Research track - Data cleaning and preparation I

Weak-to-Strong Prompts with Lightweight-to-Powerful LLMs for High-Accuracy, Low-Cost, and Explainable Data Transformation

Authors: Changlun Li[1], Chenyu Yang[1], Yuyu Luo[1], Ju Fan[2], Nan Tang[1]

Affiliations: HKUST(GZ)[1], Renmin University of China[2]

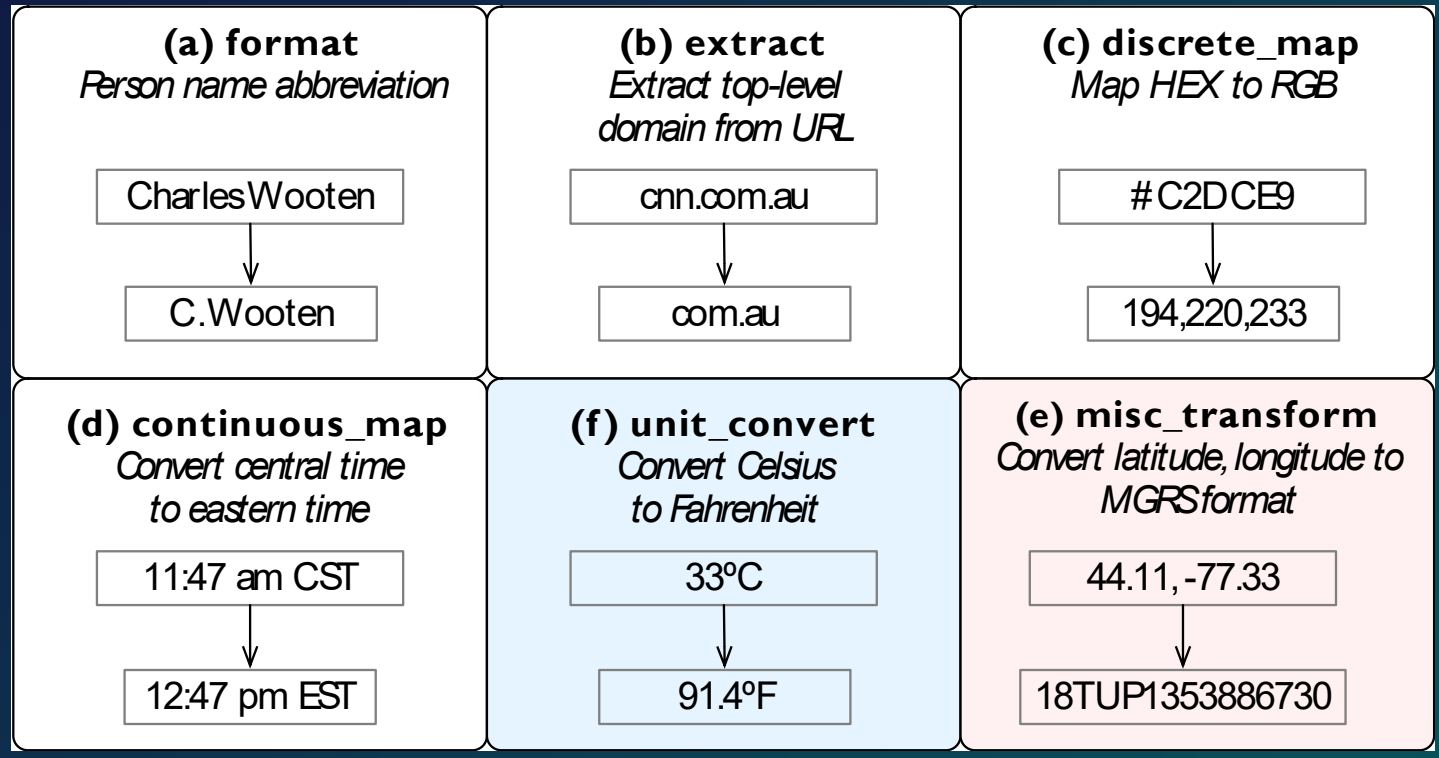
Data Transformation

What

It converts data from one format, structure, or value to another.

