香港科技大学(广州)
THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search

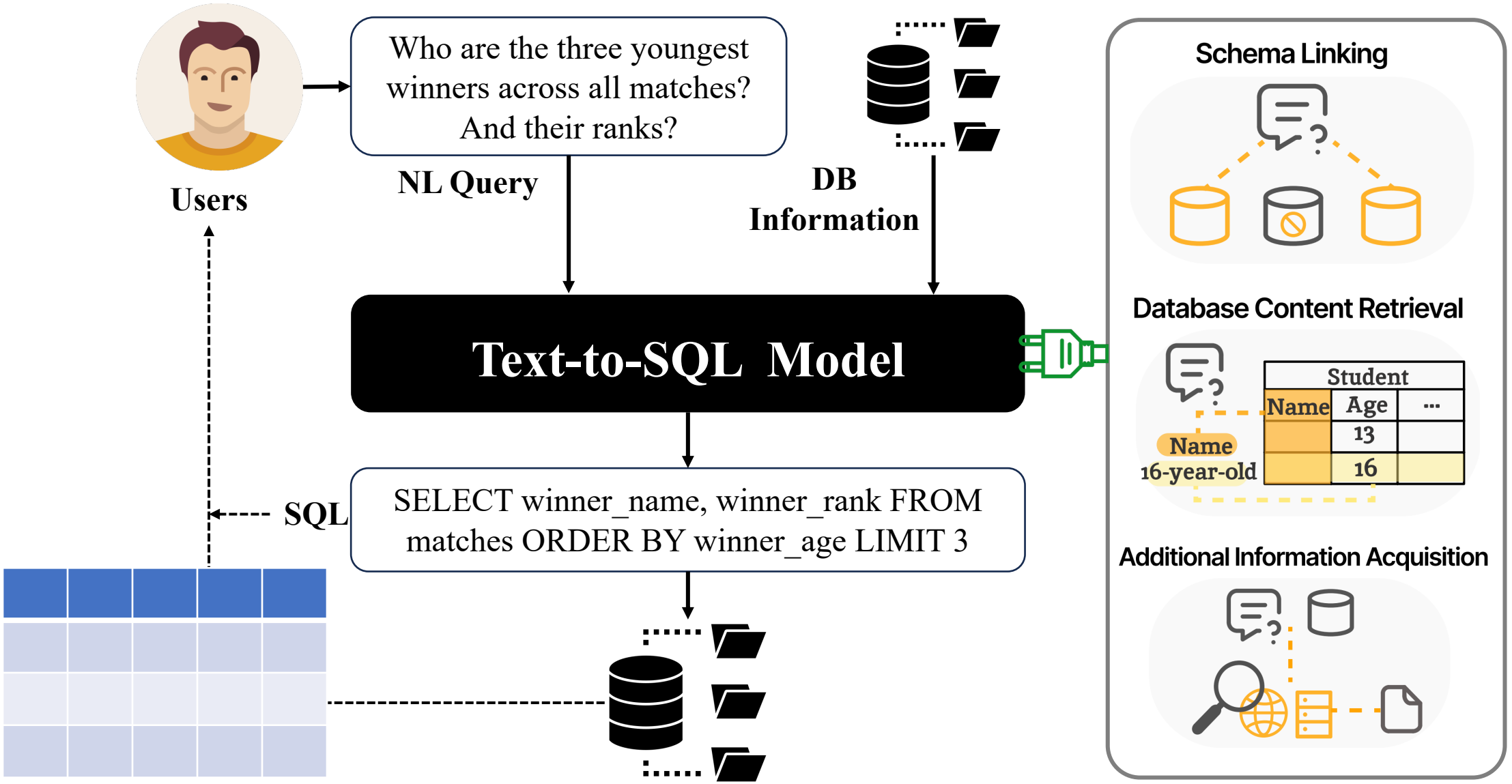
Yuyu Luo

The Hong Kong University of Science and Technology (Guangzhou)

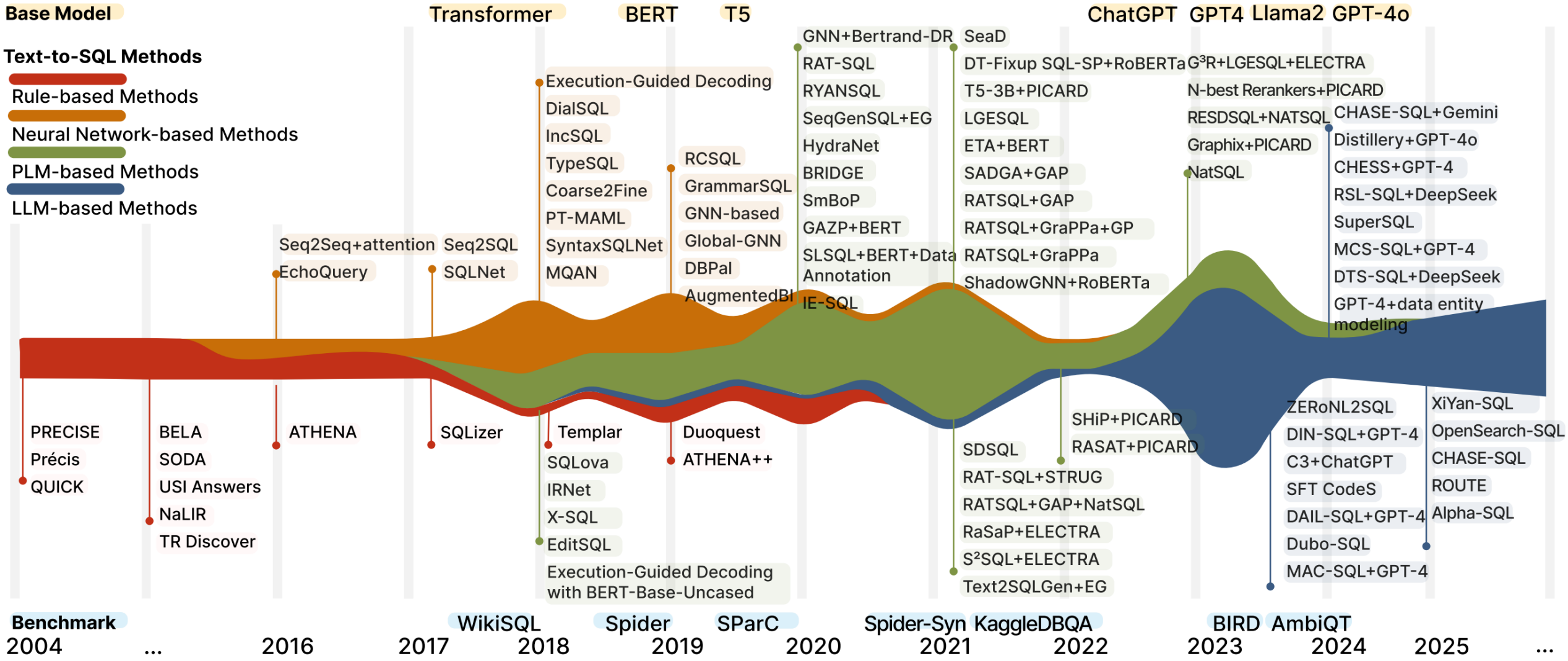
https://luoyuyu.vip/

yuyuluo@hkust-gz.edu.cn

Text-to-SQL: Bridges Humans and Databases



Where Are We?

