中国科学技术大学
University of Science and Technology of China

认知智能全国重点实验室
STATE KEY LABORATORY OF COGNITIVE INTELLIGENCE

ICML 2024 Challenges on Automated Math Reasoning

Zhiyuan Ma[1,2], Jiayu Liu[1,2], Zhenya Huang[1,2], Qi Liu[1,2] , Jing Sha[2,3], Shijin Wang[2,3], Enhong Chen[1,2]

[1]Anhui Province Key Lab. of Big Data Analysis and Application,
University of Science and Technology of China
[2]State Key Laboratory of Cognitive Intelligence
[3]iFLYTEK

Reporter: Zhiyuan Ma

Outline

Self-introduction

- Team **Cogbase**
  - Affiliation
    - [BDAA-BASE]
      - BASE Group is part of the Anhui Province Key Laboratory of Big Data Analysis and Application of School of Computer Science and Technology, University of Science and Technology of China (USTC).
      - Our research mainly focuses on applying BigData and AI techniques to education and science. Some of the topics include: educational resource understanding, cognitive diagnosis and student modeling, personalized education services, etc.
  - 1st place in Track1-1: Autoformalization
  - 1st place in Track1-2: Auto-informalization
  - 2nd place in Track2: AUTOMATED THEOREM GENERATION AND PROVING
  - 2nd place in Track3: AUTOMATED OPTIMIZATION PROBLEM-SOLVING WITH CODE

Background

- Background
  - Mathematics has always been regarded as the touchstone of artificial intelligence
  - Autoformalization (and its reverse direction) is the core building block of developing effective formal mathematical reasoning systems.
  - The Autoformalization & Auto-informalization Challenge is thus to encourage the community to investigate advanced methods or models that provide more precise formal-informal proof alignment and translation.

Background

- Task
  - Track 1-1: Autoformalization
    - Given a problem statement and its proof in natural language, generate the corresponding formal statement and proof.
    - Input & Output: (informal statement, informal proof) → formal_proof
  - Track 1-2: Auto-informalization
    - Given a formal statement and its formal proof, generate the corresponding natural language statement and proof.
    - Input & Output: formal proof → (informal_statement, informal_proof)
  - Data
    - name: the name of the math problem;
    - informal statement: the natural language statement of the problem;
    - informal proof: the natural language answer to the problem;
    - formal proof: the Lean 3 formal statement and solution to the problem.

Background

Example

```
{
   "problem_name": "correct_by_msg__ELEM_theorem_proving_1st_grade_15_round2",

   "informal_statement": "If a two-digit number is formed with 4 in the tens place and 3 in the ones place, prove that the number is 43.",

   "informal_proof": "We know that the number in the tens place represents tens and the number in the ones place represents ones. So, if 4 is in the tens place, it represents 4 * 10 = 40. If 3 is in the ones place, it represents 3. So the total number is 40 + 3 = 43.",

   "formal_proof": "def ten (n : ℕ) : ℕ := n * 10\ndef one (n : ℕ) : ℕ := n\ntheorem two_digit_number : ten 4 + one 3 = 43 :=\nbegin\n  have h1 : ten 4 = 40, by refl, \n  have h2 : one 3 = 3, by refl, \n  rw [h1, h2],\n  exact add_comm 3 40\nend."
}
```

Evaluation

- 1-1： Rouge-L， BLUE， Lean 3 formal proof code passrate
- 1-2: Rouge-L， BLUE

Outline

Idea

- Utilizing GPT-4's code generation capabilities to harness its potential for generating Lean3 code.