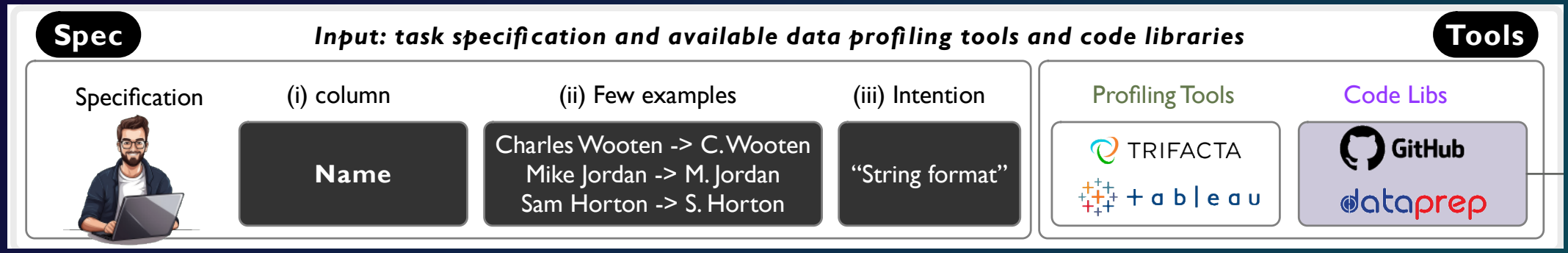
Importance

It is essential for integration and compatibility in data management studies.