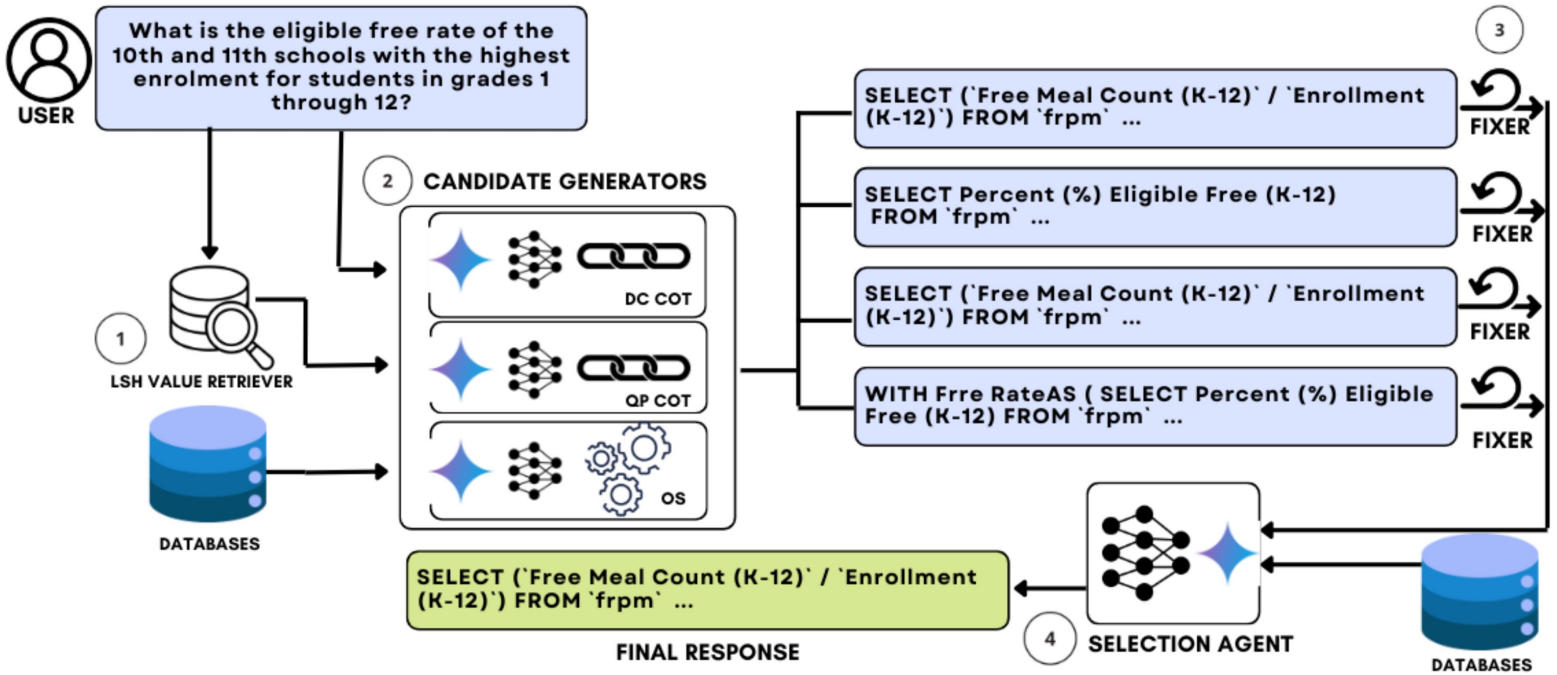


https://github.com/HKUSTDial/NL2SQL_Handbook

Where Are We?

Closed-source LLMs

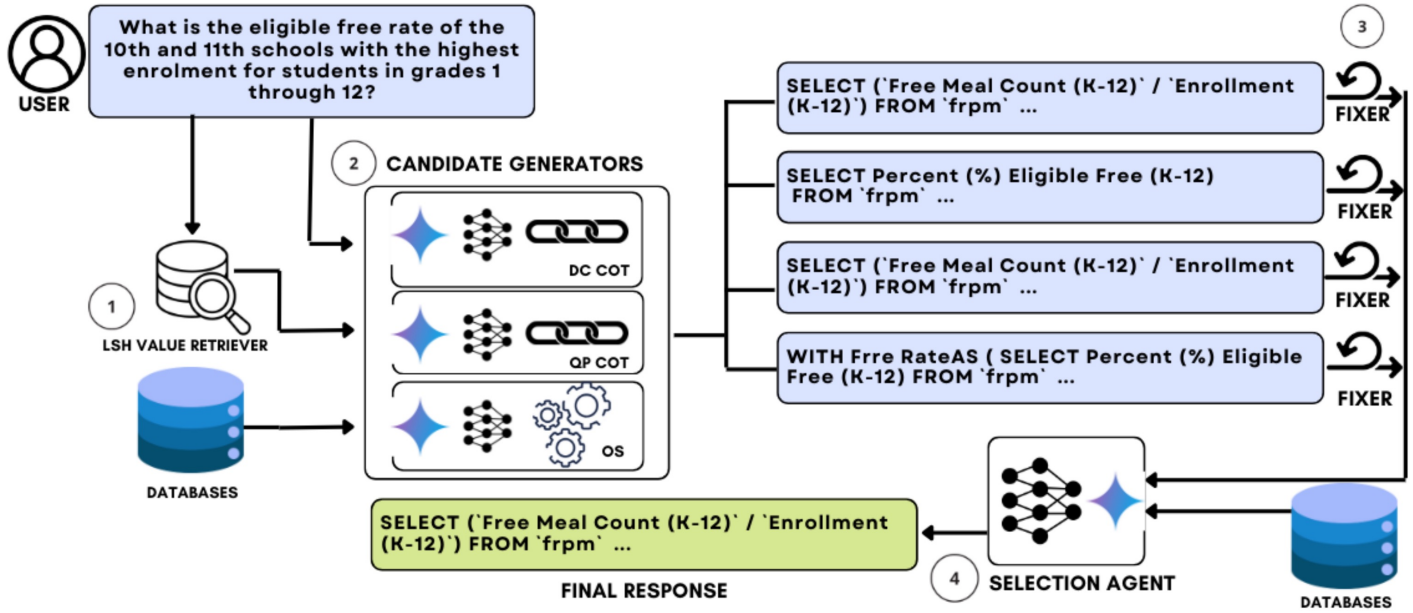
- **CHASE-SQL** [ICLR 2025], by Google Cloud and Stanford
  - Utilizes the **MinHash LSH** to search for values related to the user query
  - Multiple prompting strategies to **generate various candidate SQL queries** using LLMs, and corrects SQL queries with execution errors through prompting LLMs.
  - Employs an **SQL selection agent** fine-tuned specifically for the database to select the final SQL from multiple candidates.



Where Are We?

Closed-source LLMs

- **CHASE-SQL** [ICLR 2025], by Google Cloud and Stanford



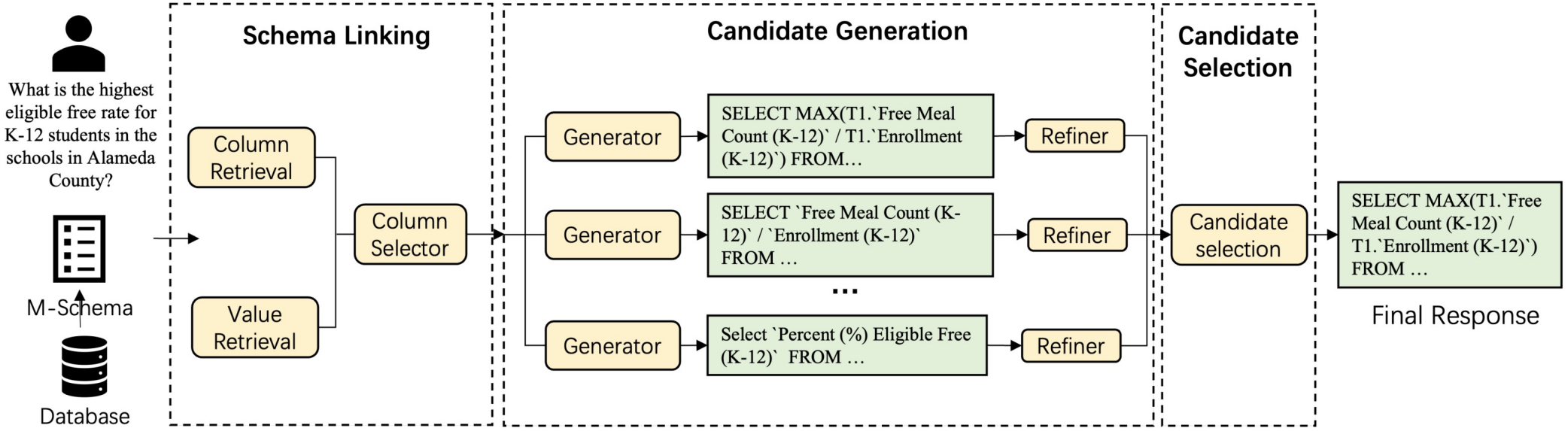
Key Limitations:

- **Reliance on closed-source large models**
  - High cost (**0.6 USD/query**), making it difficult to widely deploy in real-world industrial scenarios.
- **SQL selection agent requires fine-tuning**
  - The *Google* team *fine-tuned* the *Gemini-1.5-Flash* model specifically.
  - Limited flexibility due to reliance on domain-specific data.
- **Predefined and Fixed Reasoning Workflows**

Where Are We?

