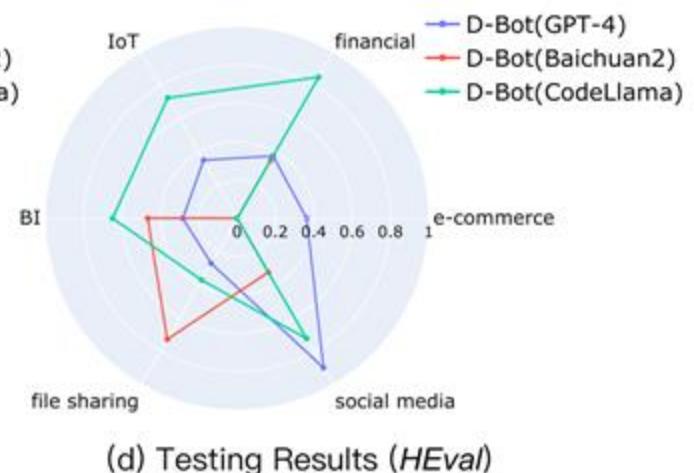
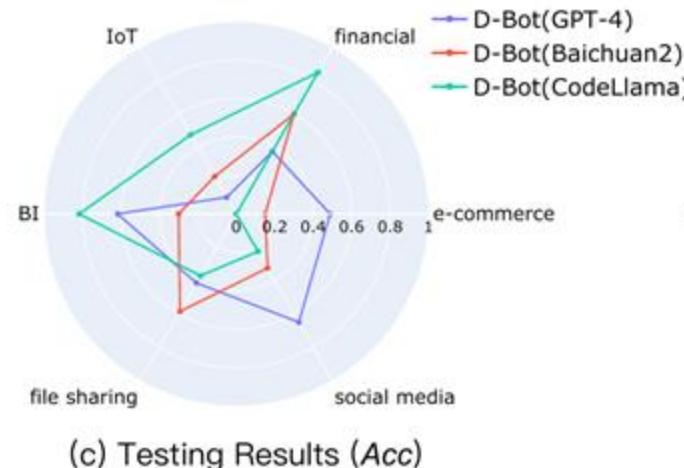
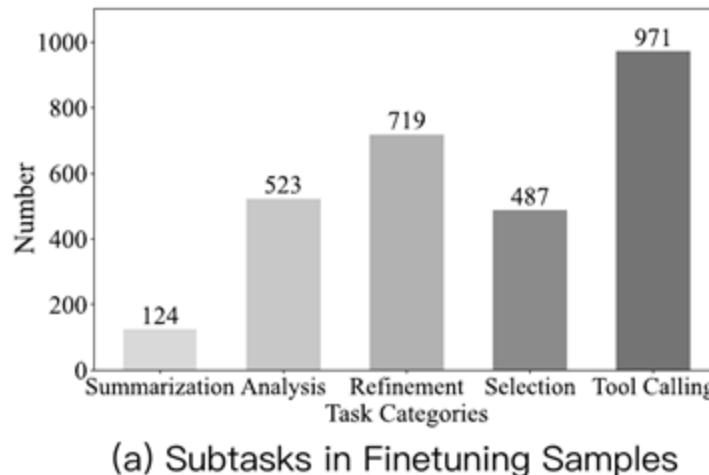
*The training loss obviously decreases
(but some epochs can overfit)*



LLM Finetuning for Anomaly-Based Diagnosis

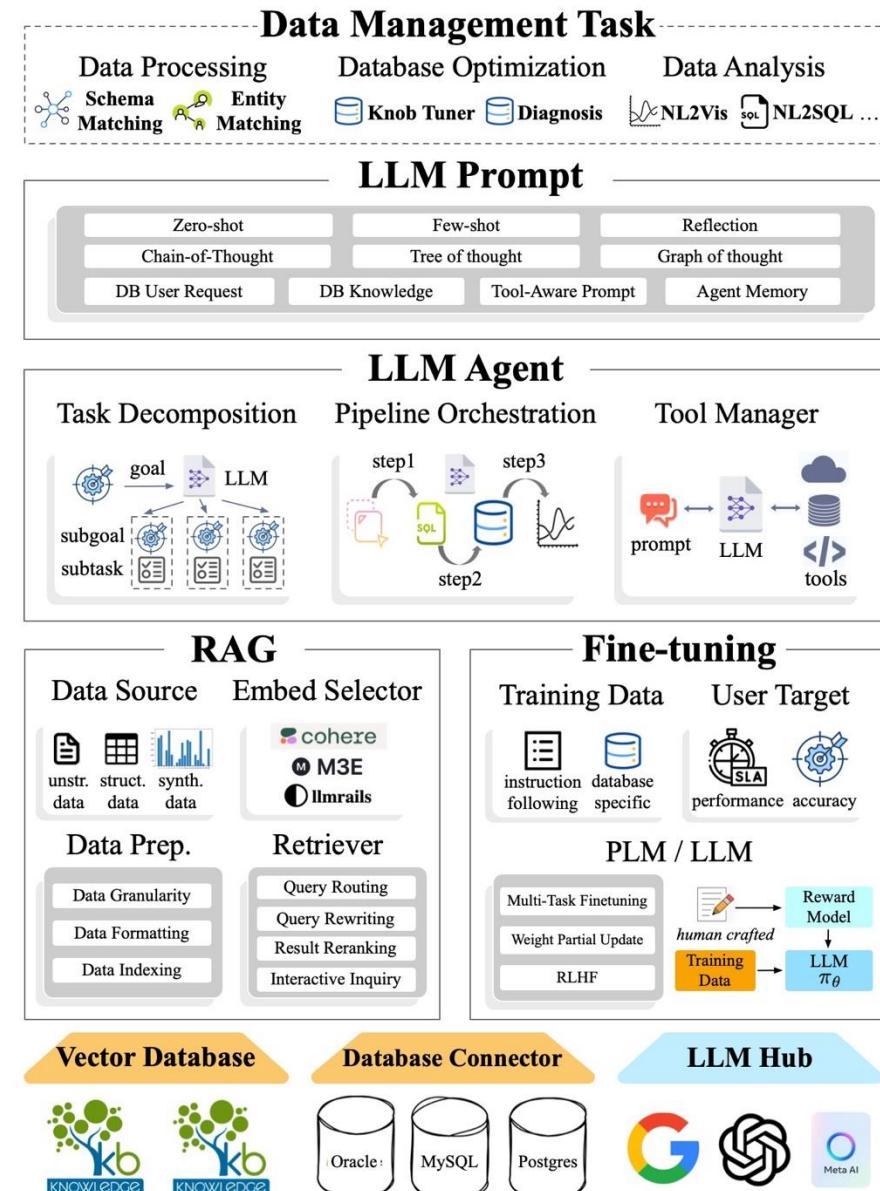
❑ Localized LLM: Comparable or higher performance than Dbot-GPT4

- 5 diagnosis subtasks and 2819 finetuning samples
- Good testing performance: Good at subtasks (e.g., tool calling, abnormal metric analysis) ; but highly rely on the quality of finetuning data



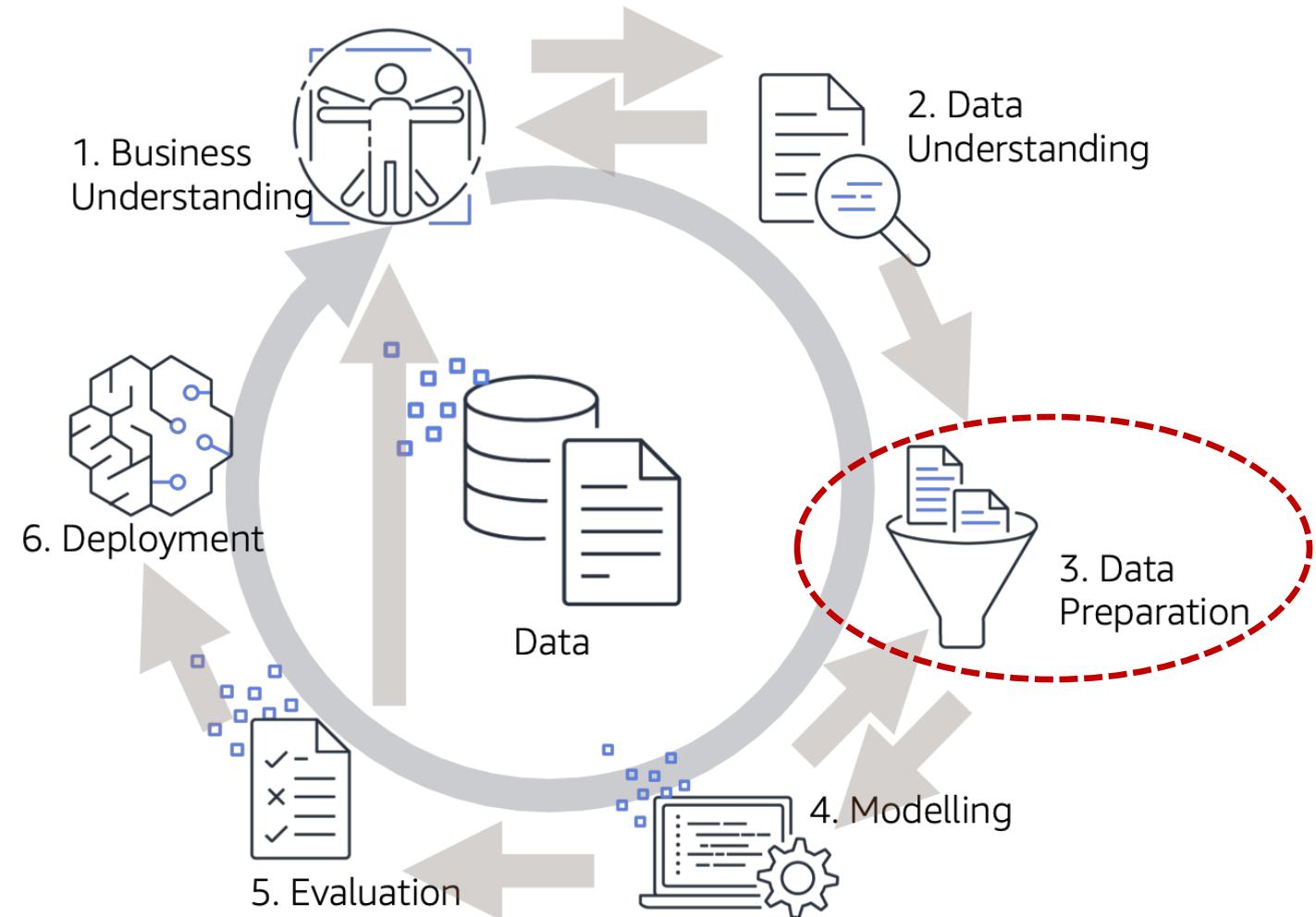
Data4LLM

- Data Management tasks
- LLM Prompt for Data Management
 - Instruction Prompting
 - Few-Shot Prompting
- LLM Agent for Data Management
 - Agent Models + Memory
 - Reasoning / Planning Strategies
 - Tool Management & Learning
- RAG for Data Management
 - Semantic Segmentation
 - Result Retrieval
 - Result Reranking
- Finetuning for Data Management
 - SFT Dataset Generation
- Data Management for LLM
 - LLM Data Preparation



