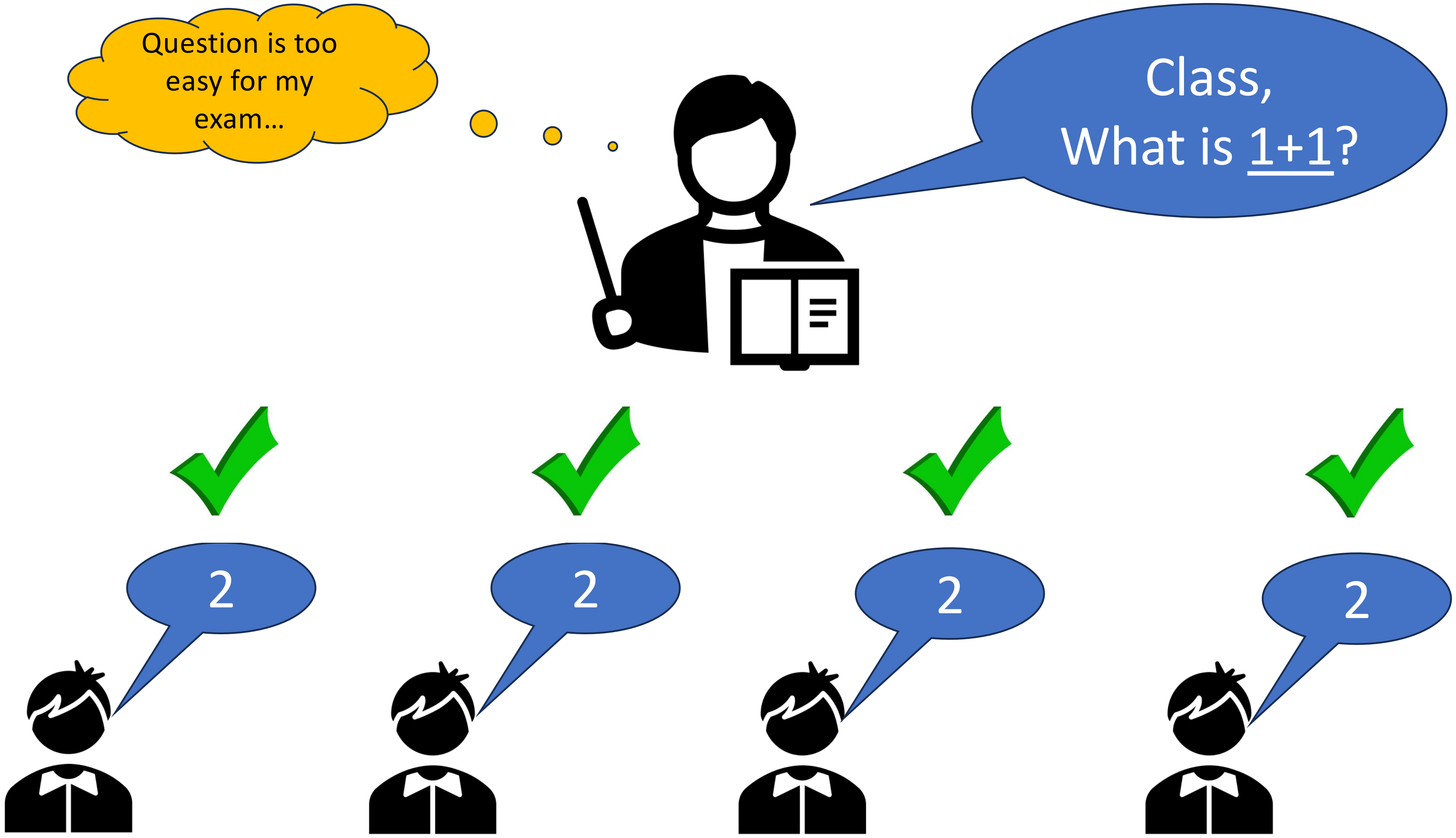
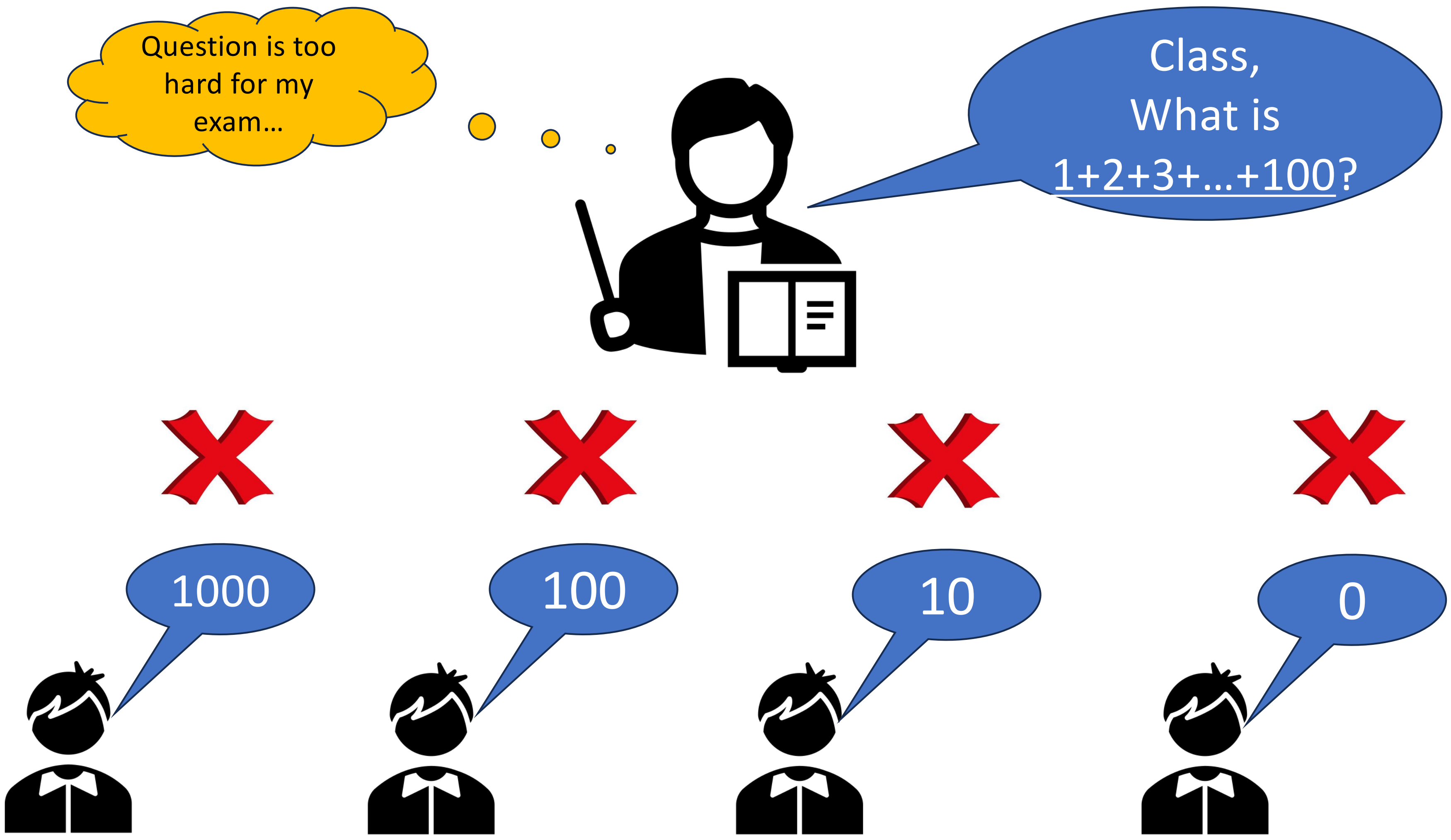
R-Zero: Self-Evolving Reasoning LLM from Zero Data