Open-source LLMs

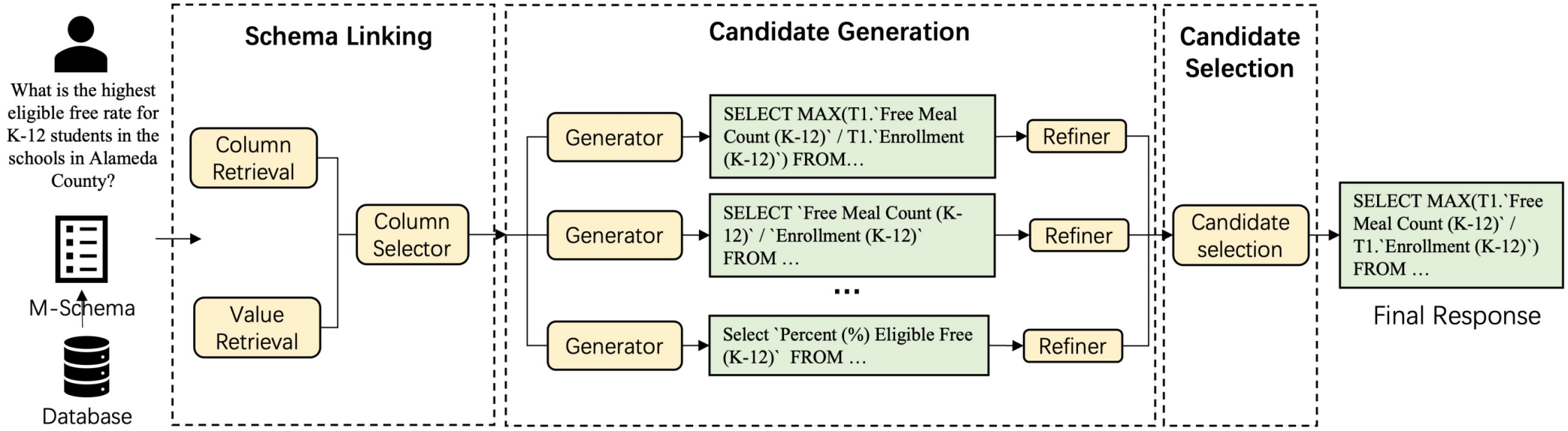
- **XiYan-SQL**, by Alibaba
  - **M-Schema**: Uses column and value retrieval to select relevant schema items from DBs.
  - **Fine-tunes a base LLM** on SQL-specific data, then **creates multiple specialized SQL-generation** models by fine-tuning with diverse Text-to-SQL syntax datasets.
  - Employs a **fine-tuned SQL selection** model to choose the best SQL from predictions made by multiple generators.



Where Are We?

Open-source LLMs

- **XiYan-SQL**, by Alibaba



Key Limitations:

- High dependency on extensive **domain-specific data**.
- Significant **costs** associated with **fine-tuning** multiple models.
- **Difficulty** in rapid **adaptation** and **generalization** across varied scenarios.
- **Predefined and Fixed Reasoning Workflows**.

Key Takeaways

Closed-source LLMs for Text-to-SQL:

- High inference API cost limits practical deployments.
- Potential data privacy concerns for sensitive applications.

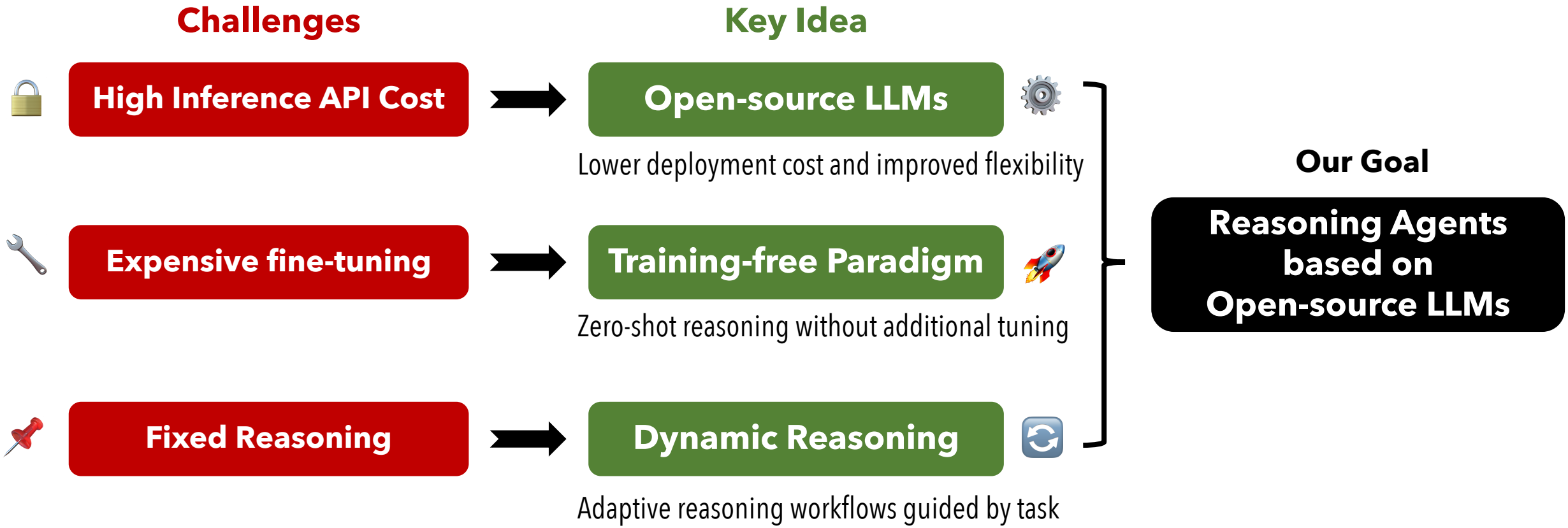
Open-source LLMs for Text-to-SQL:

- Dependence on extensive domain-specific data for model fine-tuning.
- Limited generalization capability across different use cases.

Common Limitations in Existing Solutions:

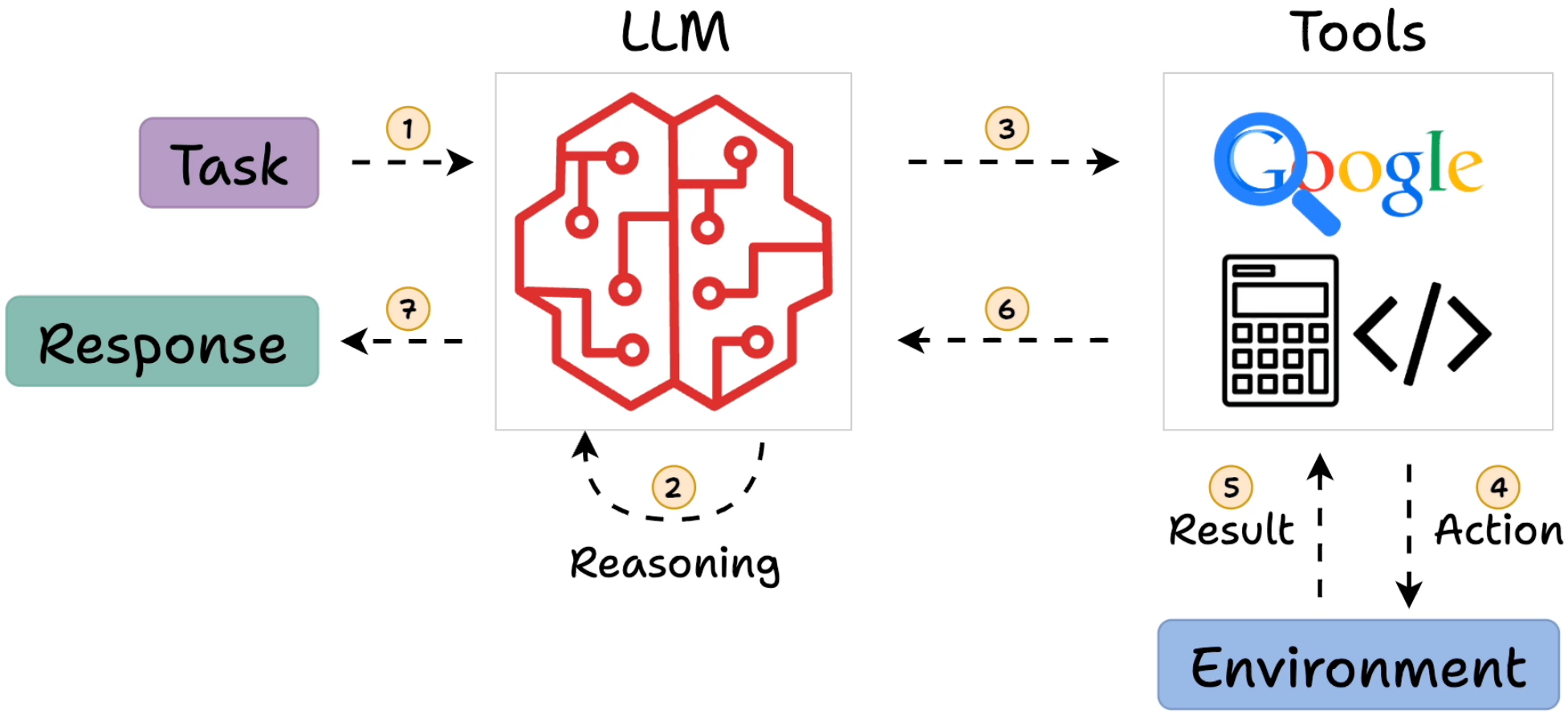
- Predefined and fixed reasoning workflows restrict adaptability.
- Domain adaptation and generalization across DB and text queries

Where Are We Going?