Data Preparation in machine learning life cycle

- Data Preparation: The prerequisite to building high-performance model
 - Turn big dirty data into a subset of good data
 - Select, clean, augment, label, mix, and even synthesize data



Data preparation for
machine learning using Amazon Timestream

Data Preparation in machine learning life cycle

- Data Preparation: Turn **big dirty data** into **a subset of good data**



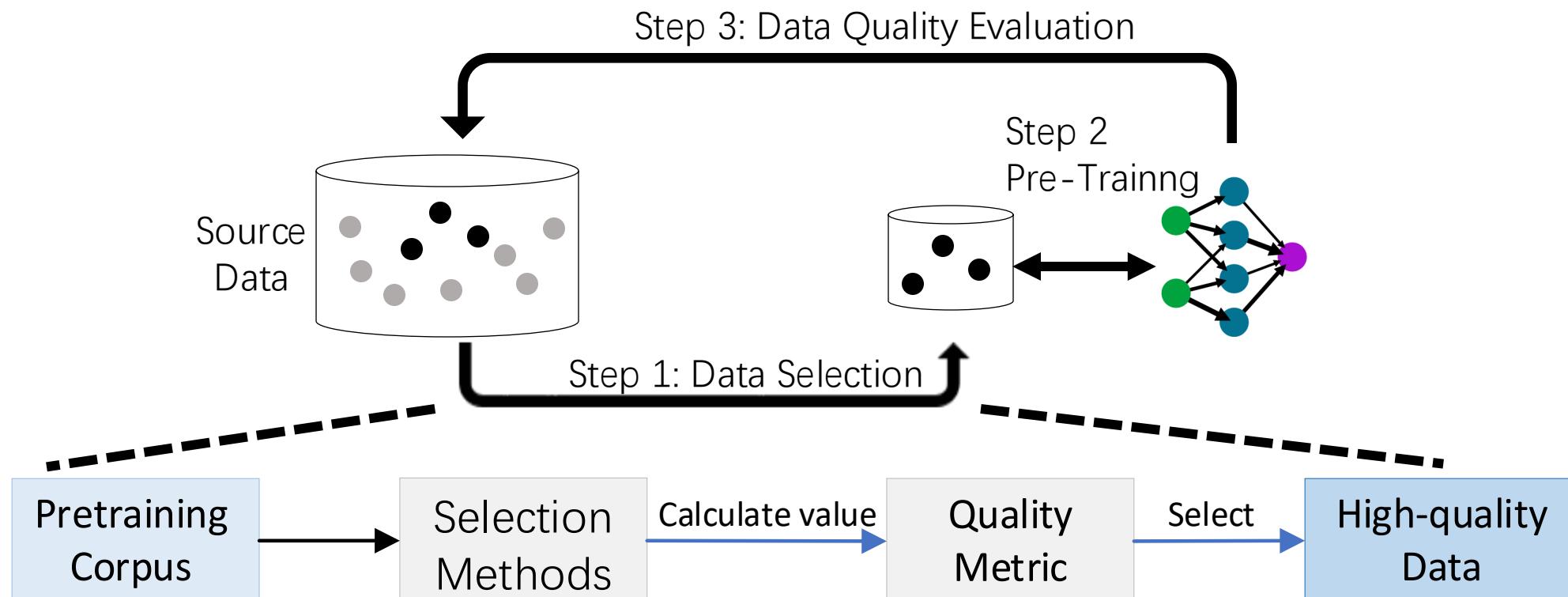
Data Preparation Pipeline

- **Challenges**

- Rely on experts
- Time-consuming
- Hard to discover the optimal solution
- E.g., numerous candidate pipelines

Data Selection for LLM

- **Data Preparation: Turn big dirty data into a subset of good data**
 - **Data Selection:** Obtain reduced representation in volume but produce similar or even better training results



Data Selection For LLM

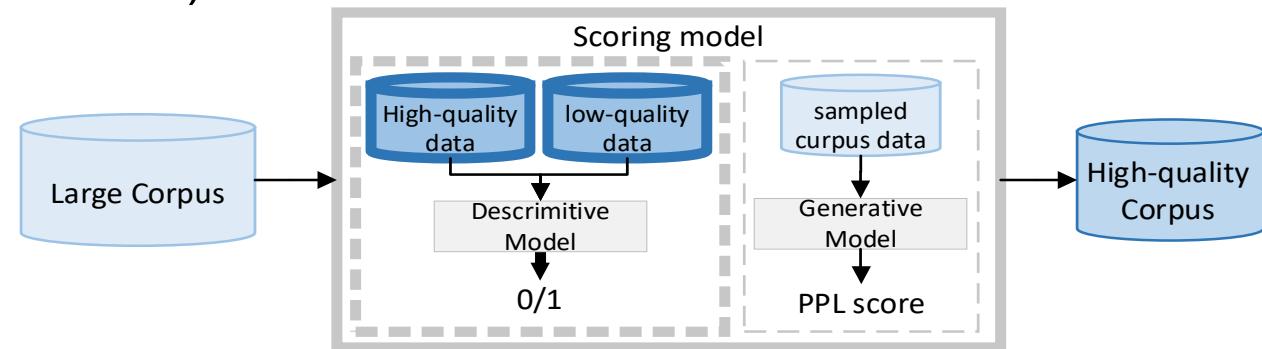
Challenge: How to select high-quality pretraining datasets?

- **Content-based Selection:** Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- **Classification-based Selection:** Identify data points that are likely from the same (or similar) distribution as a known “high-quality” corpus of data points
 - **Step 1: Feature Hashing**
 - Consider text words "the", "quick", "brown", "fox". Using a hashing function, these might be mapped to indices [5,17,3,12] in a feature vector of size 20.
 - **Step 2: Train Classifier with Curated / Other Pages**
 - Class 1 (Curated Content): High-quality sources like Wikipedia, books, and selected websites.
 - Class 2 (Other Webpages): Typical webpages found on the internet.
 - **Step 3: Score with the Well-Trained Classifier**
 - Assigns a quality score to webpages by how similar their content is to the Curated class.
 - **Step 4: Sample using Pareto Distribution**
 - Balances the inclusion of lower-quality pages to prevent bias:

Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- **Content-based Selection:** Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- **Perplexity-based Selection:** Train an LLM and evaluate on the data to achieve higher selection performance
 - Sentence example:
 - “*I love machine learning*”
 - Calculate conditional probability
 - $P(i)=0.2$
 - $P(\text{love}|i)=0.1$
 - $P(\text{machine}|i,\text{love})=0.05$
 - $P(\text{learning}|i,\text{love},\text{machine})=0.01$
 - $M=4$



A model with probability distribution P predicting a sequence of N words w_1, w_2, \dots, w_N

$$PP(W) = 2^{-\frac{1}{N} \sum_{i=1}^N \log_2 P(w_i|w_1, \dots, w_{i-1})}$$

Lower perplexity means the model's probability distribution is closer to the true data distribution

Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- **Content-based Selection:** Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)

- **Perplexity-based Selection:**

- Calculate the average value of logarithmic probabilities
 $\log P(i) = \log 0.2$

$$\log P(\text{love}|i) = \log 0.1$$

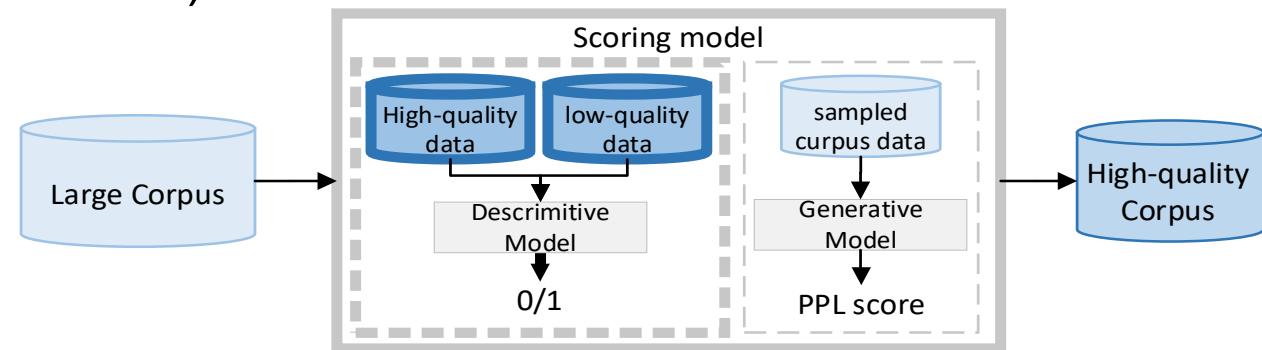
$$\log P(\text{machine}|i, \text{love}) = \log 0.05$$

$$\log P(\text{learning}|i, \text{love}, \text{machine}) = \log 0.01$$

$$\frac{1}{4}(\log 0.2 + \log 0.1 + \log 0.05 + \log 0.01) \approx -2.8782$$

- Calculate perplexity

$$\text{Perplexity}(P) = \exp(-(-2.8782)) \approx 17.77$$



A model with probability distribution P predicting a sequence of N words w_1, w_2, \dots, w_N

$$PP(W) = 2^{-\frac{1}{N} \sum_{i=1}^N \log_2 P(w_i|w_1, \dots, w_{i-1})}$$

17.77 is low perplexity (compared to 77.9) and indicates a high-quality sample

Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- **Content-based Selection:** Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- **Model-based Selection:** Use Model to rate multiple documents along various dimensions of perceived quality → *Capture human intuitions about data quality*
 - **Quality Criteria:**
 - **Writing style:** *With polished or beautiful words*
 - **Expertise:** *The difficulty level of the corpus*
 - **Facts & Trivia:** *With high density of long-tail factual knowledge*
 - **Educational value:** *Includes clear explanations, step-by-step reasoning, or questions and answers*