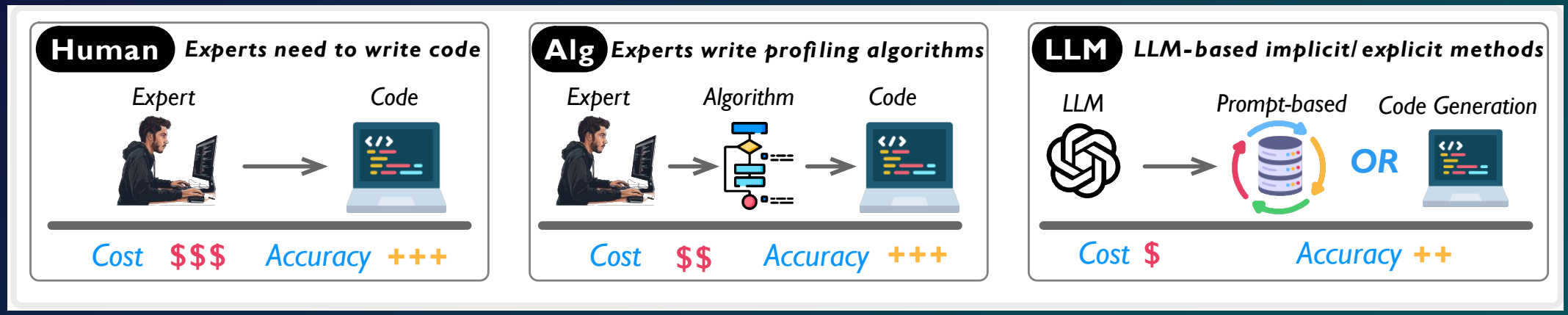


Existing Solutions

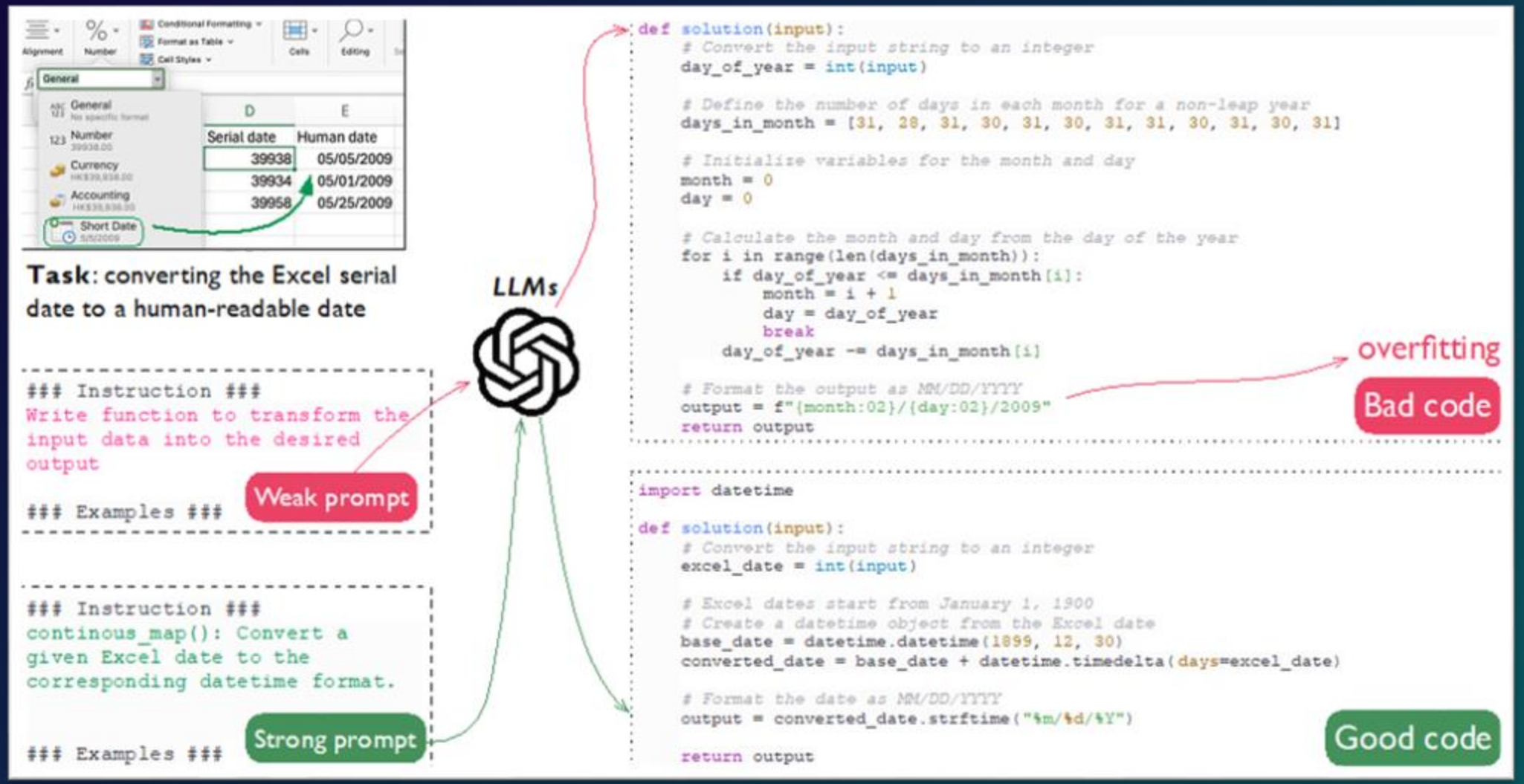
Scenario



(i) Human-based; (ii) Algorithm-based; (iii) LLM-based



Prompt quality signifies code quality



The Challenge of Code Generation Approach

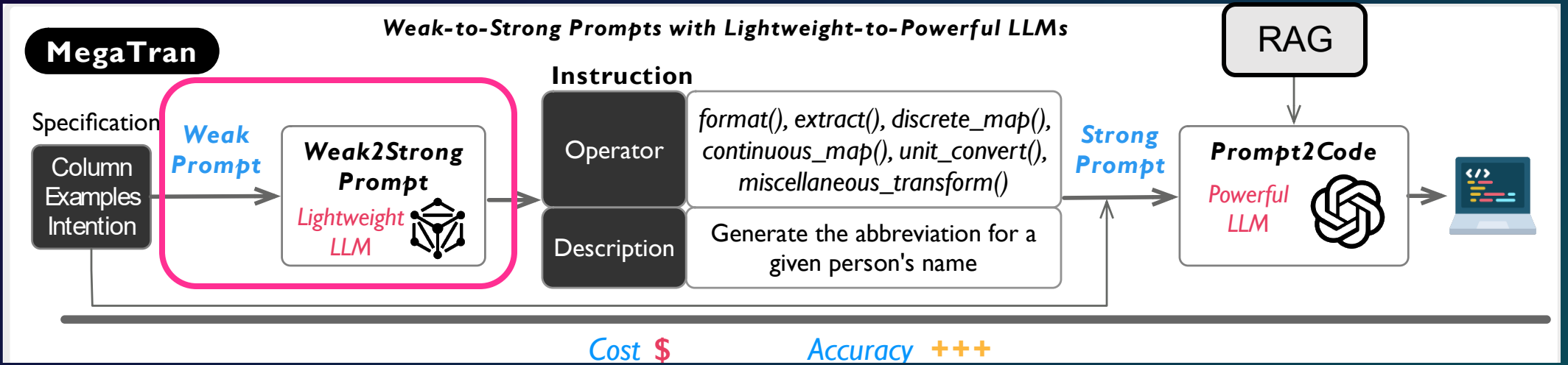
Problem Focus

- Users often provide "weak prompts”, which describe imprecise or incomplete transformation requests.
- The complexity of diverse data formats and the need for domain-specific knowledge present major hurdles.

Our Goal

- High-accuracy, low-cost, and explainable solution!