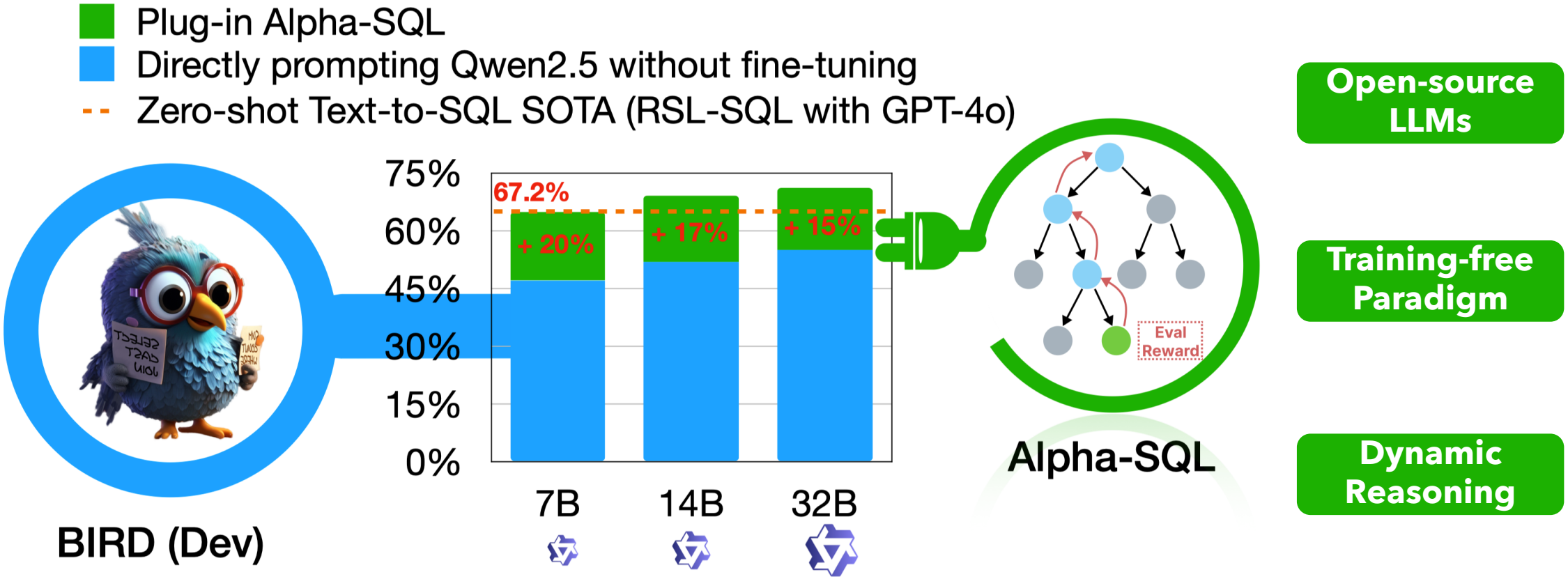
We need a new perspective that reduces deployment cost, improves flexibility, and introduces a more adaptive reasoning mechanism

What is the (Reasoning) Agent?



https://blog.dailydoseofds.com/p/intro-to-react-reasoning-and-action

Alpha-SQL:
A Plug-and-Play Text-to-SQL Reasoning Framework



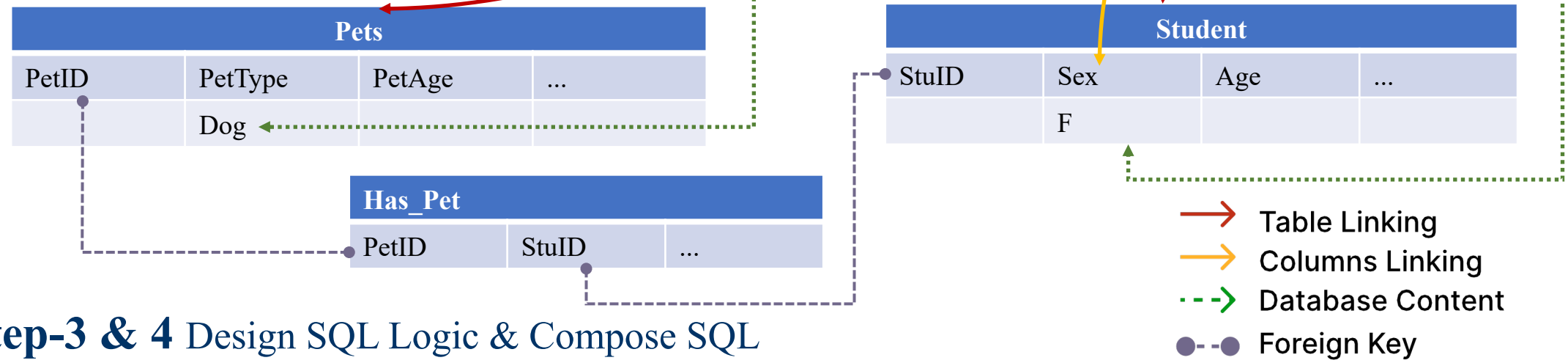
Yuyu Luo, Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search, ICML 2025.
https://alpha-sql-hkust.github.io/

NL2SQL Human Workflow

Step-1 NL Understanding

Find the number of dog pets that are raised by female student

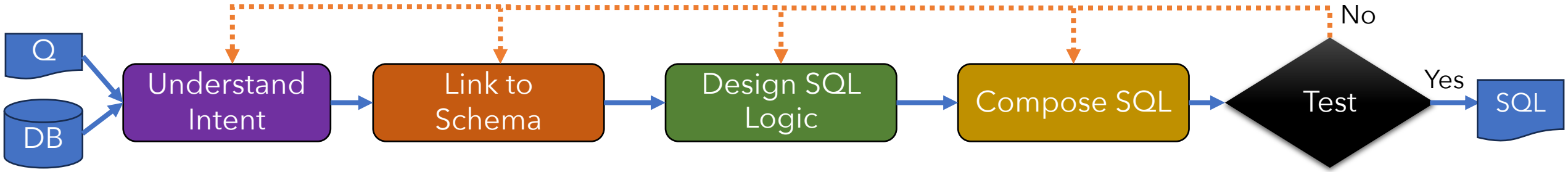
Step-2 Schema Linking and Database Content Retrieval



Step-3 & 4 Design SQL Logic & Compose SQL

Select count(*) *FROM* student *AS* T1 *JOIN* has_pet *AS* T2 *ON* T1.stuid=T2.stuid
JOIN pets *AS* T3 *ON* T2.petid=T3.petid *WHERE* T1.sex='F' *AND* T3.pettype='Dog'

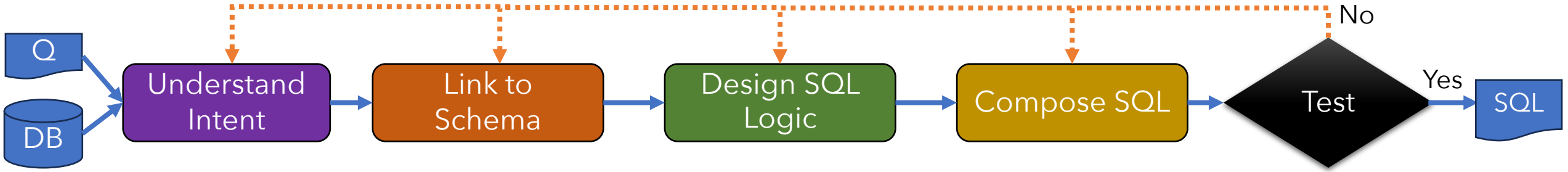
Task Formulation: Mimic Human Experts



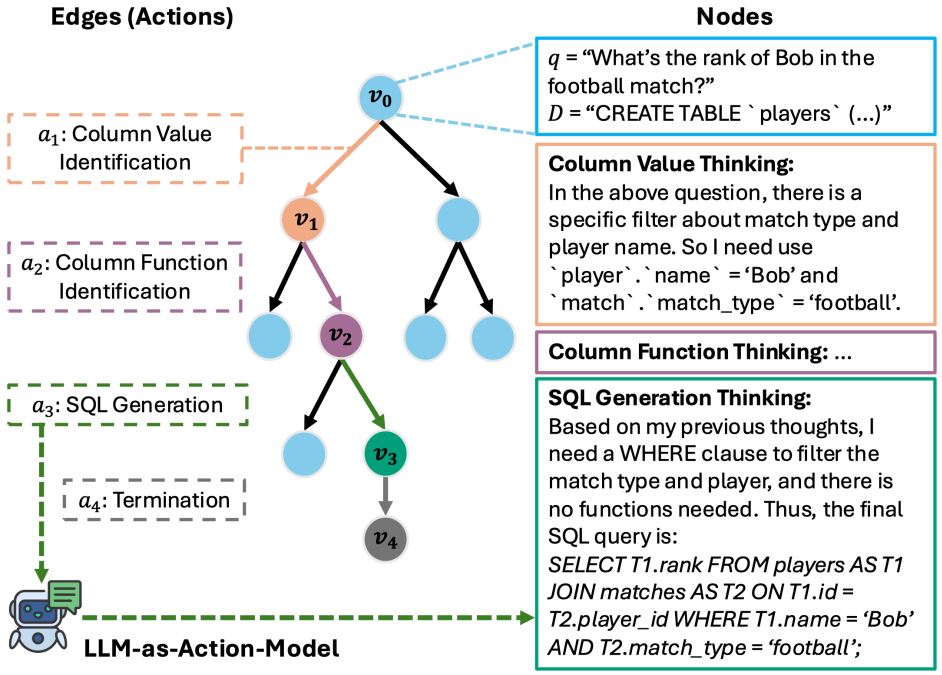
- From ***Human Actions*** to ***Agent Actions***