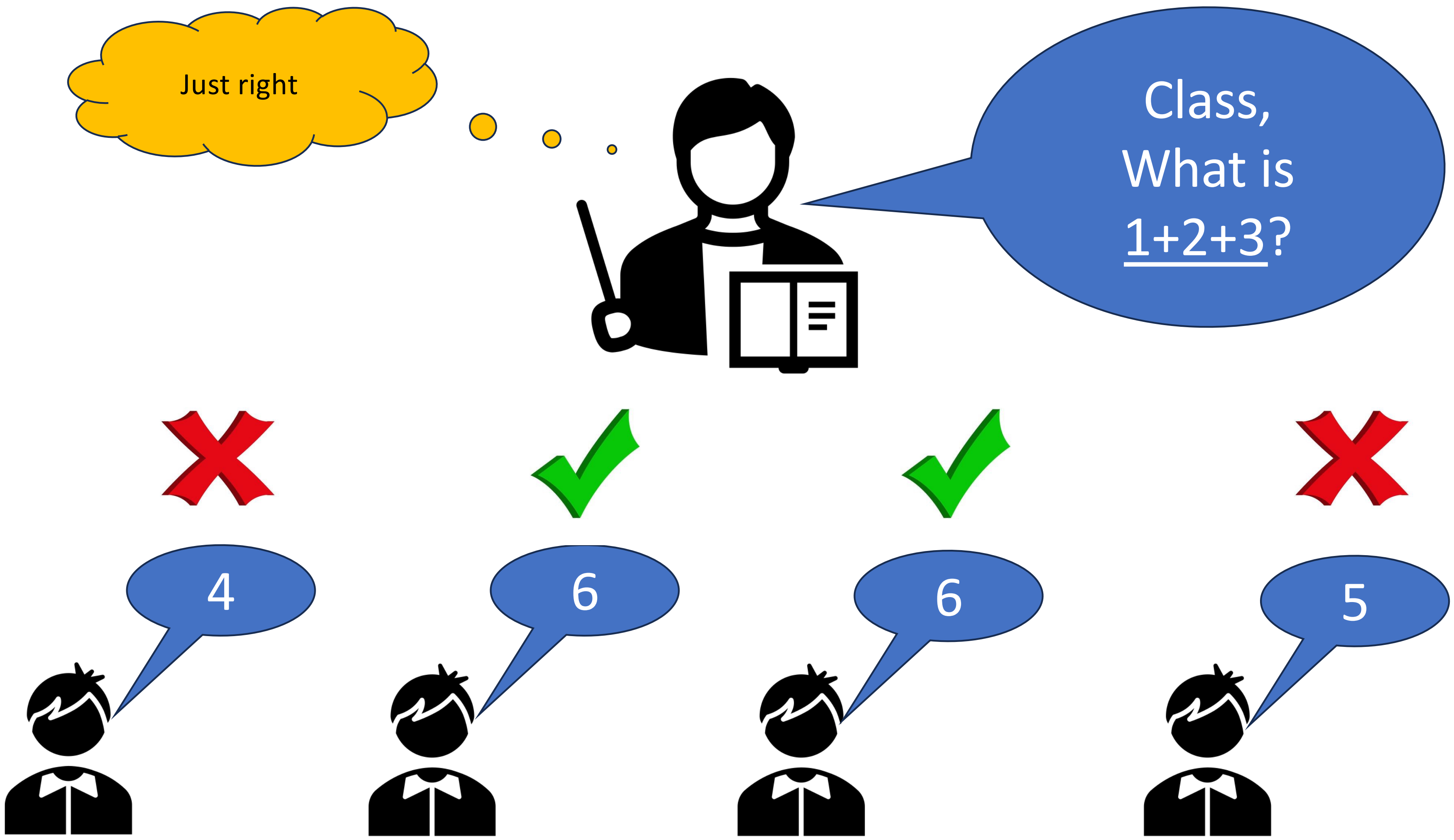
Chengsong Huang[1,2✉], **Wenhao Yu**[1✉], **Xiaoyang Wang**[1], **Hongming Zhang**[1], **Zongxia Li**[1,3], **Ruosen