MegaTran Framework Overview

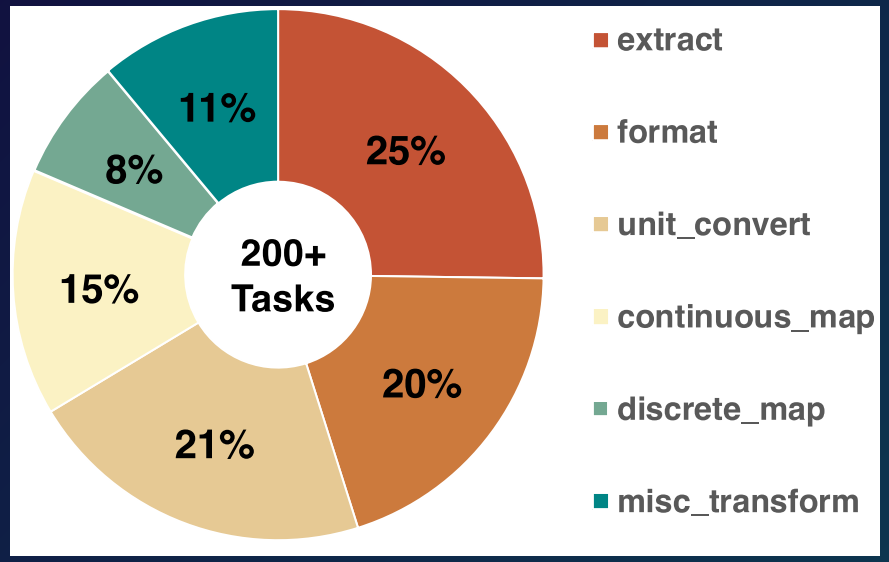


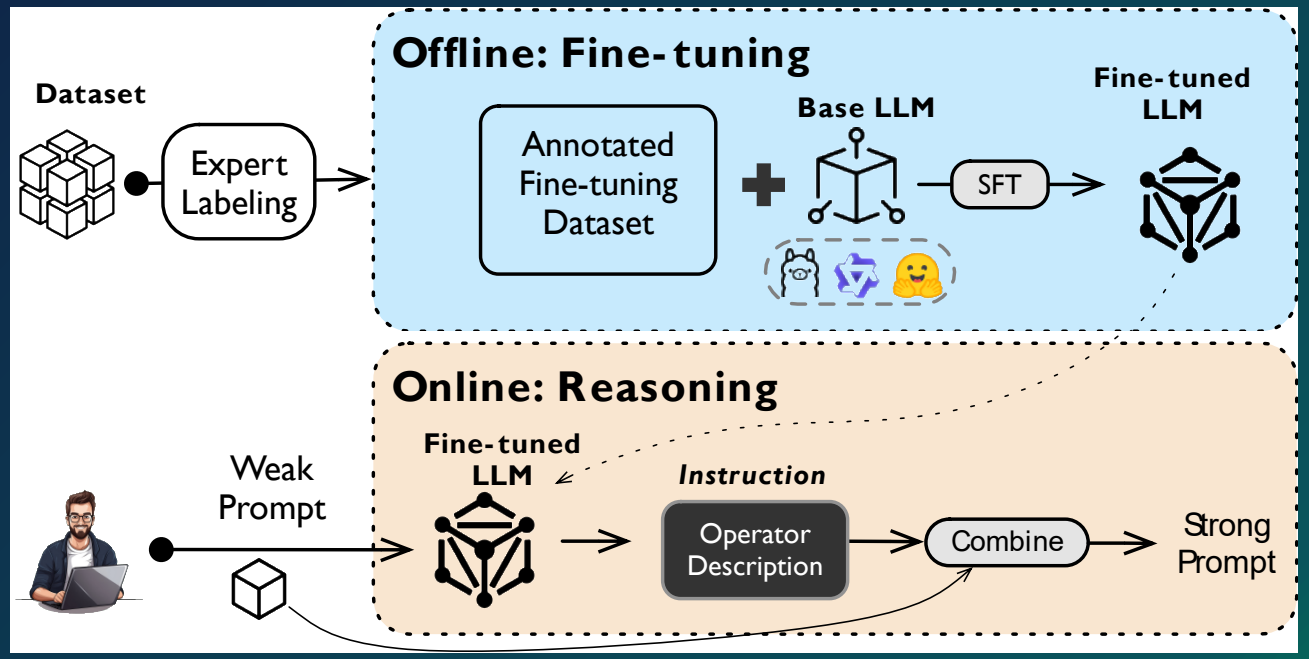
It consists of **two** main stages:

1. **Weak2StrongPrompt:** A lightweight LLM converts a user's "weak" prompt (imprecise input) into a "strong," structured prompt.
2. **Prompt2Code:** A powerful LLM uses the strong prompt to generate the transformation code with self-reflection and RAG techniques.