Understand Intent	**Question Rephrasing**	(Revise, clarify ambiguities, rephrasing)
Link to Schema	**Schema Selection**	(decides which tables / columns / values)
	Cell Value Selection	
Design SQL Logic	**Column Function**	(joins, aggregations, functions)
Compose SQL	**SQL Generation**	(assemble an initial executable query)
Validate & Iter.	**SQL Revision**	(iteratively test, debug, and optimize the query)

Task Formulation: Mimic Human Experts



- From the ***Fixed*** Action to ***Dynamic*** Actions



Tree-based Search:

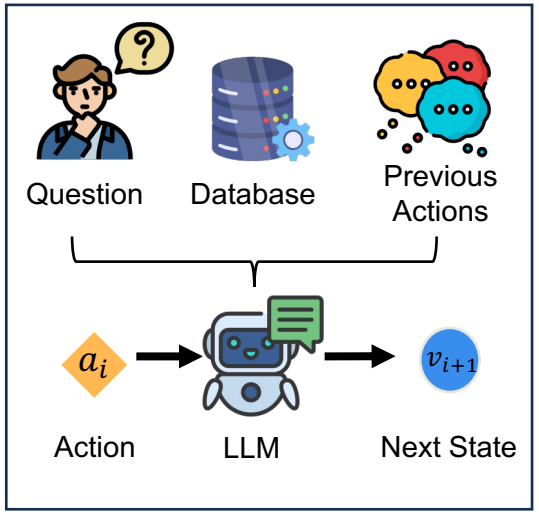
- Each ***edge*** corresponds to an ***agentic action*** in the query construction process,
- Each ***node*** represents a ***reasoning state*** at a specific step, and
- Each ***path*** corresponds ***to a sequence of SQL construction actions*** for Text-to-SQL task.

Text-to-SQL as a Tree-based Search Problem

Action Space

- a_1 Rephrase Question
- a_2 Schema Selection
- a_3 Column Value Identification
- a_4 Column Function Identification
- a_5 SQL Generation
- a_6 SQL Revision
- a_7 Termination

LLM-as-Action-Model



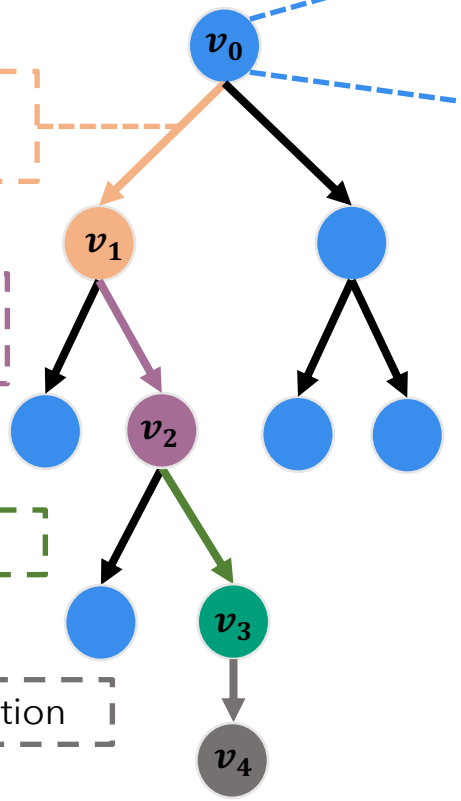
Edges (Actions)

a_3: Column Value Identification

a_4: Column Function Identification

a_5: SQL Generation

a_7: Termination



LLM-as-Action-Model

Nodes (Reasoning States)

q = "What's the rank of Bob in the football match?"
D = "CREATE TABLE `players` (…)"

Question Database

Input

Column Value Thinking:
In the above question, there is a specific filter about match type and player name. So I need use `player`.`name` = 'Bob' and `match`.`match_type` = 'football'.

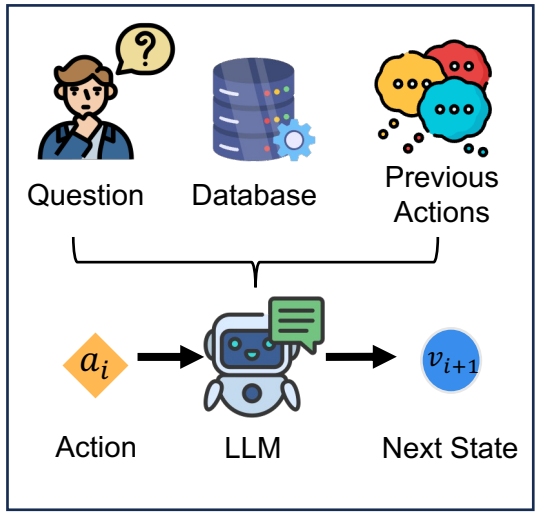
Column Function Thinking: …

SQL Generation Thinking:
Based on my previous thoughts, I need a WHERE clause to filter the match type and player, and there is no functions needed. Thus, the final SQL query is:
SELECT T1.rank FROM players AS T1 JOIN matches AS T2 ON T1.id = T2.player_id WHERE T1.name = 'Bob' AND T2.match_type = 'football';

Output

LLM-as-Action-Model

$$v_{i+1} = LLM(q, \mathcal{D}, Actions(v_0, \cdots, v_i), Prompt(a_i)),$$



Column Function Identification Action Prompt

You are an AI assistant to help me identify the potential column functions (if needed to be used in the SQL query) that are essential for answering the question.
Here is an example:
Database Schema:
CREATE TABLE businesses
(
'business_id' INTEGER NOT NULL,
'name' TEXT NOT NULL, – Column Description: the name of the eatery
PRIMARY KEY ('business_id')
);
CREATE TABLE inspections
(
'business_id' INTEGER NOT NULL, – Column Description: the unique id of the business
'score' INTEGER DEFAULT NULL, – Column Description: the inspection score
'date' TEXT NOT NULL, – Value Examples: '2014-01-24'
FOREIGN KEY ('business_id') REFERENCES 'businesses' ('business_id')
);
Question: What are the names of the businesses that passed with conditions in May 2012?
Hint: name of business refers to dba_name; passed with conditions refers to results = 'Pass w/ Conditions'; in May 2012 refers to inspection_date like '2012-05%'
Answer: Since the businesses passed with conditions in May 2012, I should consider a date-related function to filter the 'inspections'.'date' column. I find that column is of type TEXT, so I can use the strftime('%Y-%m', 'inspections'.'date') = '2012-05' to filter the date.
****
Now, answer the real question, and you need to follow the answer style of the above examples (answer in two sentences).
Database Schema: {SCHEMA_CONTEXT}
Question: {QUESTION}
Hint: {HINT}
Answer:

Text-to-SQL as a Tree-based Search Problem

- **Q1**: How to select the next action (edge)?
- **Q2**: How to effectively navigate the vast search space?
- **Q3**: How to evaluate the quality of the candidate SQL queries?