Li**[1,4], **Jiaxin Huang**[2], **Haitao Mi**[1], **Dong Yu**[1]
[1]Tencent AI Seattle Lab, [2]Washington University in St. Louis,
[3]University of Maryland, College Park, [4]The University of Texas at Dallas
`chengsong@wustl.edu; wenhaowyu@global.tencent.com`

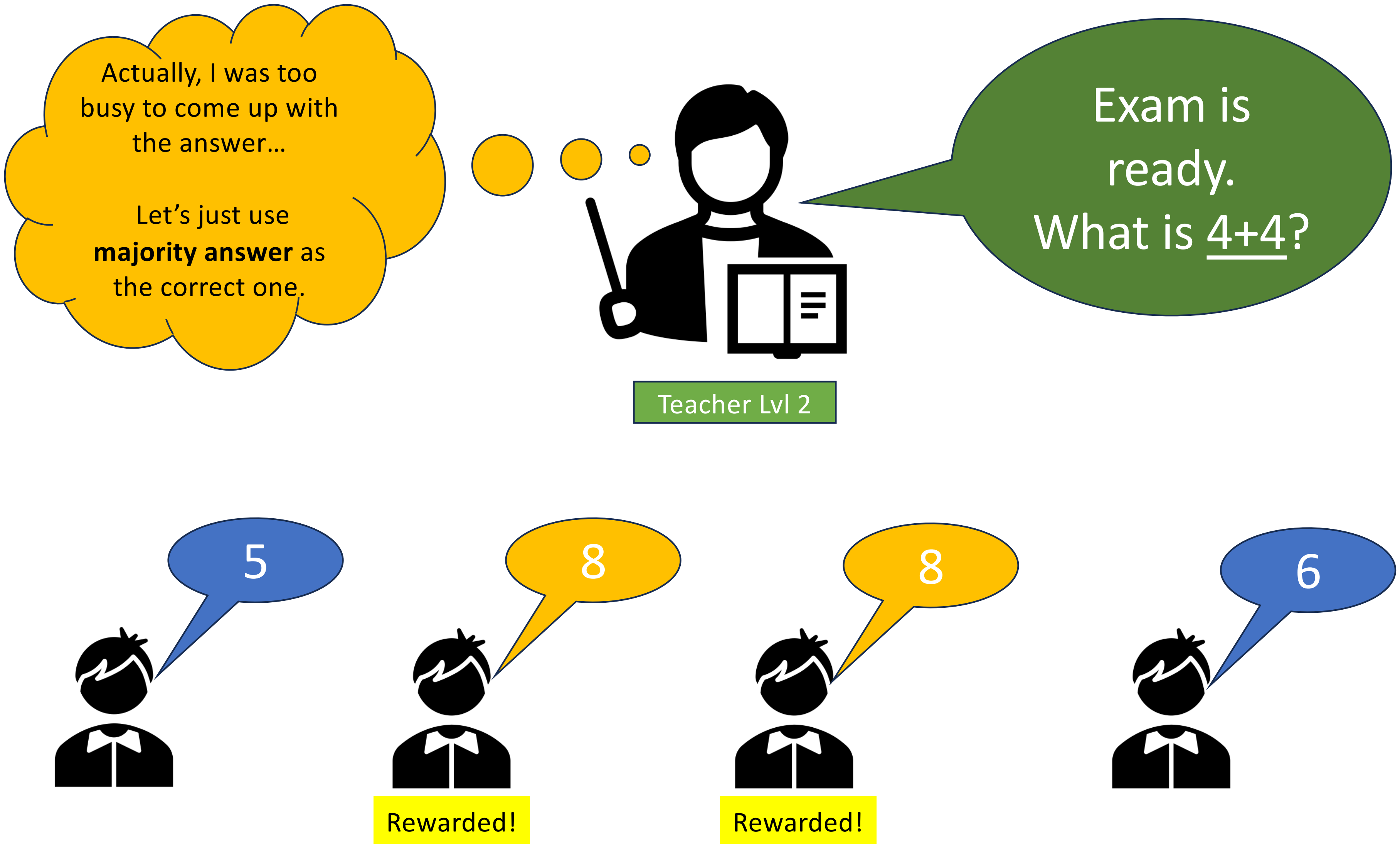