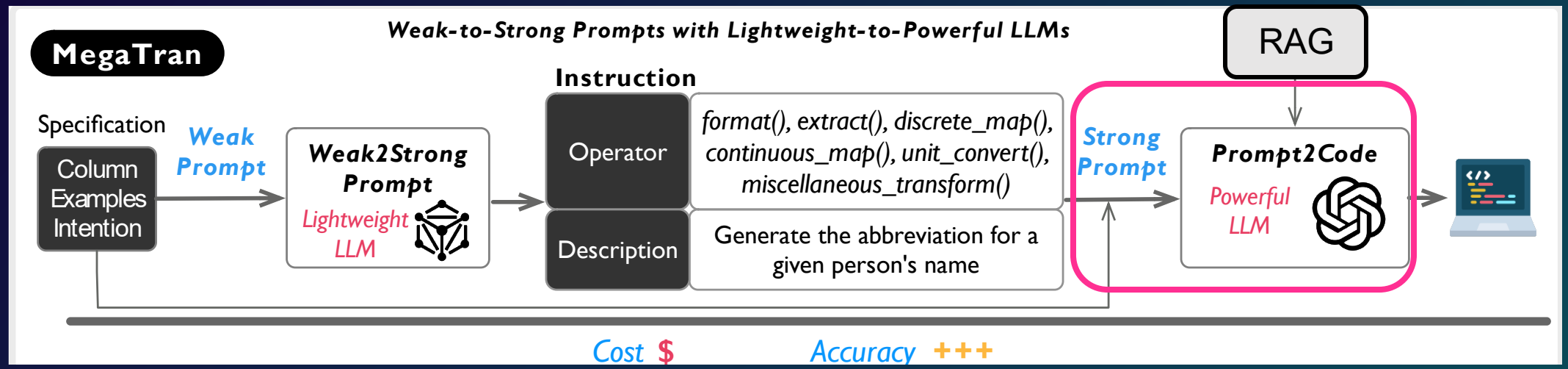
Stage 1: Weak2Strong Prompt

- We define the common transformation type then curate fine-tuning dataset.
- **Offline**: base LLM (Llama3-8B in use) fine-tuning with annotated dataset.
- **Online**: argument weak prompt with operator-level instructions.





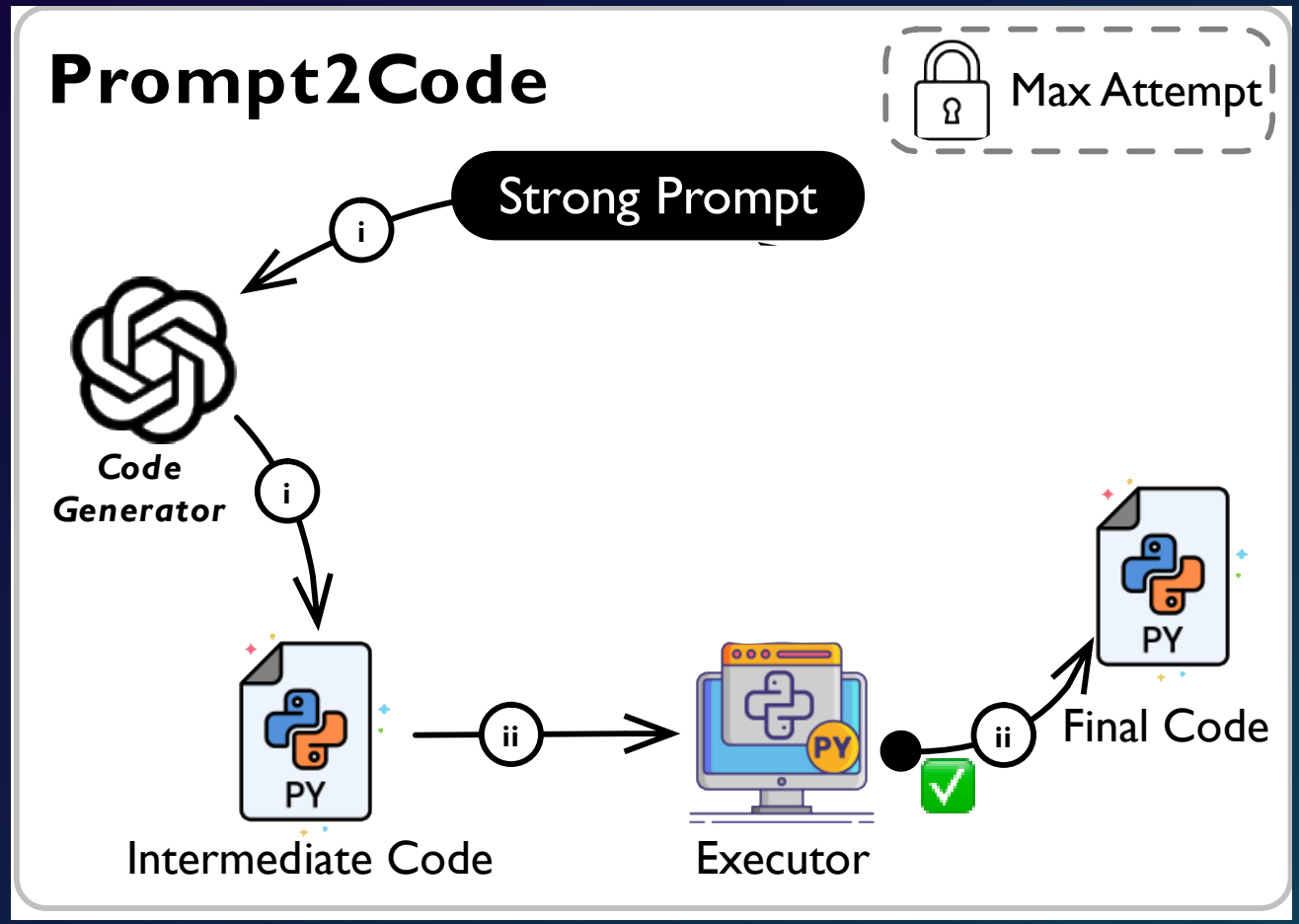
MegaTran Framework Overview (Cont'd)