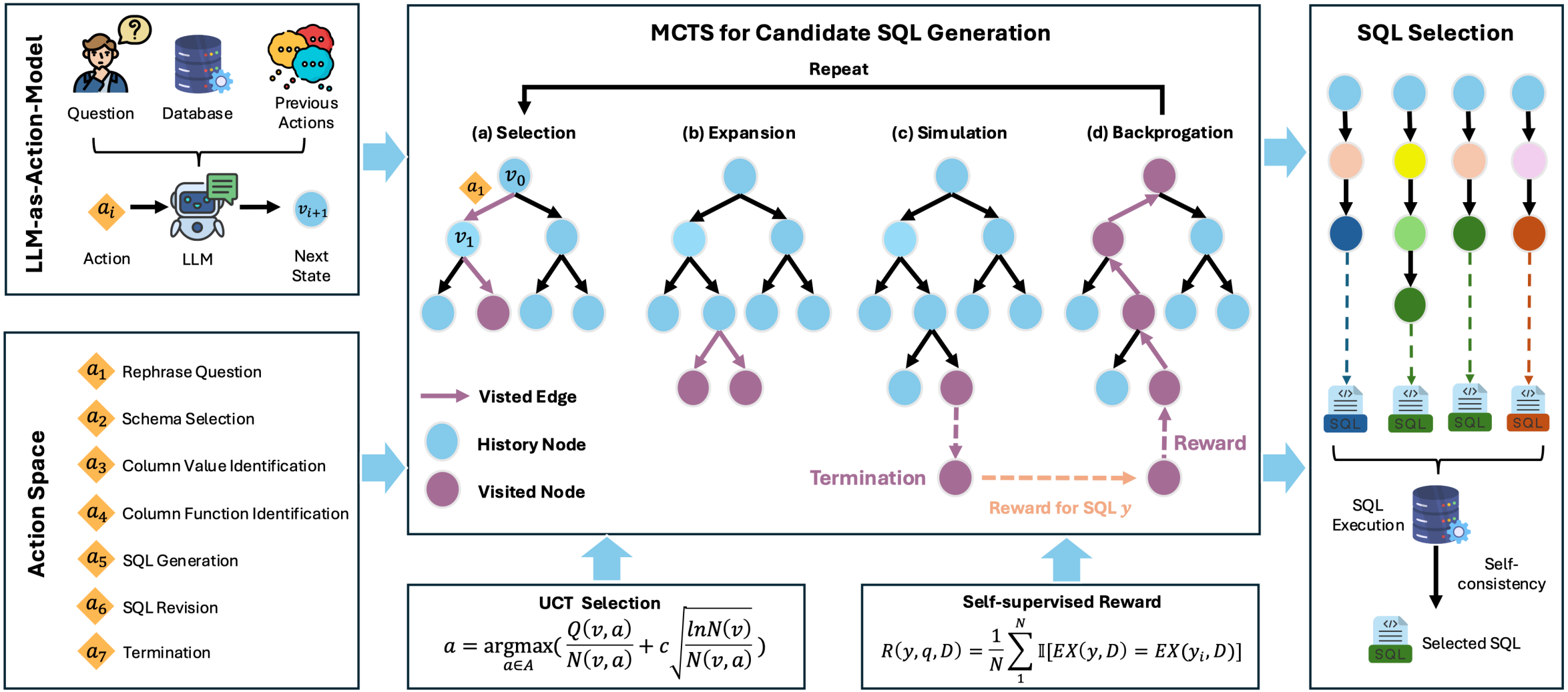
Q1 & Q2

- Monte Carlo Tree Search (MCTS) addresses this by balancing *exploration* (testing *uncertain* actions) and *exploitation* (choosing actions *likely to yield good* results)

Q3

- We need a *self-supervised* reward function since our goal is to avoid reliance on labeled data
- Resampling the LLMs M times to compute the self-consistent scores

Alpha-SQL Solution Overview



Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search, ICML 2025.
https://alpha-sql-hkust.github.io/

Pruning the Candidate Actions (for efficiency)

Action Ordering and Constraints

- Each reasoning trajectory follows a structured, ordered sequence to ensure **logical consistency**.
- Certain actions (e.g., **SQL Revision**) can only be performed after specific preceding actions.
- Each action can occur **only once** within a single reasoning path, preventing **infinite loops**.

Table 1. Action Space with Ordering.

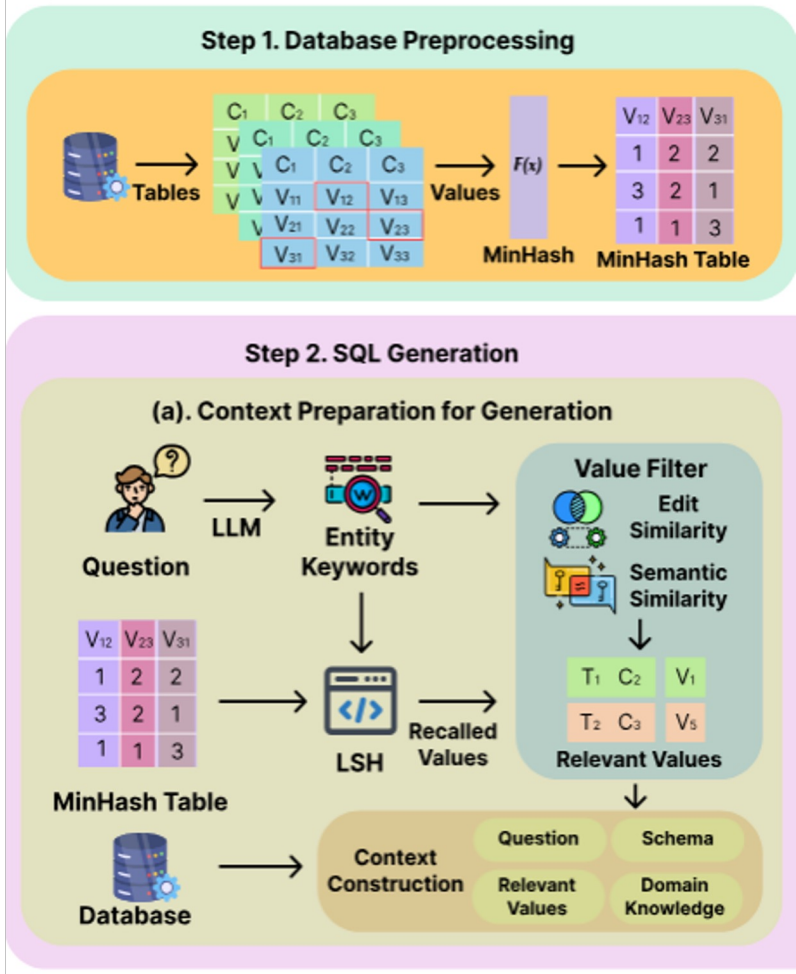
Previous Action	Valid Next Actions
–	A_1, A_2, A_3, A_4, A_5
A_1: Question Rephrasing	A_2, A_3, A_4, A_5
A_2: Schema Selection	A_3, A_4, A_5
A_3: Column Value Identification	A_2, A_4, A_5
A_4: Column Function Identification	A_2, A_3, A_5
A_5: SQL Generation	A_6, A_7
A_6: SQL Revision	A_7
A_7: Termination	–

Pruning the Search Paths (for efficiency)

- Alpha-SQL incorporates **schema constraints** and **semantic rules** into the search process to prune invalid paths early.
- A key aspect of our pruning strategy is the elimination of redundant nodes. For example, when performing a Schema Selection action, we may sample the LLM *M* multiple times (e.g., 3 times). Although the Chain-of-Thought content generated by *M* may differ in each sample, if the final selected schema subset is identical, we create only one child node instead of three duplicate nodes. This de-duplication significantly reduces the branching factor of the search tree without loss of information.

Offline: Database Value Retrieval

- The databases value are extracted and processed offline.
- First, we extract keywords from questions using few-shot LLM prompts.
- We then use LSH to retrieve relevant values, filtering them based on editing similarity and semantic similarity thresholds (ϵ_{edit} , $\epsilon_{semantic}$).
- The semantic matching employs OpenAI's text-embedding-3-large model. The retrieved values will be used as part of the database schema prompt for our *LLM-as-Action-Model* module.



Alpha-SQL: Effectiveness

Table 2. Execution Accuracy on BIRD Development Dataset.

Method	Inference Model	Selection Model	Zero-shot Setting	Accuracy (%)			
				Simple	Moderate	Challenging	All
SFT CodeS (Li et al., 2024b)	CodeS-7B	-	✗	64.6	46.9	40.3	57.0
SFT CodeS (Li et al., 2024b)	CodeS-15B	-	✗	65.8	48.8	42.4	58.5
Distillery (Maamari et al., 2024)	GPT-4o	-	✗	-	-	-	67.2
CHESS-SQL (Talaei et al., 2024)	Deepseek-Coder-33B	GPT-4-Turbo	✗	-	-	-	65.0
CHESS-SQL (Talaei et al., 2024)	Deepseek-Coder-33B	LLaMA3-70B	✗	-	-	-	61.5
CHASE-SQL (Pourreza et al., 2024)	Gemini-1.5-Pro	Gemini-1.5-Flash	✗	-	-	-	73.0
XiYan-SQL (Gao et al., 2024b)	?	?	✗	-	-	-	73.3
XiYan-SQL (Gao et al., 2024b)	Qwen2.5-Coder-32B	Qwen2.5-Coder-32B	✗	-	-	-	67.0
DAIL-SQL (Gao et al., 2024a)	GPT-4	SC Selection	✓	63.0	45.6	43.1	55.9
SuperSQL (Li et al., 2024a)	GPT-4	SC Selection	✓	66.9	46.5	43.8	58.5
MCS-SQL (Lee et al., 2024)	GPT-4	GPT-4	✓	-	-	-	64.4
RSL-SQL (Cao et al., 2024)	GPT-4o	GPT-4o	✓	74.4	57.1	53.8	67.2
Alpha-SQL (Ours)	**Qwen2.5-Coder-7B**	**SC Selection**	✓	**72.6**	**59.3**	**53.1**	**66.8**
Alpha-SQL (Ours)	**Qwen2.5-Coder-14B**	**SC Selection**	✓	**74.6**	**61.0**	**55.9**	**68.7**
Alpha-SQL (Ours)	**Qwen2.5-Coder-32B**	**SC Selection**	✓	**74.5**	**64.0**	**57.2**	**69.7**