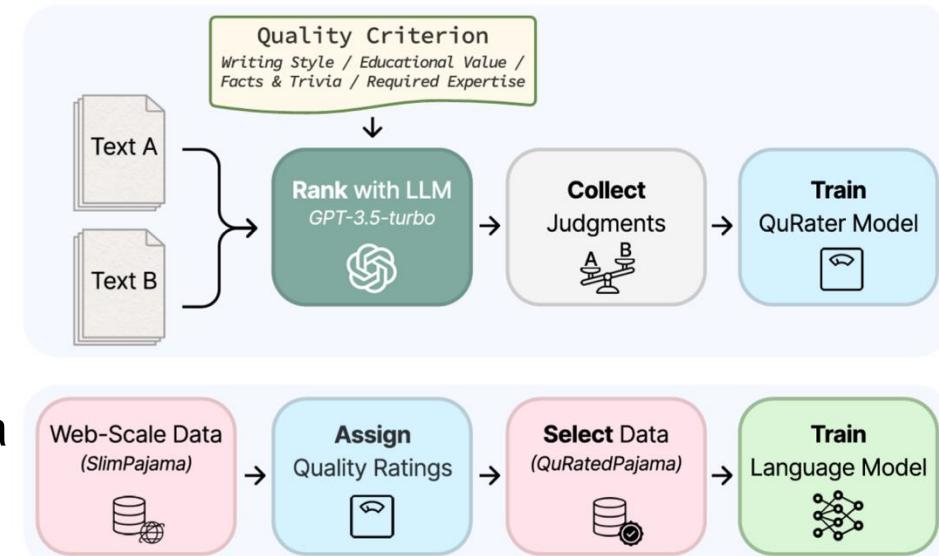
Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- **Content-based Selection:** Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- **Model-based Selection:** Use Model to rate multiple documents along various dimensions of perceived quality
 - 1. Sample text pairs (A, B) from a vast collection of documents
 - 2. With the criteria and a pair (A, B), LLM (e.g., GPT3.5) gives a confidence of B is better than A , i.e., $p_{B \succ A} \in [0, 1]$
 - 3. Generate a dataset of judgement

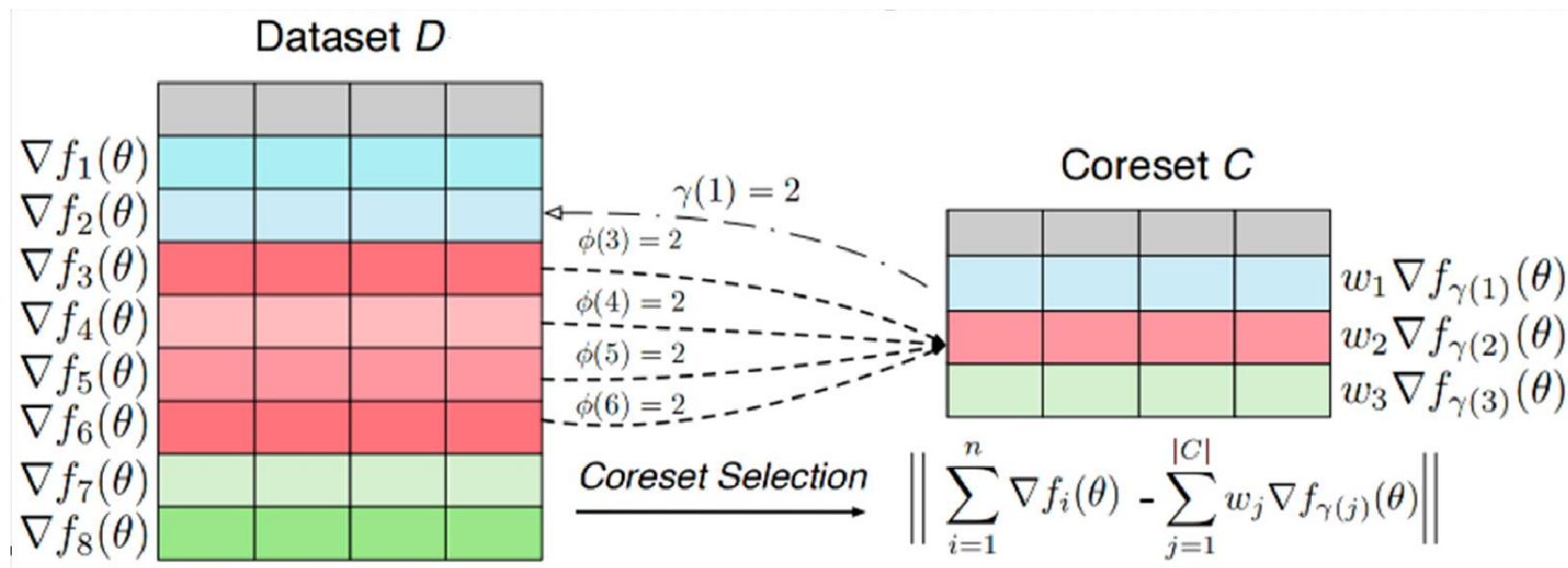
$$\mathcal{J} = \{(t_i, t_j, p_{i \succ j})\}$$

- 4. Fine-tune a 1.3B Sheared-Llama
 - Predict quality ratings under the four criteria



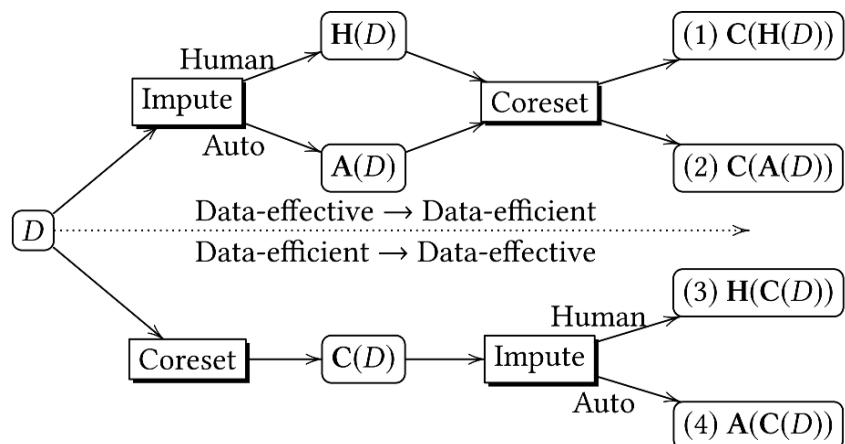
Gradient-based Data Selection

- **Algorithm: stochastic gradient decent**
- Data: Coreset
 - Given a large train set D , **Coreset** $C(D)$ is a core subset of D , which is selected to represent D such that $\mathbf{M}(C(D)) \approx \mathbf{M}(D)$, denoting that $C(D)$ has the same performance theoretically with D .



Gradient-based Data Selection

- **Intuitive baselines (sequential)**



Challenge

- Computing a good coresset from dirty data is to **accurately estimate the ground truth** of each missing value, which has multiple possible repairs.
- The **combinations of all possible repairs** constitute a huge search space.

- *First data-effective (impute) then data-efficient (coreset):*

- (1) Impute-Human: $H(D) \rightarrow \text{Coreset}: C(H(D))$
- (2) Impute-Auto: $A(D) \rightarrow \text{Coreset}: C(A(D))$

- *First data-efficient (coreset) then data-effective (impute):*

- (3) Coreset: $C(D) \rightarrow \text{Impute-Human}: H(C(D))$
- (4) Coreset: $C(D) \rightarrow \text{Auto-Human}: A(C(D))$

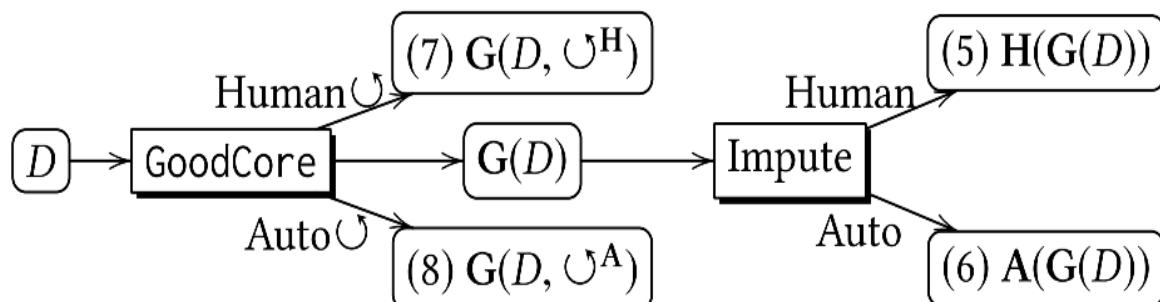
Solution	Accuracy	Human Cost	Machine Cost
(1) $C(H(D))$	High	High	Low
(2) $C(A(D))$	Low	None	Low
(3) $H(C(D))$	Low	Low	Low
(4) $A(C(D))$	Low	None	Low

Gradient-based Data Selection

Key idea of GoodCore

- Model the combinations of possible repairs as *possible worlds* of the original dirty data D
- Selecting *an expected optimal cores*et that can *represent the possible worlds of D* via gradient approximation without training in advance