It consists of **two** main stages:

1. **Weak2StrongPrompt:** A lightweight LLM converts a user's "weak" prompt (imprecise input) into a "strong," structured prompt.
2. **Prompt2Code:** A powerful LLM uses the strong prompt to generate the transformation code with self-reflection and RAG techniques.

Stage 2: Prompt2Code



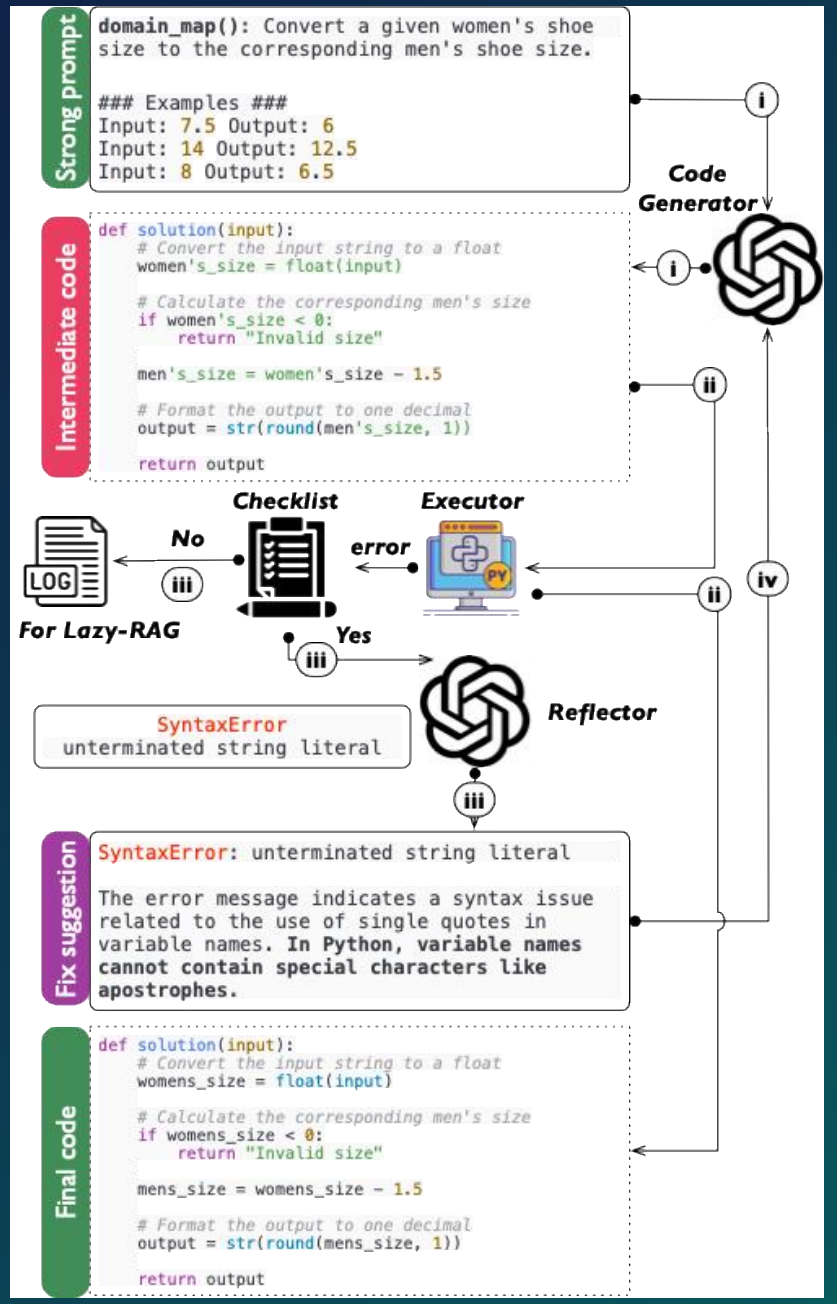
A powerful LLM generates transformation code given the strong prompt.