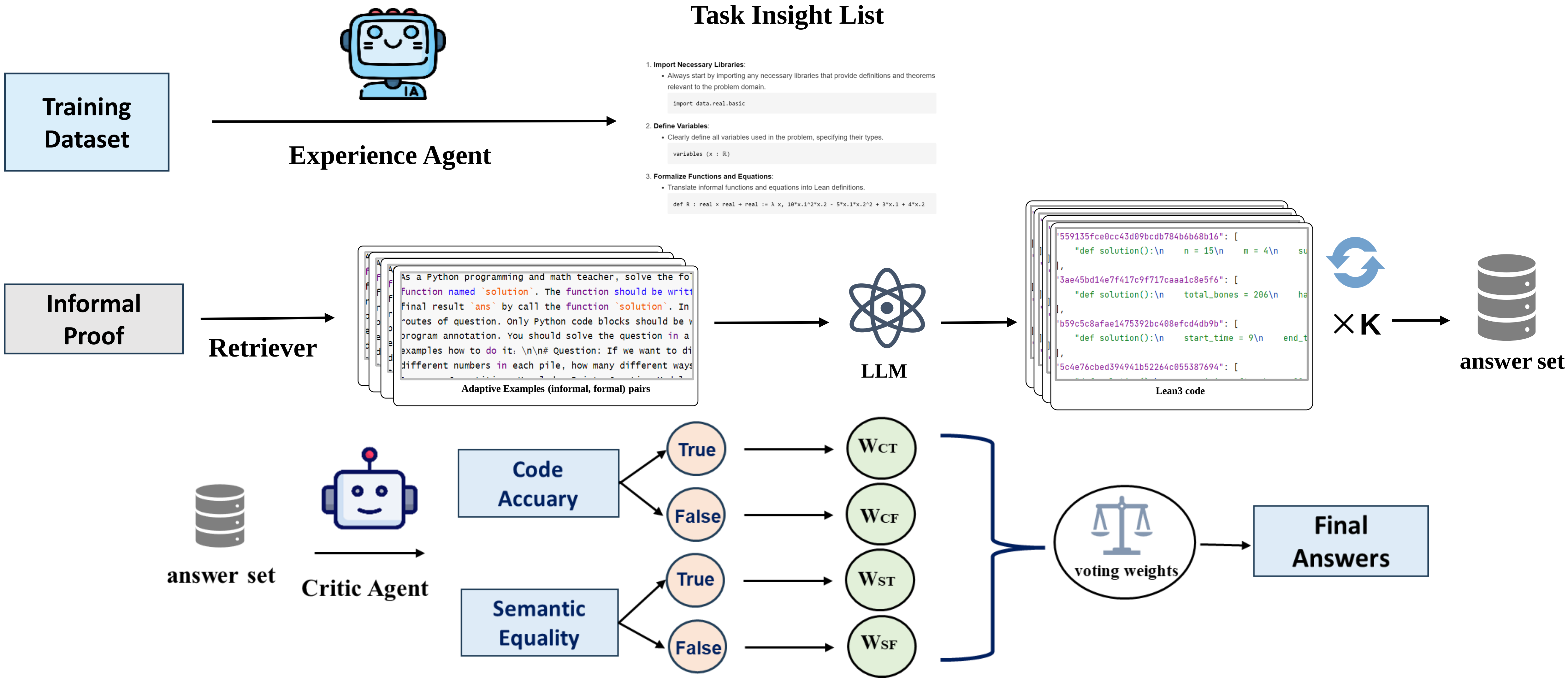
- To better utilize the data provided by the training set:
  - Using RAG (Retrieval-Augmented Generation) technology to retrieve similar problems as examples
  - Constructing an Agent to summarize experiences from input-output pairs

- Verify the quality of generated code from multiple perspectives

Outline

Track 1-1 Solution

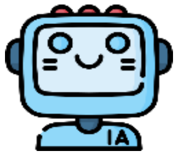
➢Framework



Solution

- Details
  - Summarize Experiences
    - Summarize experiences and insights from the training dataset
    - The insight is an abstraction of the task, as well as its patterns, templates, and common pitfalls.

Training Dataset



Experience Agent

1. **Import Necessary Libraries**:
   - Always start by importing any necessary libraries that provide definitions and theorems relevant to the problem domain.

```
import data.real.basic
```

2. **Define Variables**:
   - Clearly define all variables used in the problem, specifying their types.

```
variables (x : ℝ)
```

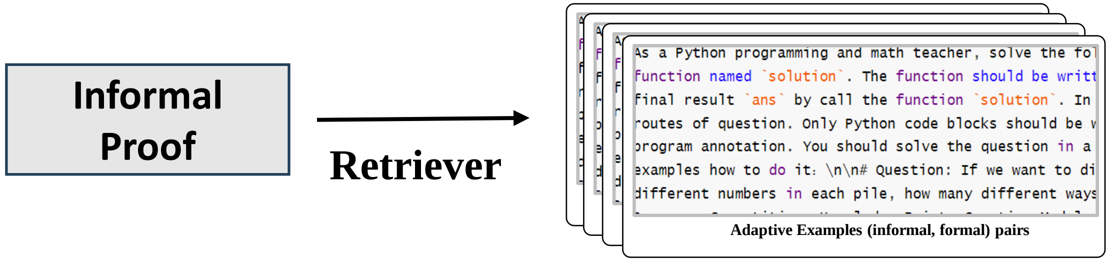
3. **Formalize Functions and Equations**:
   - Translate informal functions and equations into Lean definitions.