$D \rightarrow G(D)$: Very Time-consuming

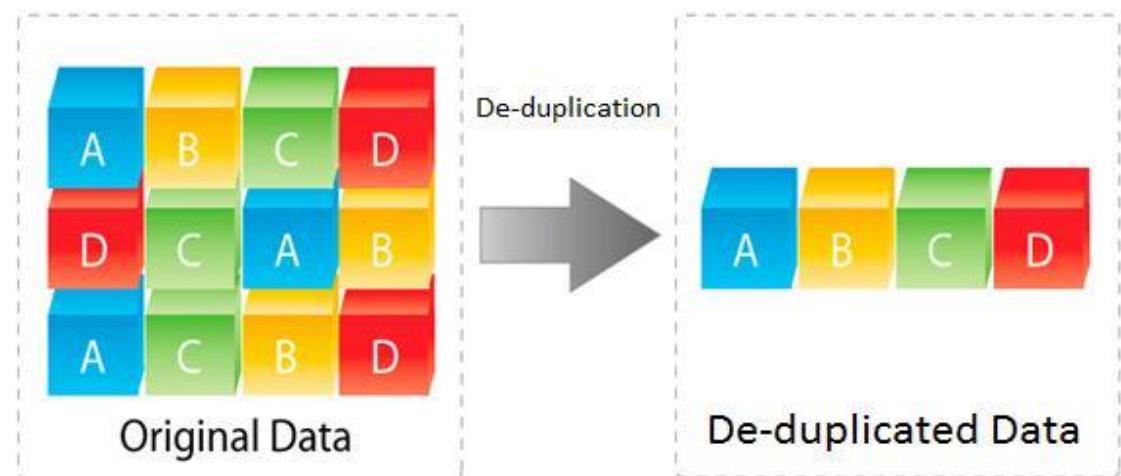


Solution	Accuracy	Human Cost	Machine Cost
(1) $C(H(D))$	High	High	Low
(2) $C(A(D))$	Low	None	Low
(3) $H(C(D))$	Low	Low	Low
(4) $A(C(D))$	Low	None	Low
Our goal	High	None or Low	Low
(5) $H(G(D))$	High	Low	High
(6) $A(G(D))$	Medium	None	High
(7) $G(D, \cup^H)$	High	Low	Low
(8) $G(D, \cup^A)$	Medium	None	Low

Data Cleaning For LLM

Data Preparation: Turn big dirty data into a subset of good data

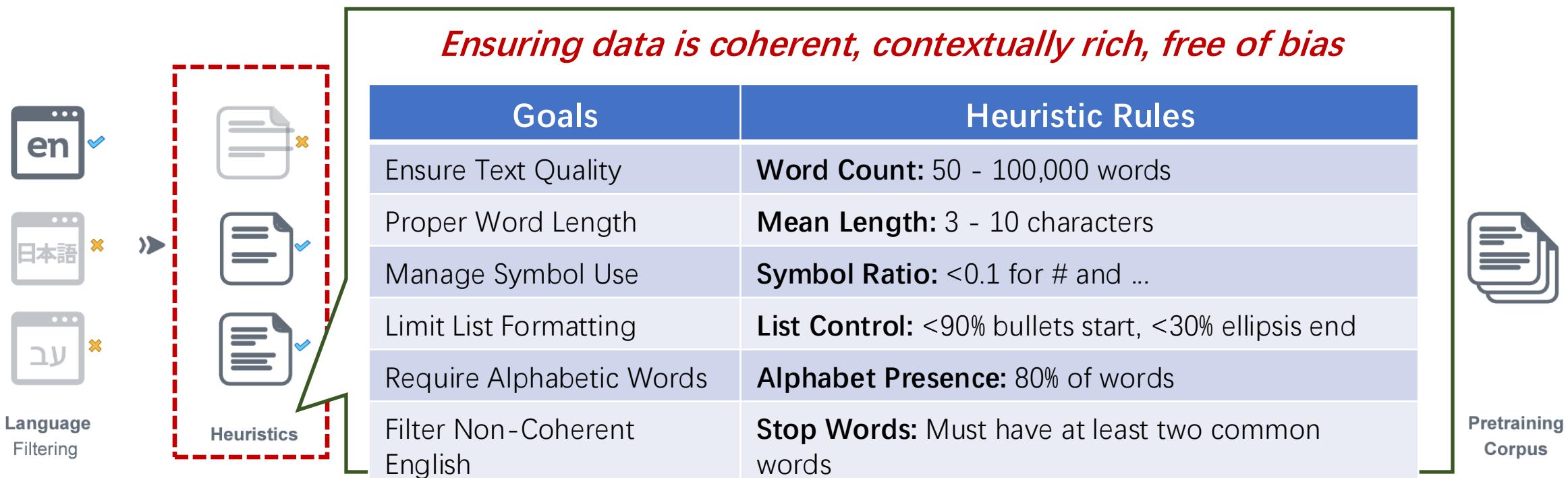
- **Data Cleaning:** Remove duplicate records; Remove (noise) outliers; Resolve inconsistencies; Fill in missing values (generally not conducted in LLM)
- **Data Deduplication:** Training on identical documents slows down training and may harm the performance → Identify same/similar documents and retain one
- **Exact Matching:** Leverage MD5 hashing to ensure documents are identical.
- **Near Matching:** Use min-hash/sim-hash to locate overlapped text, measured by jaccard similarity scores
- **Semantical Matching:** Clustering documents with pretrained embeddings



Data Cleaning For LLM

Challenge: How to select high-quality pretraining datasets?

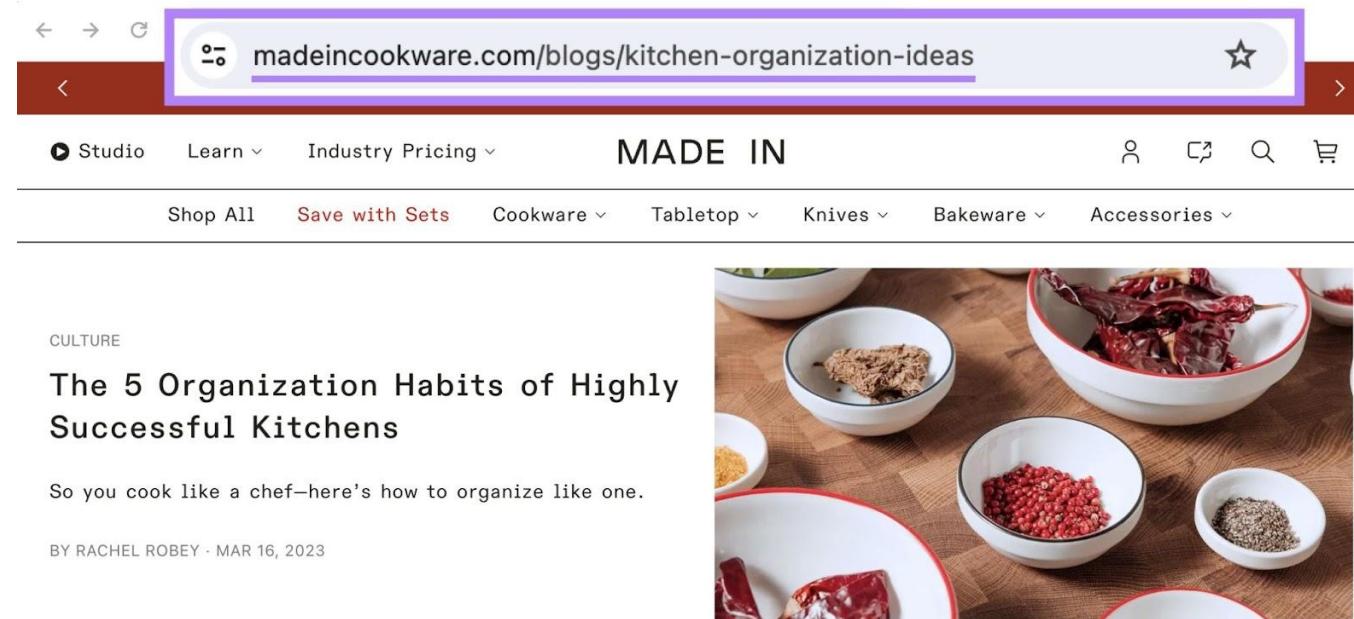
- Rule-based Cleaning: Remove undesirable data with Heuristic Rules



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training performance
- **Exact Matching Techniques:**
 - **1. URL Deduplication:** Remove data that shares the same URL
 - Individual web pages may appear multiple times



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training performance
- **Exact Matching Techniques:**

- **2. Hash Functions:** Guarantee to find all exact matches

(1) Initialize a Set for Hashes

A set ~ The hashes of encountered text entries.

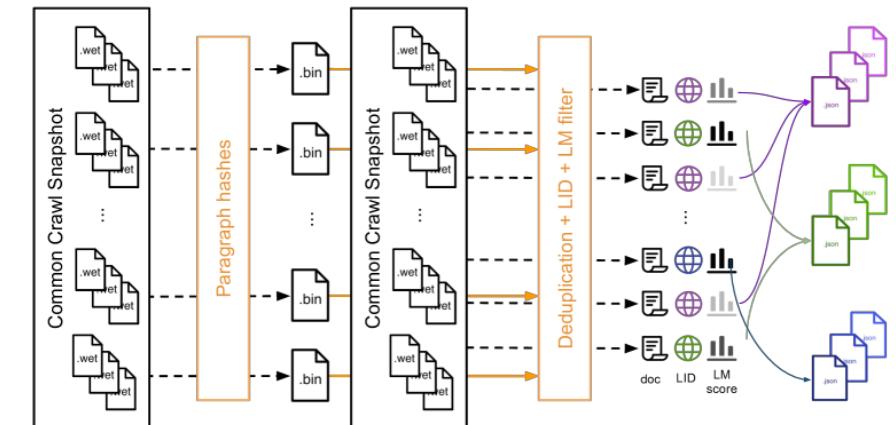
(2) Hash Each Text Entry

For each text entry, compute a simple hash (e.g., the sum of ASCII values of its characters).

(3) Check for Duplicates

If the hash of the current entry is already in the set, it is a duplicate and will be ignored.

If the hash is not in the set, add the hash to the set and keep the entry.



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training performance
- Exact Matching Techniques:

- 2. Hash Functions: Guarantee to find all exact matches

(1) Initialize a Set for Hashes

A set ~ The hashes of encountered text entries.

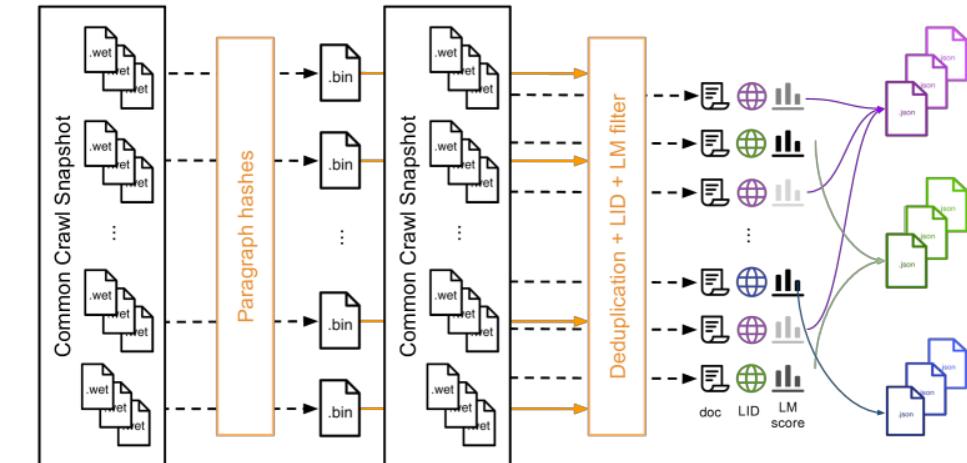
(2) Hash Each Text Entry

For each text entry, compute a simple hash (e.g., the sum of ASCII values of its characters).