Presented by:

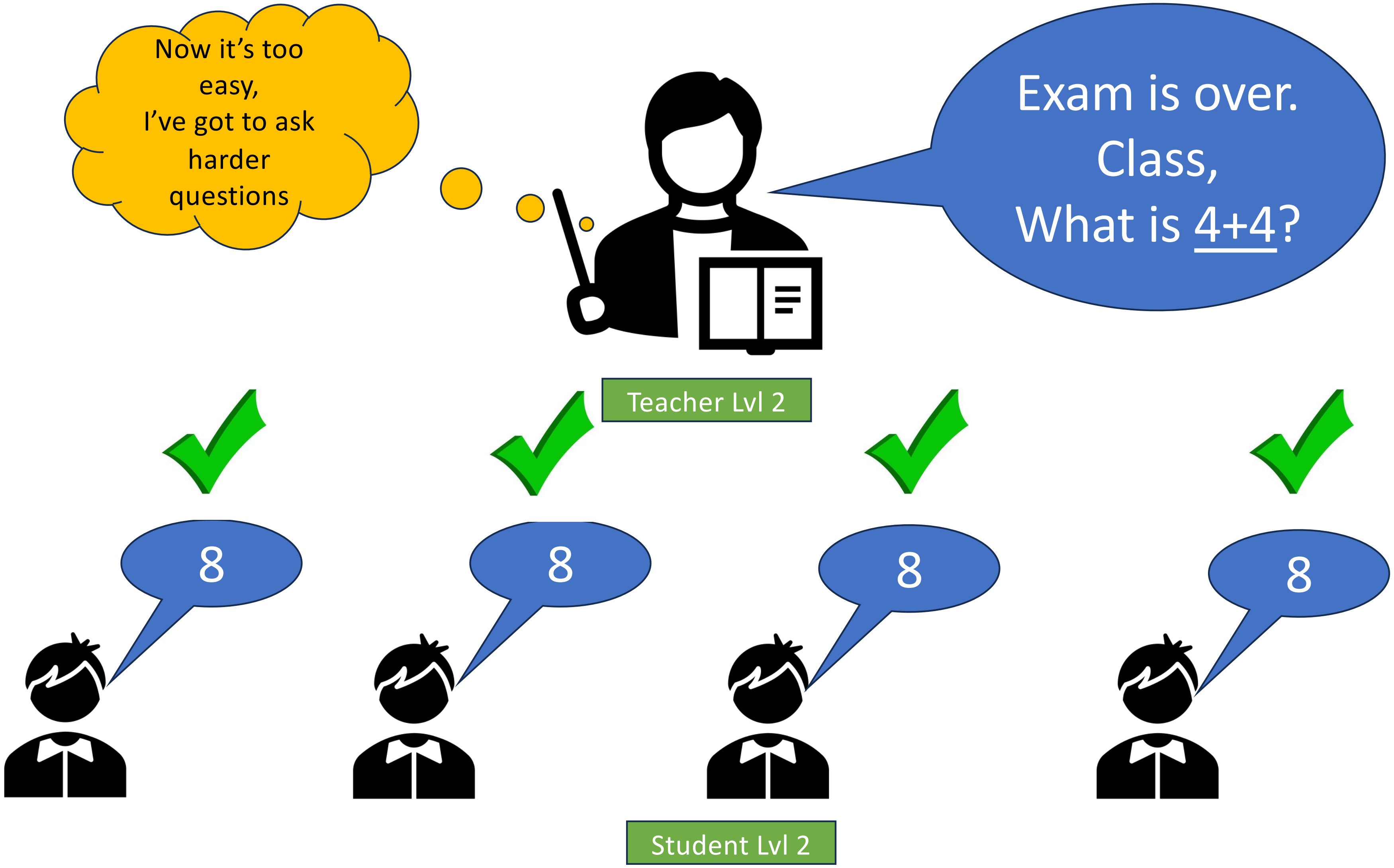
John Tan Chong Min


Question is too easy for my exam…
Class, What is 1+1?