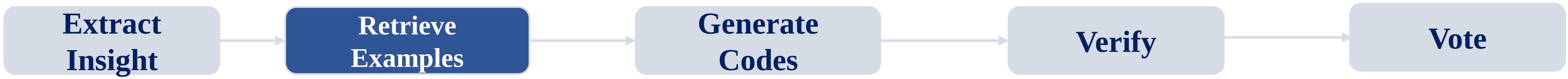
```
def R : real × real → real := λ x, 10*x.1^2*x.2 - 5*x.1*x.2^2 + 3*x.1 + 4*x.2
```

Task Insight List

Extract Insight → Retrieve Examples → Generate Codes → Verify → Vote

- Details
  - **Retrieve Examples**
    - Retrieve similar examples from the training set, based on semantics, knowledge points, difficulty level, etc.
    - Rank and score examples based on similarity, selecting the top K most relevant examples.
    - Concatenate these examples to the prompt for contextual learning, enhancing the LLM's understanding of the task.





Solution

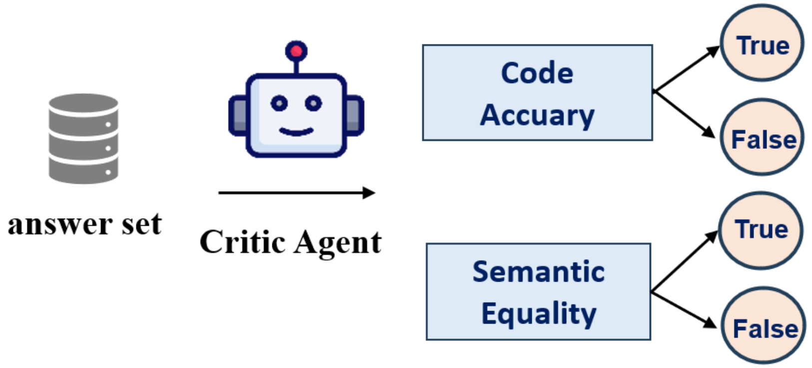
- Details
  - **Generate Codes**
    - Input into the LLM: The model will use the provided insights and examples to generate Lean 3 code.

````
You are a math expert and familar with Lean 3 formal language. Now
please translate the following statement and solution of a math word
problem into Lean 3 formal solution. Given a informal problem and its
informal solution, analyze the mathematical problem and gain an in-depth
understanding of the informal solution, then generate the corresponding
formal solution in Lean 3. You should output the code in the ```lean xxx
``` tag. Please note that the formal solution should be able to pass the
Lean 3 compiler at first, then the informal solution and the formal
solution need to be identical.
````

Extract Insight → Retrieve Examples → **Generate Codes** → Verify → Vote
````

Solution

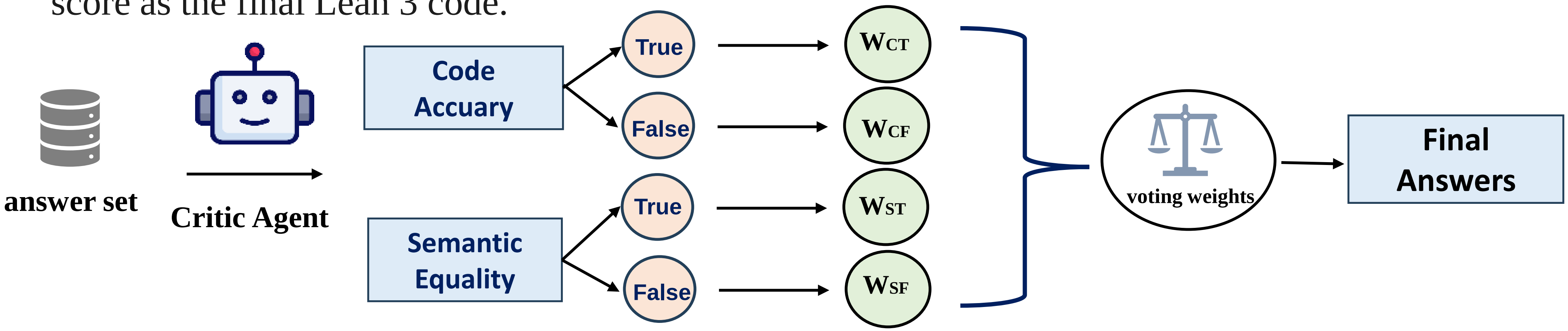
- Verify
  - CRITIC Model
    - Code Check: Ensures that the generated code is syntactically correct and can be compiled without errors
    - Syntax Check: Verifies that the code logically solves the problem as intended, matching the semantics of the original questions.
  - Multi-Angle Assessment
    - Code Accuary
    - Semantic Accuary