(3) Check for Duplicates

If the hash of the current entry is already in the set, it is a duplicate and will be ignored.

If the hash is not in the set, add the hash to the set and keep the entry.

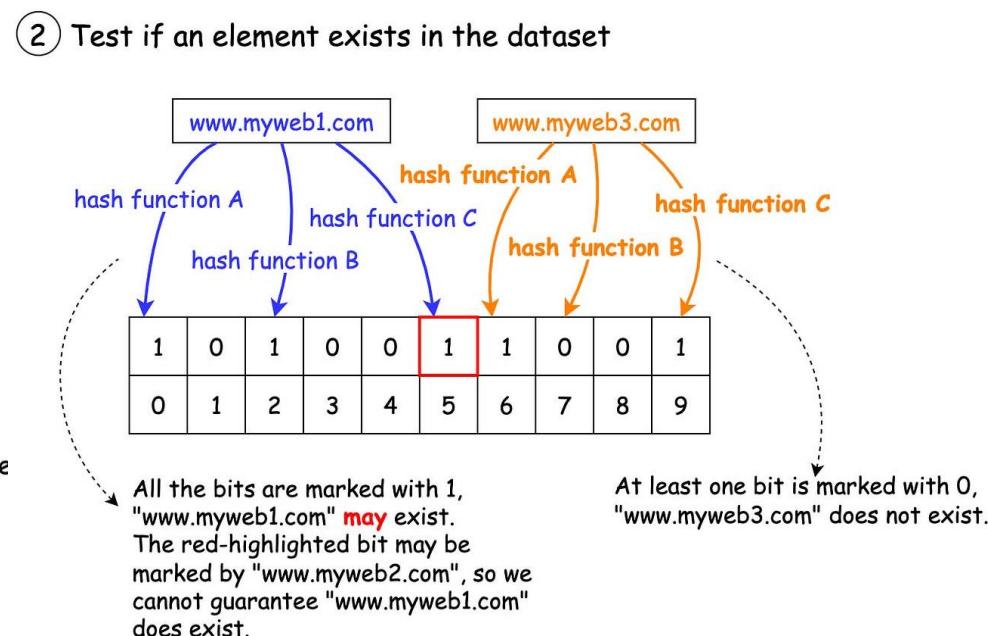
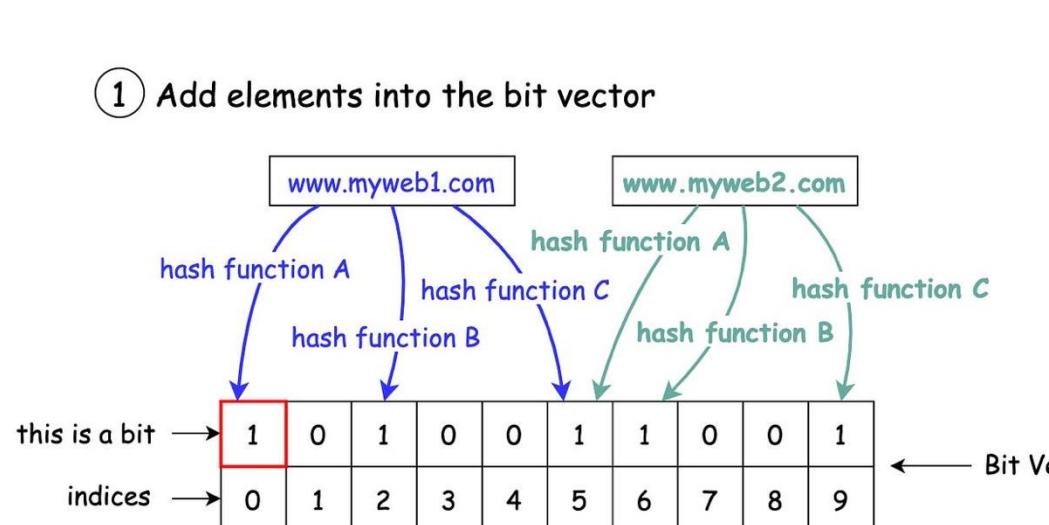


Efficient and Fast, but may find false positives due to hash collisions and remove non-matching documents

Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

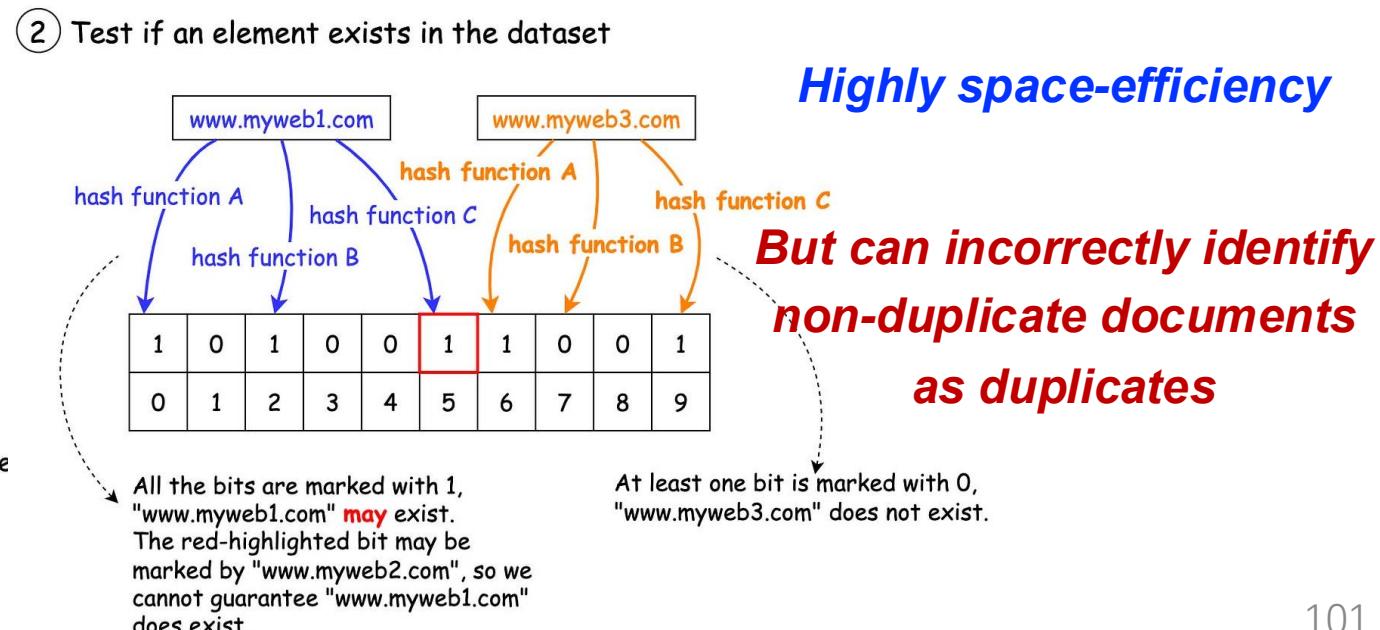
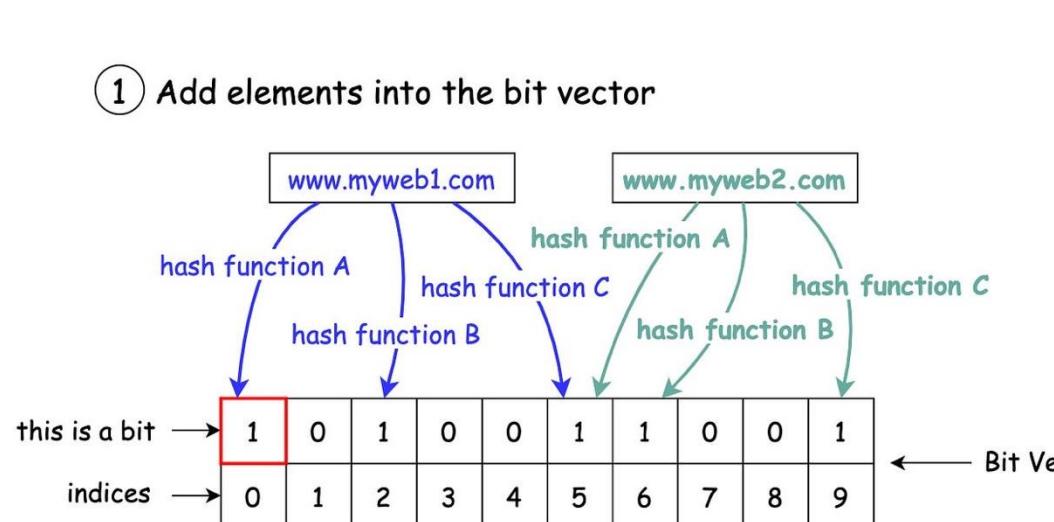
- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Exact Matching Techniques:**
 - **Bloom Filters:** Space-efficient method using bit arrays for document comparison.



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Exact Matching Techniques:**
 - **Bloom Filters:** Space-efficient method using bit arrays for document comparison.