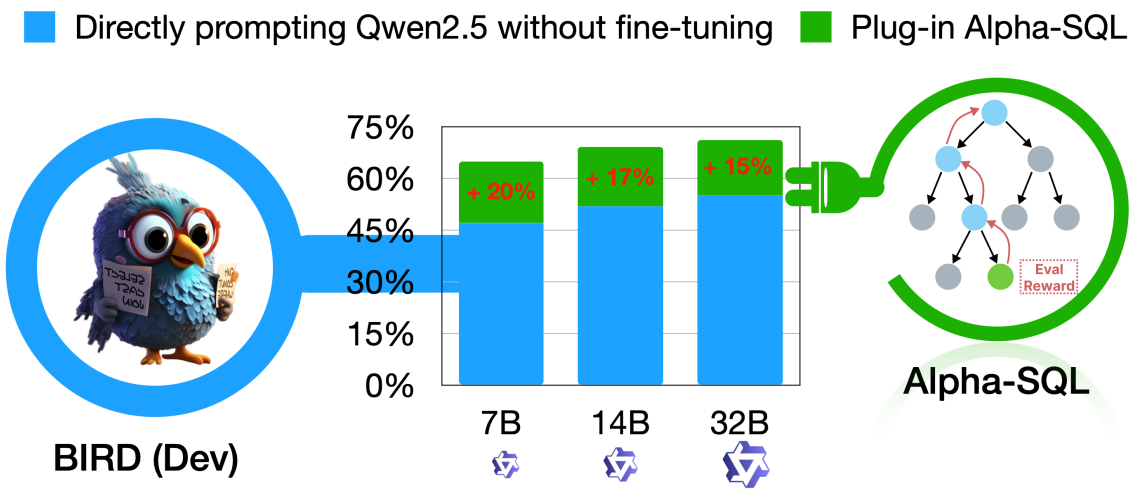
Table 3. Execution Accuracy on Spider Development Dataset.

Method	Inference Model	Selection Model	Zero-shot Setting	Accuracy (%)				
				Easy	Medium	Hard	Extra Hard	All
SFT CodeS (Li et al., 2024b)	CodeS-7B	-	✗	94.8	91.0	75.3	66.9	85.4
SFT CodeS (Li et al., 2024b)	CodeS-15B	-	✗	95.6	90.4	78.2	61.4	84.9
C3-SQL (Dong et al., 2023)	GPT-3.5-Turbo	SC Selection	✓	92.7	85.2	77.6	62.0	82.0
DIN-SQL (Pourreza & Rafiei, 2023)	GPT-4	-	✓	92.3	87.4	76.4	62.7	82.8
DAIL-SQL (Gao et al., 2024a)	GPT-4	SC Selection	✓	91.5	90.1	75.3	62.7	83.6
ZeroNL2SQL (Fan et al., 2024)	GPT-4	-	✓	-	-	-	-	84.0
MAC-SQL (Wang et al., 2023)	GPT-4	-	✓	-	-	-	-	86.8
SuperSQL (Li et al., 2024a)	GPT-4	SC Selection	✓	94.4	91.3	83.3	68.7	87.0
Alpha-SQL (Ours)	**Qwen2.5-Coder-7B**	**SC Selection**	✓	**94.0**	**89.2**	**76.4**	**63.3**	**84.0**
Alpha-SQL (Ours)	**Qwen2.5-Coder-14B**	**SC Selection**	✓	**94.0**	**91.0**	**79.9**	**72.3**	**87.0**

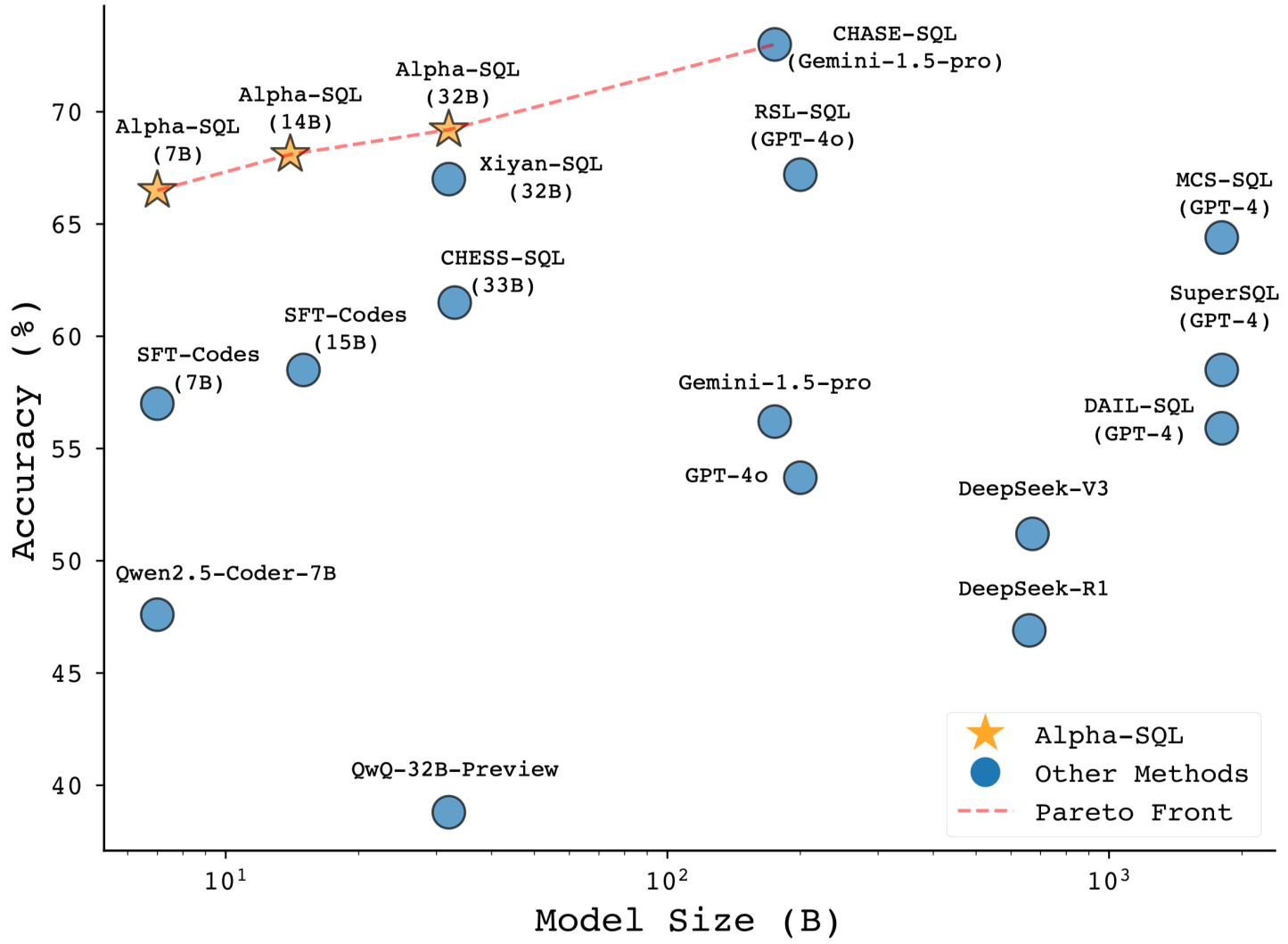
Alpha-SQL: Plug-and-Play Capabilities

Table 4. Comparison with Baseline LLMs on the SDS dataset.