Optimizations:

1. **Sanity-Check Reflection:** Iterative debugging and refinement.
2. **Lazy-RAG:** Retrieves relevant code snippets, docs to improve code quality.

Sanity-Check Reflection

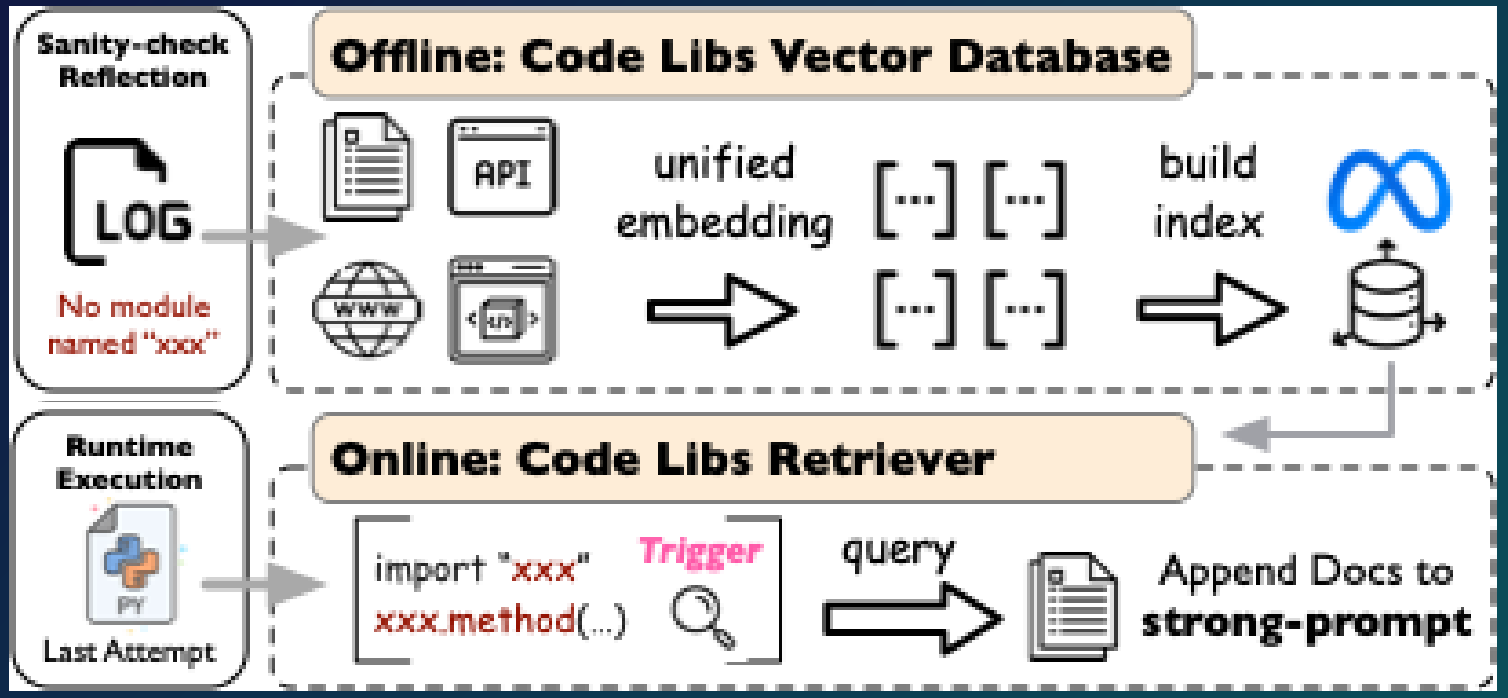
- It uses a pre-defined **checklist** of common programming errors (e.g., *SyntaxError, ValueError*).
- The Reflector LLM analyzes the error and generate a fix suggestion.
- The Code Generator will incorporate the suggestion to fix the code.