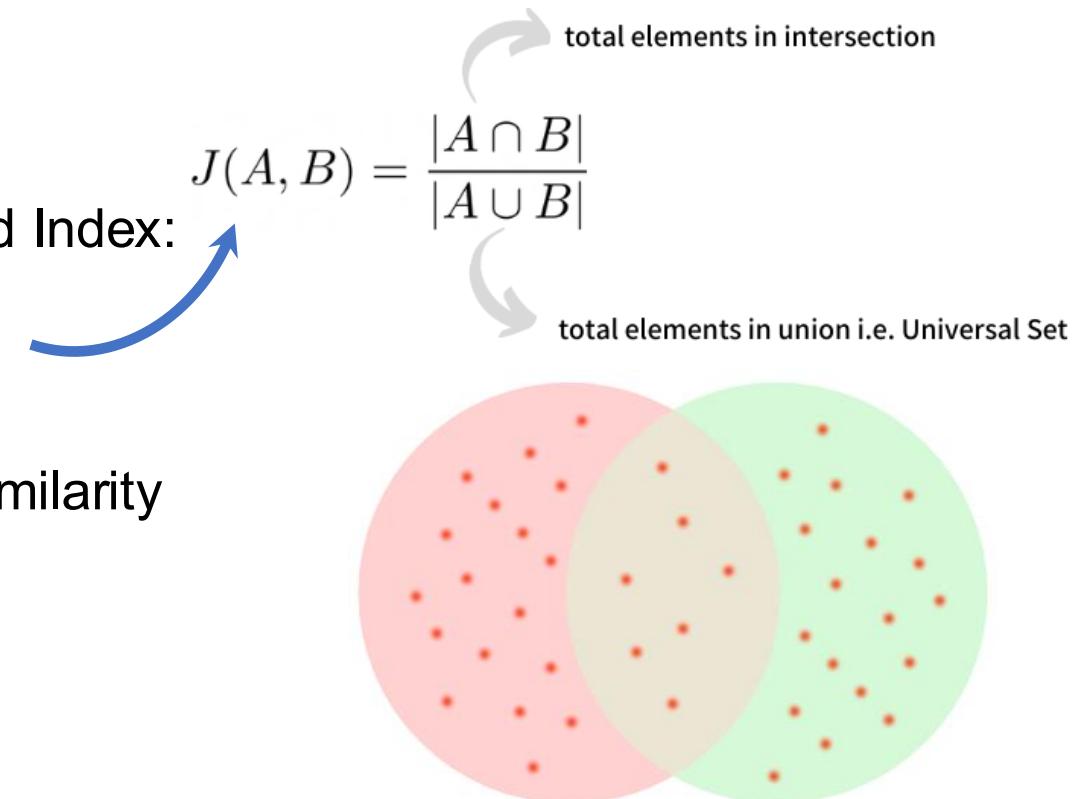
Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Approximate Matching Techniques:**
 - **1. String Metric Method**
 - **S1:** Use MinHash to approximate the Jaccard Index:

$$\text{Jaccard}(d_i, d_j) = |d_i \cap d_j| / |d_i \cup d_j|$$

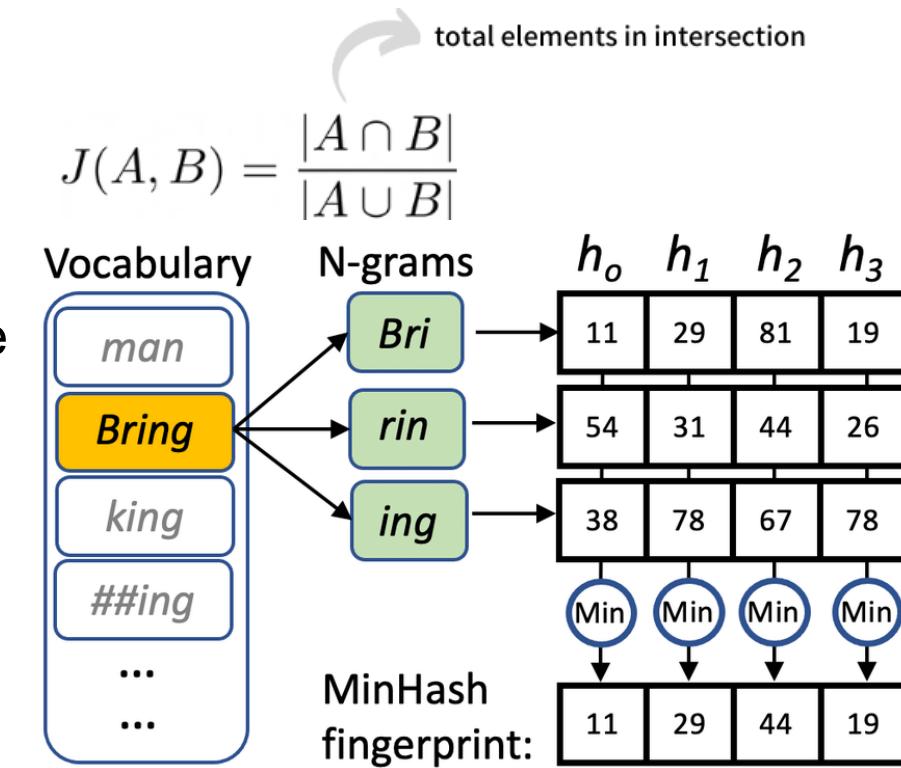
- d_i : The n-grams of document I
- High Jaccard Index indicates high text similarity



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Approximate Matching Techniques:**
 - **1. String Metric Method**
 - **S1:** Use MinHash to approximate the Jaccard Index:
 - **MinHash:** Construct document signatures by sorting each n-gram via a hash function; Then keep only the k smallest hashed n-grams.

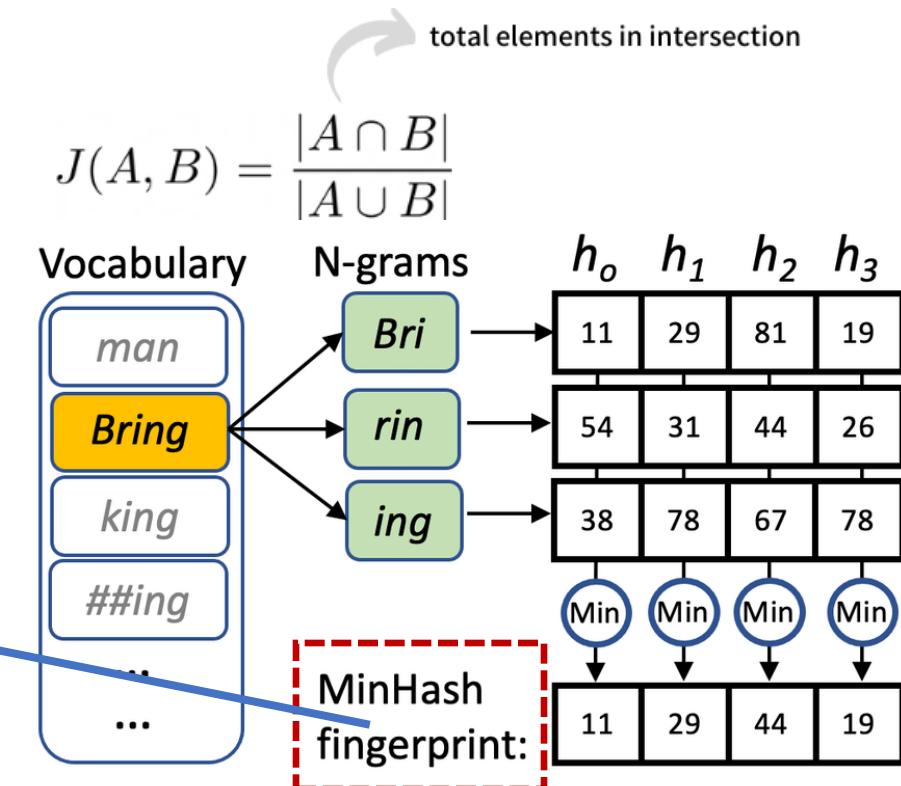


Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Approximate Matching Techniques:**
 - **1. String Metric Method**
 - **S1:** Use MinHash to approximate the Jaccard Index:
 - **MinHash:** Construct document signatures by sorting each n-gram via a hash function; Then keep only the k smallest hashed n-grams.

- These **MinHash fingerprints** are then partitioned into r bucket (with b hashes per bucket).
- In each bucket, the b hashes are augmented into one value.
 - If two documents have the same value in at least one bucket, they'll be marked as a **potential match**.



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Approximate Matching Techniques:**
 - **1. String Metric Method**
 - **S1:** Use MinHash to approximate the Jaccard Index: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
 - **MinHash:** Construct document signatures by sorting each n-gram via a hash function; Then keep only the k smallest hashed n-grams.
 - **S2:** For each “potentially similar” pair, compute edit similarity

$$\text{EditSim}(x_i, x_j) = 1 - \frac{\text{EditDistance}(x_i, x_j)}{\max(|x_i|, |x_j|)}$$

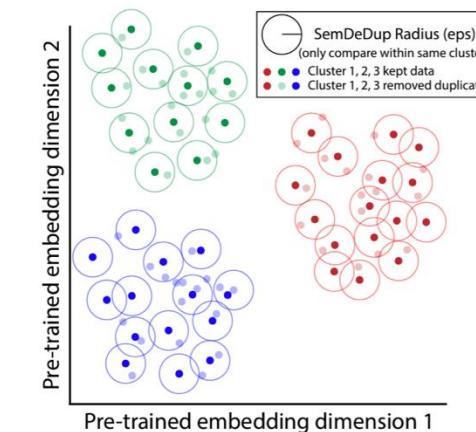
- E.g., two documents are similar if EditSim is greater than 0.8



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

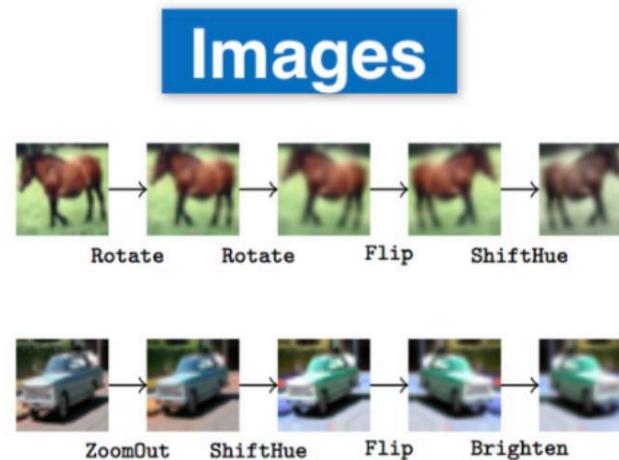
- **Data Deduplication:** Remove duplicates to enhance training or sometimes improve accuracy
- **Approximate Matching Techniques:**
 - **2. Model-based Method:** Use pretrained models for semantic deduplication
 - **S1:** Leverage embedding spaces created by pre-trained LLM, providing a semantically meaningful distance metric for identifying duplicates
 - **S2:** Each data point is embedded using the LLM
 - **S3:** The embedded data points are clustered using k-means
 - **S4:** Within each cluster, pairwise cosine similarities between data points are calculated.
 - **S5:** For identified duplicates within a cluster, only the point with the lowest cosine similarity to the cluster centroid is kept, and the others are removed.