Extract Insight → Retrieve Examples → Generate Codes → **Verify** → Vote

Solution

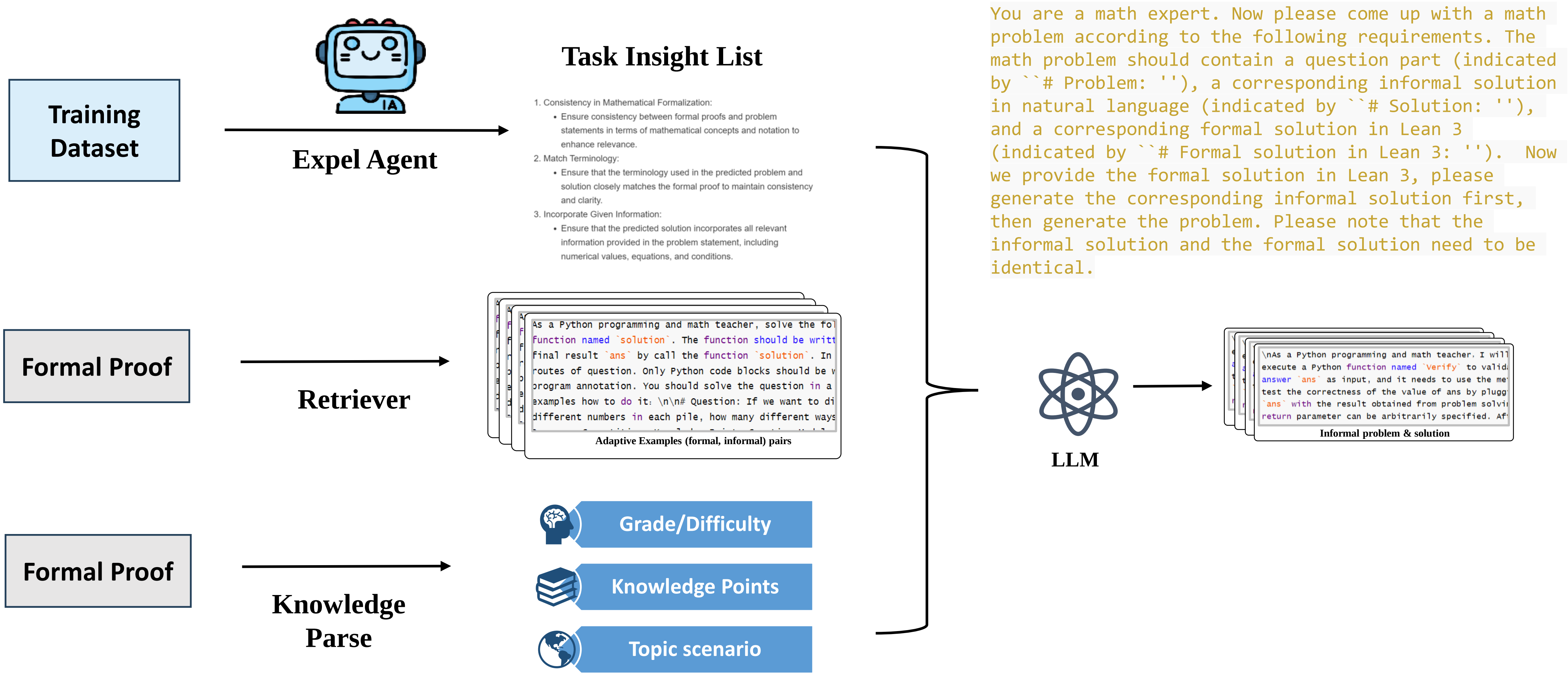
- Details
  - Multiple sampling
  - Selecting the best result as the final code
    - Select the Best Sample: Compare the scores of all samples and select the one with the highest total score as the final Lean 3 code.



Extract Insight → Retrieve Examples → Generate Codes → Verify → **Vote**

Track 1-2 Solution

➢Framework



Outline

1	Background
2	Idea
3	Solution
4	Conclusion

Conclusion

- Advantages of Our Method
  - We use both RAG and Experience Agent to stimulate the potential of LLM
  - We use self-verification adaptive dynamic voting to select right results from the output space, which improves the stability
- Future Direction
  - The automatic conversion between formal and informal proof is challenging for LLMs
  - Fine-tuning the LLM to inject mathematical knowledge in order to better accomplish this task.

Q & A

Thanks for your listening!

Reporter: Zhiyuan Ma

zhyma@mail.ustc.edu.cn
https://bigdata.ustc.edu.cn

Workshop Organizers

COLLABORATING ORGANIZATIONS