Model	Accuracy (%)
Deepseek-V3	51.2
GPT-4o	53.7
Gemini-1.5-Pro	56.2
QwQ-32B-Preview	38.8
DeepSeek-R1	50.3
Gemini-2.0-Flash-Thinking-Exp	60.8
Qwen2.5-Coder-7B	47.6
+ Alpha-SQL (Ours)	**64.6 (↑ 17.0)**
Phi-4	43.5
+ Alpha-SQL (Ours)	**60.0 (↑ 16.5)**



Performance-Scale Trade-off Analysis



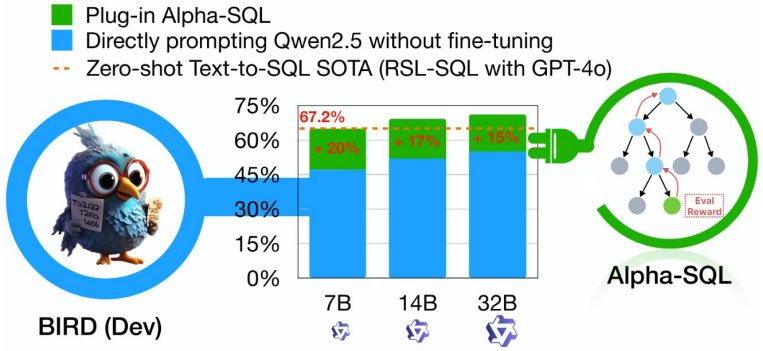
Research Opportunities

- **Human-as-an-Agent and Human-in-the-Reasoning-Loop**
  - How can we dynamically *integrate human experts into the reasoning loop* to address complex tasks beyond LLM agents' current capabilities and clarify the question ambiguities?
- **Explainable and Interpretable SQL Reasoning Agents**
  - Users typically require explanations for the reasoning steps and decisions underlying SQL generation (i.e., knowing both "what" and "why").
  - How can we design reasoning agents that transparently communicate their thought processes, decisions, and final SQL statements to improve system transparency and foster user trust?
- **Metadata Management and Schema Interpretation**
  - Real-world databases commonly feature complex schemas, detailed metadata (e.g., column annotations, table descriptions, foreign key constraints, data types).
  - How can we enable *data agents* to effectively extract, manage, and utilize this metadata to generate more accurate semantic mappings, informed reasoning processes, and precise SQL generation?

🚀 Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search

Homepage | ICML 2025 | arXiv 2502.17248 | Python 3.11.11 | License MIT

✨ If you find our work helpful, please don't hesitate to give us a star ⭐ !



https://github.com/HKUSTDial/Alpha-SQL

Thanks!

Dr. Yuyu LUO

Data Science and Analytics Thrust

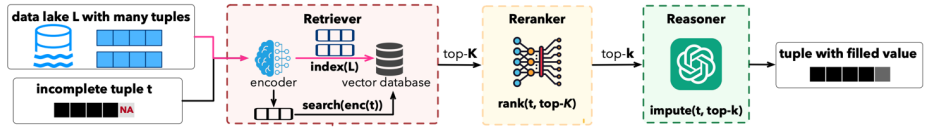
Information Hub, HKUST(GZ)

yuyuluo@hkust-gz.edu.cn

http://luoyuyu.vip

Agents-powered Data Analytics

RAG | **Text-to-SQL** | **Document QA** | **Data Visualization**



[SuperSQL, VLDB 2024] [Alpha-SQL, ICML 2025] [NL2SQL-Bugs, KDD 2025]

[LineNet, SIGMOD 2023] [HAIChart, VLDB 2024] [LakeFill, VLDB 2025]

[StatQA, NeurIPS 2024] [Weak2Strong, VLDB 2025] [HARVis, CHI 2025]

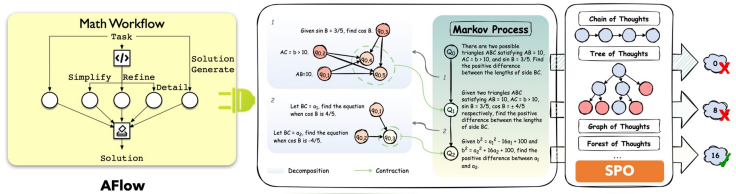
LLMs & Agents

1. OpenManus: An open-source framework for building general AI agents

OpenManus | Watch 373 | Fork 7.8k | Starred 45.1k

2. From LLM Agents to Foundation Agents *[ICLR 2025, Oral Paper, Top-1.8%]*

- We address three core challenges for Foundation Agents
  - (Q1) Designing agentic workflows with AFlow.
  - (Q2) Enhancing cost-efficient reasoning with AoT.
  - (Q3) Optimizing action-specific prompts with SPO.



AFlow
Automating Agentic Workflow Generation
[ICLR 2025 Oral Paper, top-1.8%]

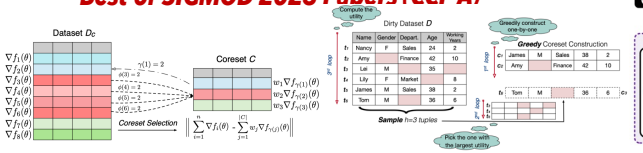
AoT
Atom of Thoughts for LLM Test-Time Scaling

SPO
Self-Supervised Prompt Optimization

Data-centric AI

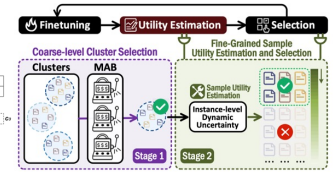
GoodCore: Data-effective and Data-efficient Machine Learning

Best-of-SIGMOD 2023 Papers (CCF-A)



[TODS 2025, VLDB 2022]

Iterative Data Selection for LLM Instruction Tuning



Alpha-SQL: Upper Bound Accuracy

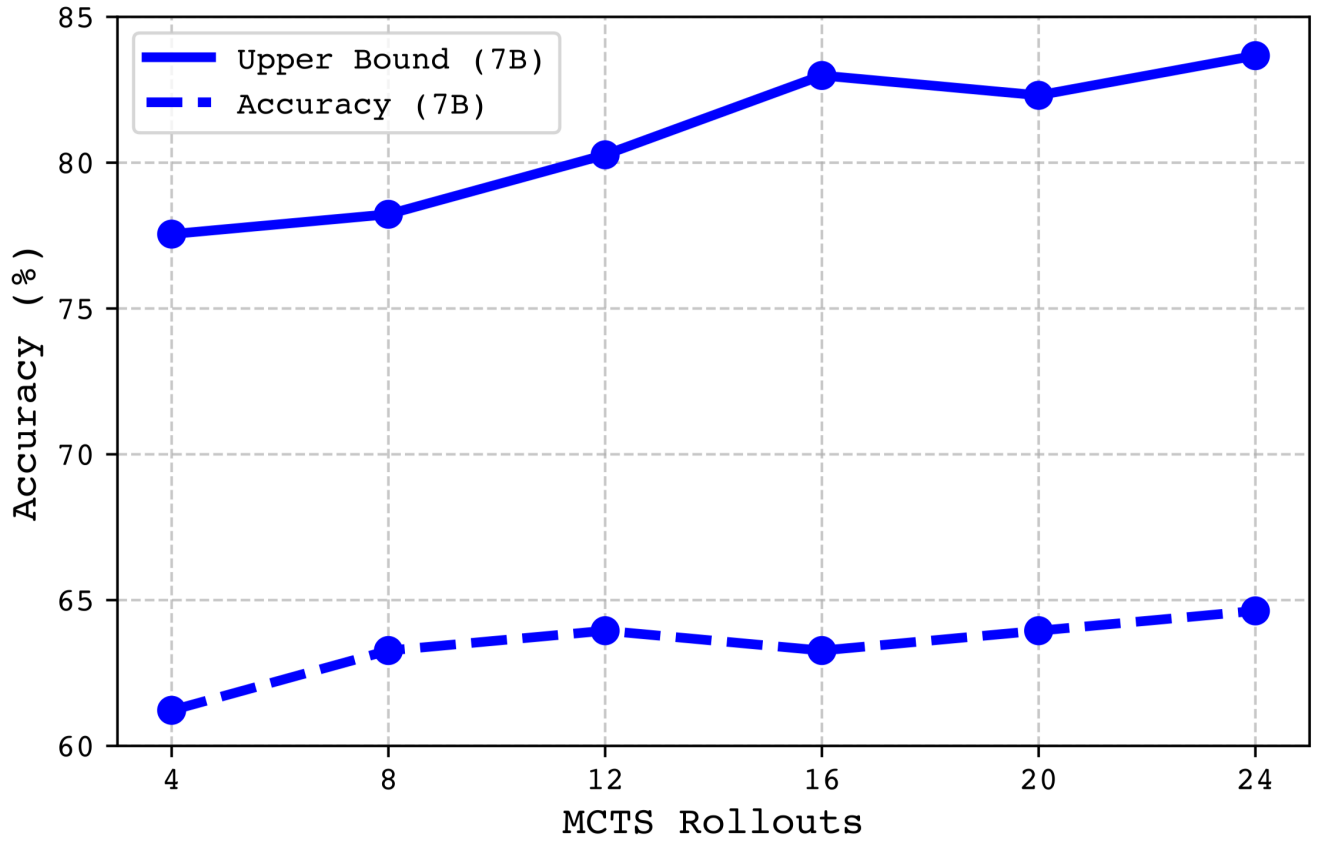


Figure 5. Accuracy vs. MCTS Rollouts.

Alpha-SQL: Top-K Accuracy

Methods	Base LLMs	Top-K	Accuracy
Alpha-SQL	Qwen2.5-Coder-32B	Top-1	**69.7%**
Alpha-SQL	Qwen2.5-Coder-32B	Top-2	**78.4%**
Alpha-SQL	Qwen2.5-Coder-32B	Top-3	**80.8%**
Alpha-SQL	Qwen2.5-Coder-32B	Top-4	**81.6%**
Alpha-SQL	Qwen2.5-Coder-32B	Top-5	**81.7%**