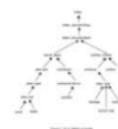
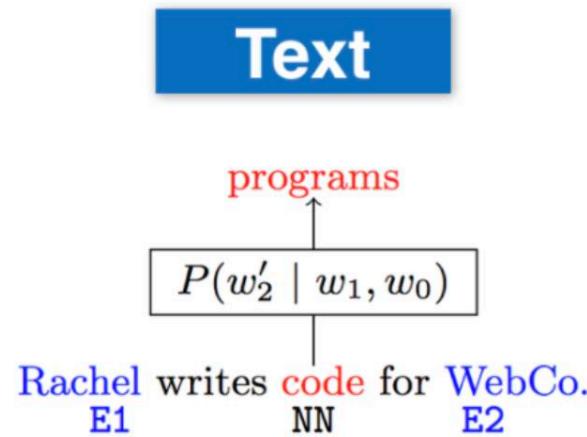
Data Augmentation For LLM

Data Preparation: Turn big dirty data into a subset of good data

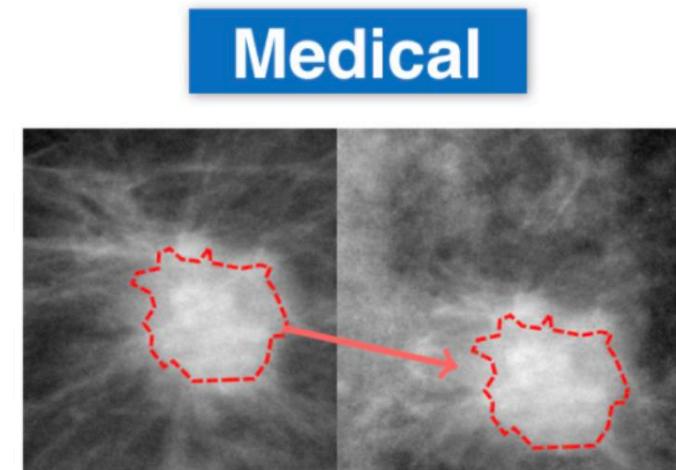
- **Data Augmentation:** Find auxiliary data which most resembles the distribution of desired data distribution (e.g., medicine or law).



- Rotations
- Scaling / Zooms
- Brightness
- Color Shifts
- Etc...



- Synonymy
- Positional Swaps
- Etc...



Domain-specific transformations.
Ex:

1. Segment tumor mass
2. Move
3. Resample background tissue
4. Blend

Data Augmentation For LLM

Challenge: How to select high-quality pretraining datasets?

- **Data Augmentation:** The goal is to find the **auxiliary data** which most resembles the distribution of in-domain data.
- **Domain-Specific Selection:** Let I be in-domain dataset, N be general purpose dataset, N_I be a subset of N that is in-domain that we wish to discover. The probability of “a data point $x(i)$ drawn randomly from N being in N_I ” is:

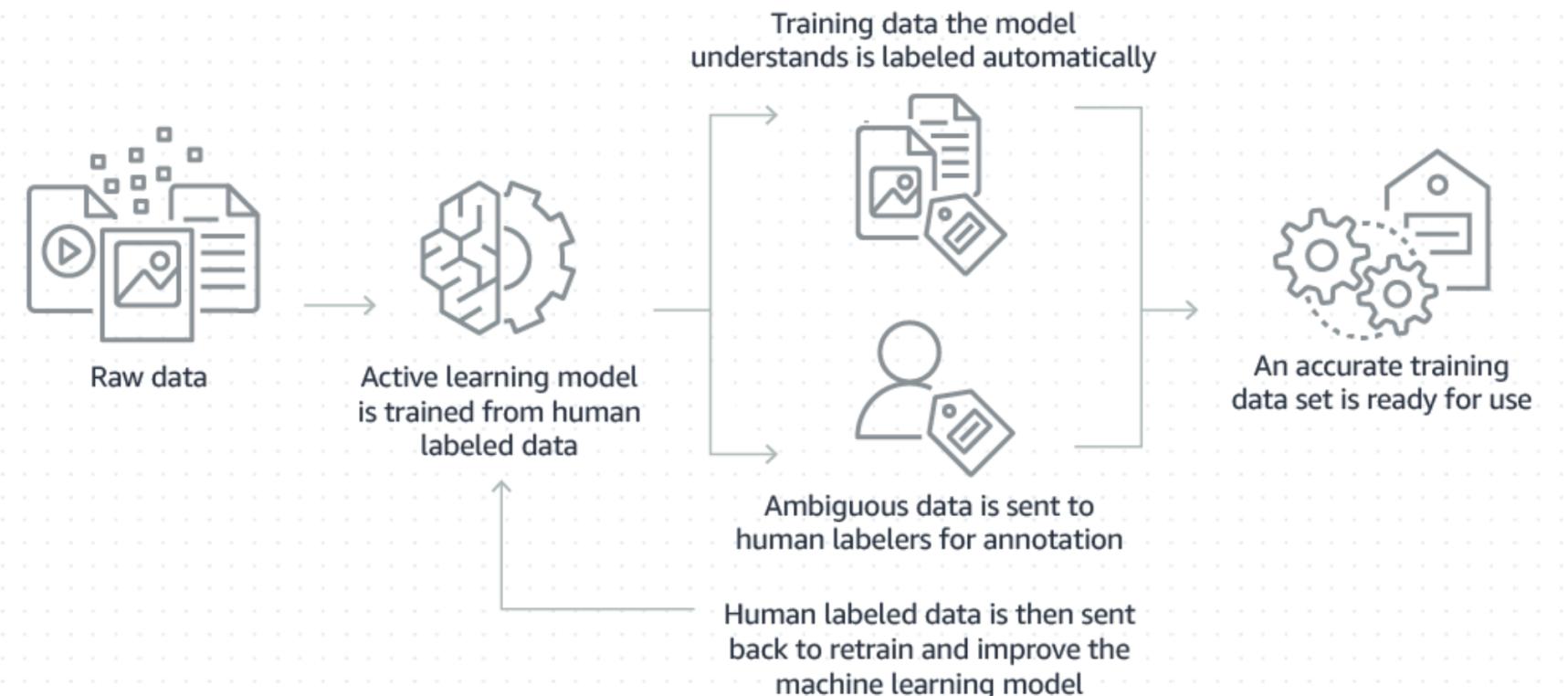
Moore-Lewis selection
$$P(N_I|x^{(i)}, N) = \frac{P(x^{(i)}|I)P(N_I|N)}{P(x^{(i)}|N)}, \quad \frac{P(x^{(i)}|I)}{P(x^{(i)}|N)} \propto \frac{P(x^{(i)}|I)P(N_I|N)}{P(x^{(i)}|N)}$$

- Train models to estimate for $P(x^{(i)}|I)$ and $P(x^{(i)}|N)$ on I and a sample of N
- $\frac{P(x^{(i)}|I)}{P(x^{(i)}|N)}$ is approximated by $\log(P(x^{(i)}|I)) - \log(P(x^{(i)}|N))$ i.e., the cross-entropy loss from models trained on I and N .

Data Labeling For LLM

Data Preparation: Turn big dirty data into a subset of good data

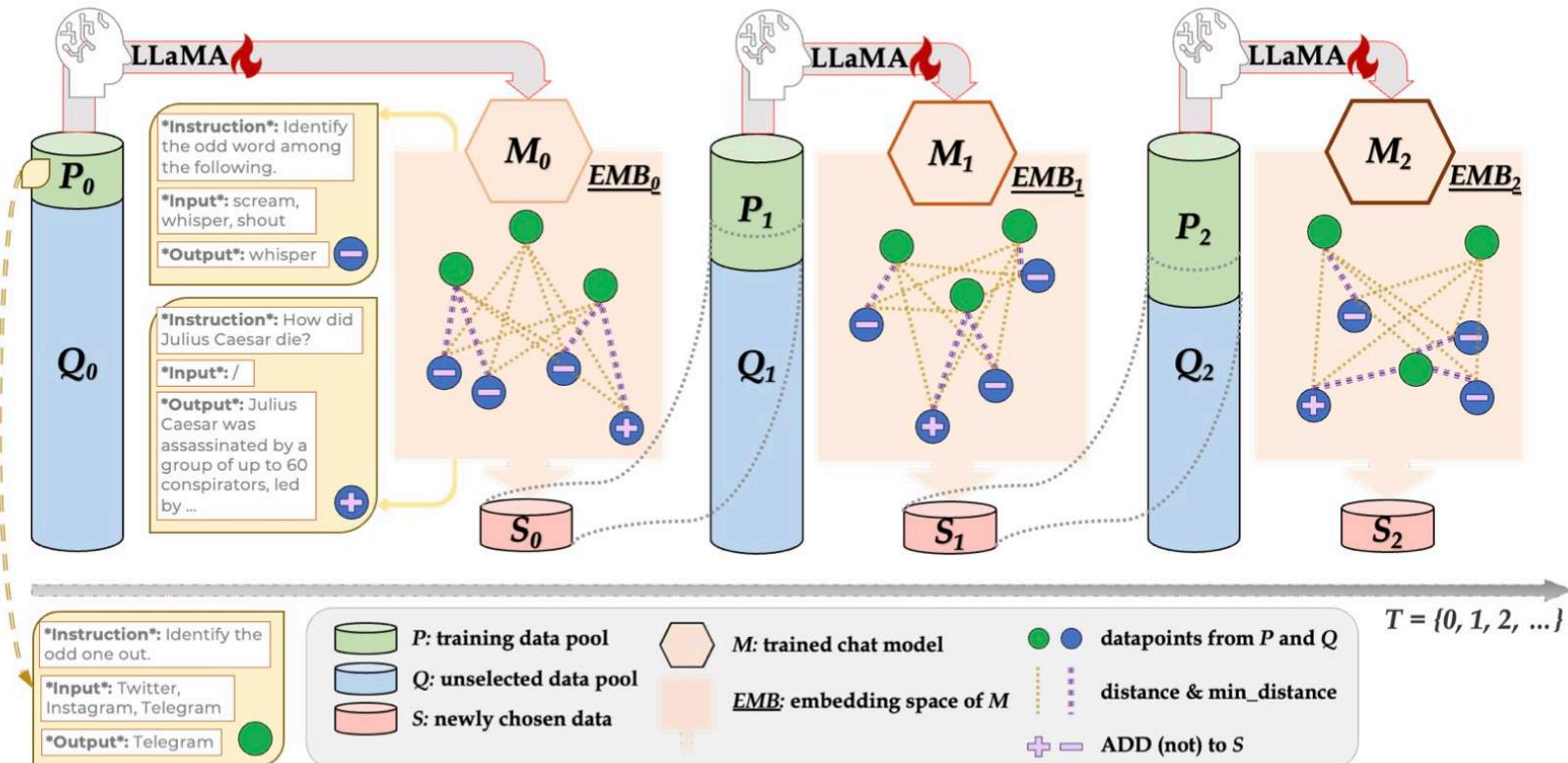
- **Data Labeling:** Annotate or tag raw data (images, texts, videos, etc) with meaningful information to provide context for LLM to learn
- A properly labeled dataset is “**Ground Truth**” in model training and assessing



Distance-based Data Labeling

Motivation: No intervention from humans or more advanced LLMs

- Use the model itself to iteratively guide data selection



Target: Enhance the instruction-following capabilities

Core idea: Select new data points most distinct from existing ones

Data Mixing For LLM

Data Preparation: Turn big dirty data into a subset of good data

- **Data Mixing:** Data mixing optimizes the **weighting of different data domains** in training corpora to enhance model training efficiency and performance.