Lazy-RAG

Offline: A pre-built **Code Libs Vector Database** to store documentation and code snippets from sources like GitHub and Stack Overflow.

Online: Similarity-based **Code Libs retriever** conducts analysis and avoids introducing noise and irrelevant information into the code generation process.



Experiment Setup

Datasets

- **Five** benchmark datasets
- Dev set: StackOverflow (SO), Bing-QueryLogs (BQ)
- Test set: ETL, CommonSense (COM), and Science (SCI)

Evaluation Metrics

- **Pass@K Accuracy:** Measures task-level proportion if at least one correct solution that passed all test within K attempts.
- **Cost Efficiency:** Inference time, token consumption, and API calls.

Baselines

- Compared against **search-based/ LLM-based** SoTA methods
- TDE [VLDB'18], FM [VLDB'22], Code LLM, DTT [SIGMOD'24], and UniDM [MLSys'24].

Exp-1: High Accuracy

Pass@3 accuracy gains from +2.2% to +26.1% compared to SoTA methods.

Dataset	SO	BQ	ETL	COM	SCI
No. Tasks	**49**	**50**	**46**	**43**	**23**
TDE [VLDB'18]	63.0	32.0	-	-	-
FM [VLDB'22]	65.3	54.0	-	-	-
UniDM [MLSys'24]	67.4	56.0	-	-	-
DTT [SIGMOD'24]	4.1	1.1	26.1	9.3	4.4
By GPT-4o					
FM	75.6	64.0	56.5	65.1	30.4
Code LLM	67.3	56.0	65.2	55.8	39.1
MegaTran	**77.6**	**78.0**	**67.4**	**74.4**	**65.2**
By GPT-4o-mini					
FM	67.3	54.0	56.5	53.5	21.7
Code LLM	61.2	56.0	58.7	44.2	34.8
MegaTran	**75.5**	**74.0**	**60.9**	**69.8**	**52.2**

Exp-2: Low cost

- Fewer API calls for code generation (up to 30.6%).

API Calls	SO	BQ	ETL	COM	SCI
MegaTran	68	76	73	66	52
Code LLM	98	94	89	80	50

- Less time cost with Lazy-RAG (up to 39.5%).

Time Cost	SO	BQ	ETL	COM	SCI
Lazy	4m13s	6m24s	5m9s	7m52s	4m33s
Eager	4m57s	7m32s	8m31s	8m21s	6m1s

Exp-3: Ablation of each module

- The Weak2StrongPrompt module is the most critical; removing it causes the huge degradation, 20% ↓ .
- Both the Sanity-check Reflection and Lazy-RAG optimizations contribute meaningfully to code generation quality.

	SO	BQ	ETL	COM	SCI
GPT-4o	77.6	78.0	67.4	74.4	65.2
- w/o Weak2StrongPrompt	-8.2	-14.0	-2.2	-13.9	-13.0
- w/o Sanity-check Reflection	-2.1	-4.0	0	-9.3	-30.4
- w/o Lazy-RAG	-	-2.0	-	-2.3	-
GPT-4o-mini	75.5	74.0	60.9	69.8	52.2
- w/o Weak2StrongPrompt	-12.2	-20.0	-6.6	-18.6	-17.4
- w/o Sanity-check Reflection	-4.1	-8.0	-3.1	-16.3	-21.8
- w/o Lazy-RAG	-	-4.0	-	-2.4	-

Takeaway

Conclusion

- We present a high-accuracy, low-cost, and explainable framework for handling data transformation tasks by LLM-driven code generation.

Open problem

- Explore multivariate and complex nested transformations.
- Representative few-shot sampling for code generation.

Thanks for listening! Q&A

Paper with Code

Contact Us: Prof. Nan Tang, nantang@hkust-gz.edu.cn

Data Intelligence and Analytics Lab

Data Science and Analytics Thrust, Information Hub

The Hong Kong University of Science and Technology (Guangzhou)