Component	Raw Size	Weight	Epochs	Effective Size	Mean Document Size
Pile-CC	227.12 GiB	18.11%	1.0	227.12 GiB	4.33 KiB
PubMed Central	90.27 GiB	14.40%	2.0	180.55 GiB	30.55 KiB
Books3 [†]	100.96 GiB	12.07%	1.5	151.44 GiB	538.36 KiB
OpenWebText2	62.77 GiB	10.01%	2.0	125.54 GiB	3.85 KiB
ArXiv	56.21 GiB	8.96%	2.0	112.42 GiB	46.61 KiB
Github	95.16 GiB	7.59%	1.0	95.16 GiB	5.25 KiB
YoutubeSubtitles	3.73 GiB	0.60%	2.0	7.47 GiB	22.55 KiB
PhilPapers	2.38 GiB	0.38%	2.0	4.76 GiB	73.37 KiB
NIH ExPorter	1.89 GiB	0.30%	2.0	3.79 GiB	2.11 KiB
Enron Emails [†]	0.88 GiB	0.14%	2.0	1.76 GiB	1.78 KiB
The Pile	825.18 GiB			1254.20 GiB	5.91 KiB

Table 1: Overview of datasets in the Pile before creating the held out sets. Raw Size is the size before any up- or down-sampling. Weight is the percentage of bytes in the final dataset occupied by each dataset. Epochs is the number of passes over each constituent dataset during a full epoch over the Pile. Effective Size is the approximate number of bytes in the Pile occupied by each dataset. Datasets marked with a [†] are used with minimal preprocessing from prior work.

Data Mixing For LLM

Challenge: How to select high-quality pretraining datasets?

- **Data Mixing:** Determine the optimal domain ratios to improve the training efficiency and model performance
- **Empirical-Determined Method**
 - **Rule 1:** Prevent small sources (e.g., MultiUN) from oversampled;
 - **Rule 2:** Large proportion of code (e.g., 50%) does not harm to NL performance, and can benefit reasoning-based tasks;
 - **Rule 3:** Test different combinations over small-sized LLMs like 1B parameters.

Github	Microsoft	2008-4	-	All
mC4	Google Research	2021-6	251 GB	All
MNBVC	Liwu Community	2023-1	20811 GB	All
MTP	BAAI	2023-9	1.3 TB	All
MultiUN	German Research Center for Artificial Intelligence (DFKI) GmbH	2010-5	4353 MB	All
News-crawl	UKRI et al.	2019-1	110 GB	All

Data Mixing For LLM

Challenge: How to select high-quality pretraining datasets?

- **Data Mixing:** Determine the optimal domain ratios to improve the training efficiency and model performance
- **Model-Determined Method:** Optimize the ratios assigned to different domains in training a model without relying on downstream tasks
 - Optimize domain ratios using a small proxy model

$$\min_{\theta} \max_{g \in \mathcal{G}} \mathbb{E}_{(x,y) \sim \mathcal{D}_g} [\ell(f_{\theta}(x), y)]$$

- θ : Model parameters
- g : Group/domain
- \mathcal{D}_g : Data distribution for group g
- ℓ : Loss function

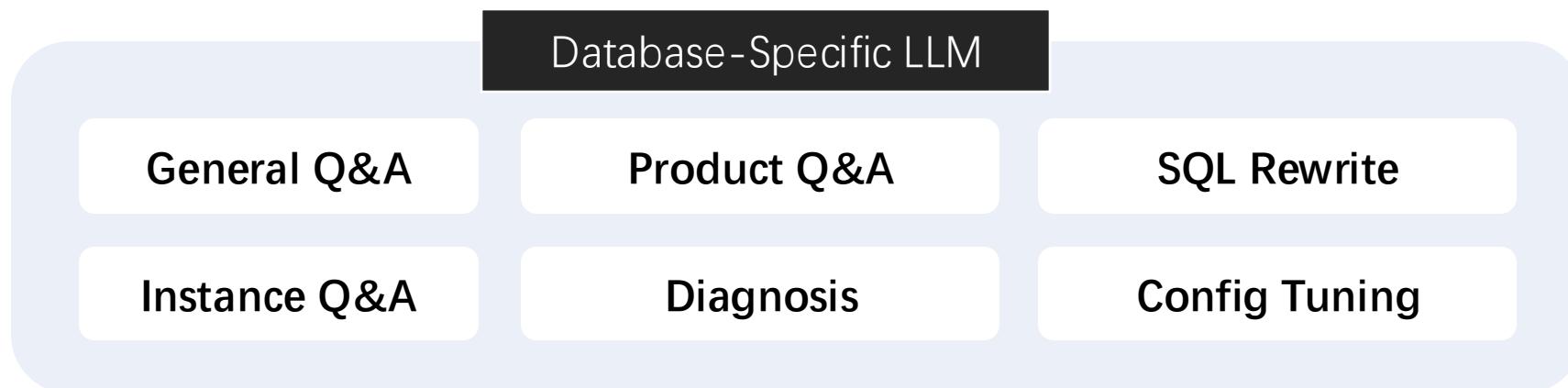
Minimize the maximum loss across all domains

- Train a larger model using the optimized domain ratios

Open Problem 1: Database Specific LLM

□ Case-by-Case LLM Finetuning → Database-Specific LLM Building

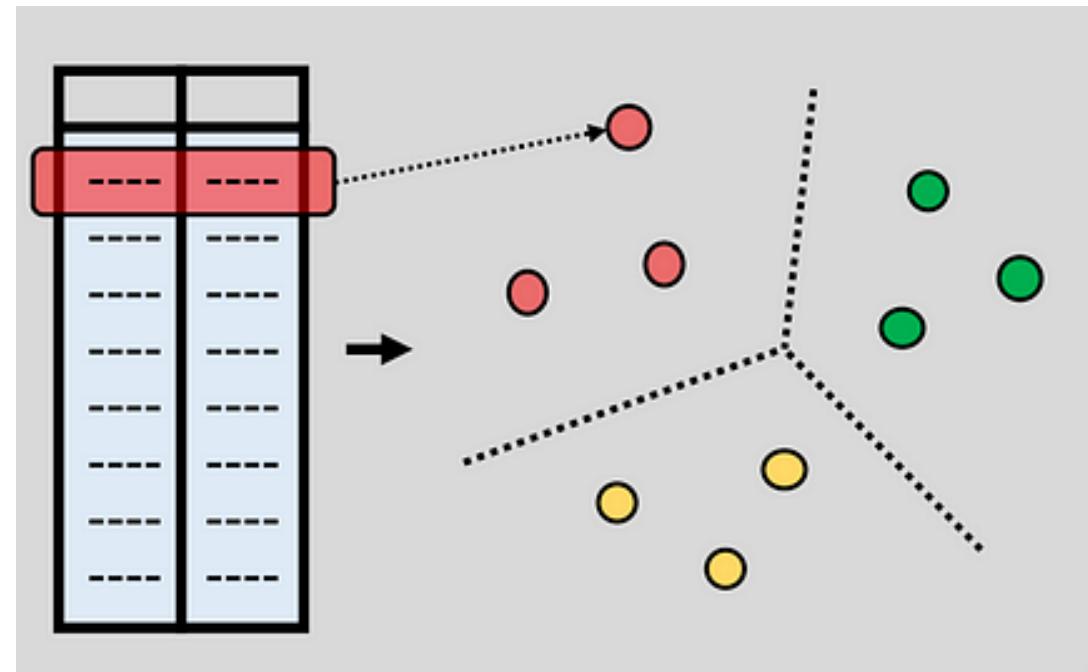
- **Pretrain:** Collect sufficient database-domain tokens (e.g., in millions) as pre-training corpora from sources like database textbook and query analysis
- **Finetune:** SQL / Data Understanding → DB / Product / Instance Q&A → Database-Relevant Task-Solving → Database Expert Alignment
- **Evaluation:** Evaluate the accuracy and robustness of the database model with carefully-crafted dataset, metrics, and testbed.



Open Problem 2: Tabular Data Learning

□ Sequential Textual Data → Structured Tables + Textual Task

- Table Position Encoding
- Table Content Encoding
- Table-Text Alignment
- Excessively Large Tables
- ...



Open Problem 3: Lightweight LLM Usage

□ Reduce the implementation costs of LLM in DB tasks

- **Training:** Post-Pretraining of swallow layers
- **Prompt:** Prompt Template Recommendation
- **Tool:** Automatic API Integration
- **RAG:** Knowledge Graph + Vector Database
- **Cloud:** Agent-As-A-Service
- ...

Open Problem 4: Data Preparation System for LLM

□ Reduce the implementation costs of LLM in DB tasks

- **Training:** Post-Pretraining of swallow layers
- **Prompt:** Prompt Template Recommendation
- **Tool:** Automatic API Integration
- **RAG:** Knowledge Graph + Vector Database
- **Cloud:** Agent-As-A-Service
- ...