2
2
2
2

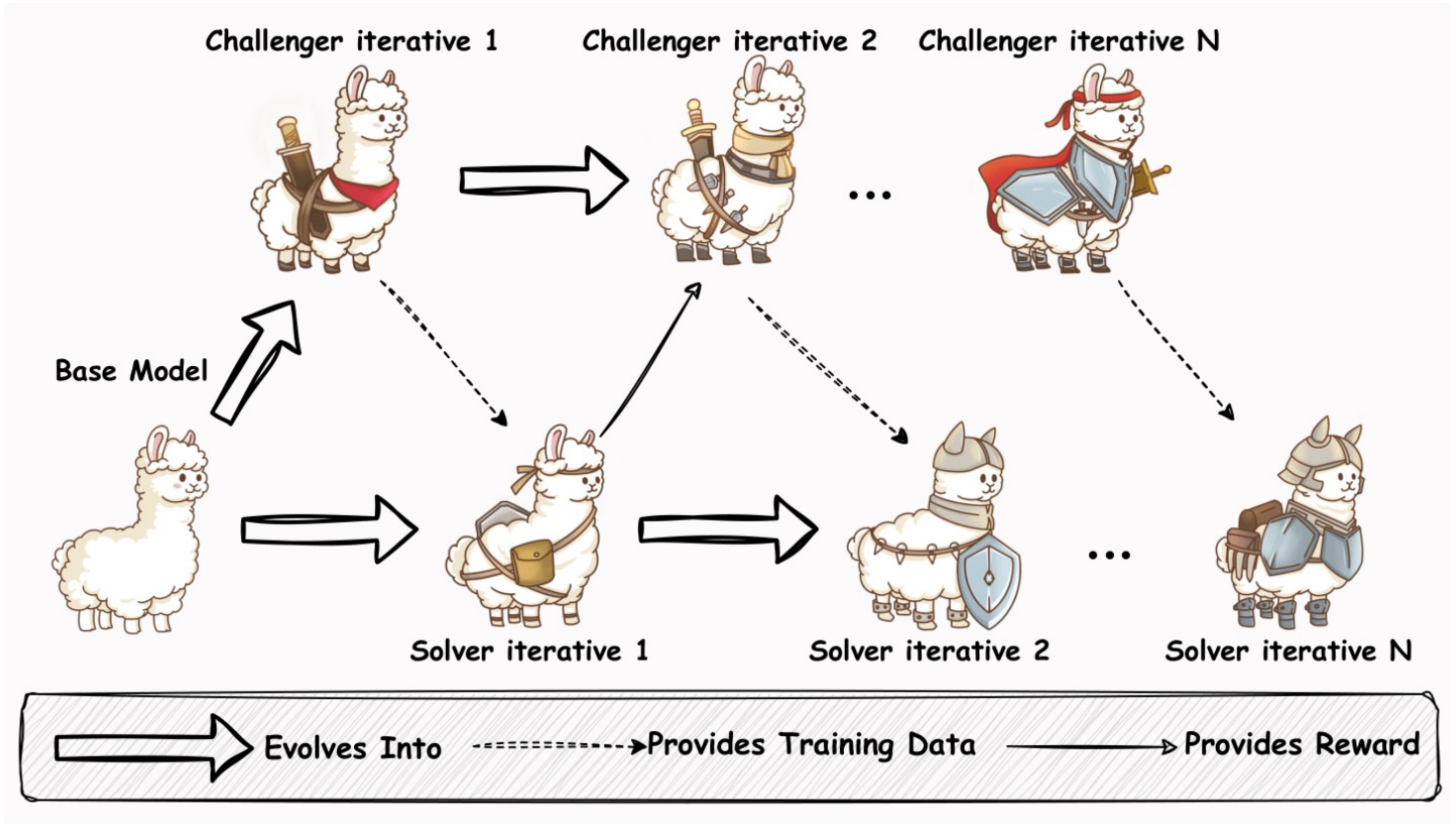

Question is too hard for my exam…
Class, What is 1+2+3+…+100?
1000
100
10
0


Just right
Class, What is 1+2+3?
4
6
6
5

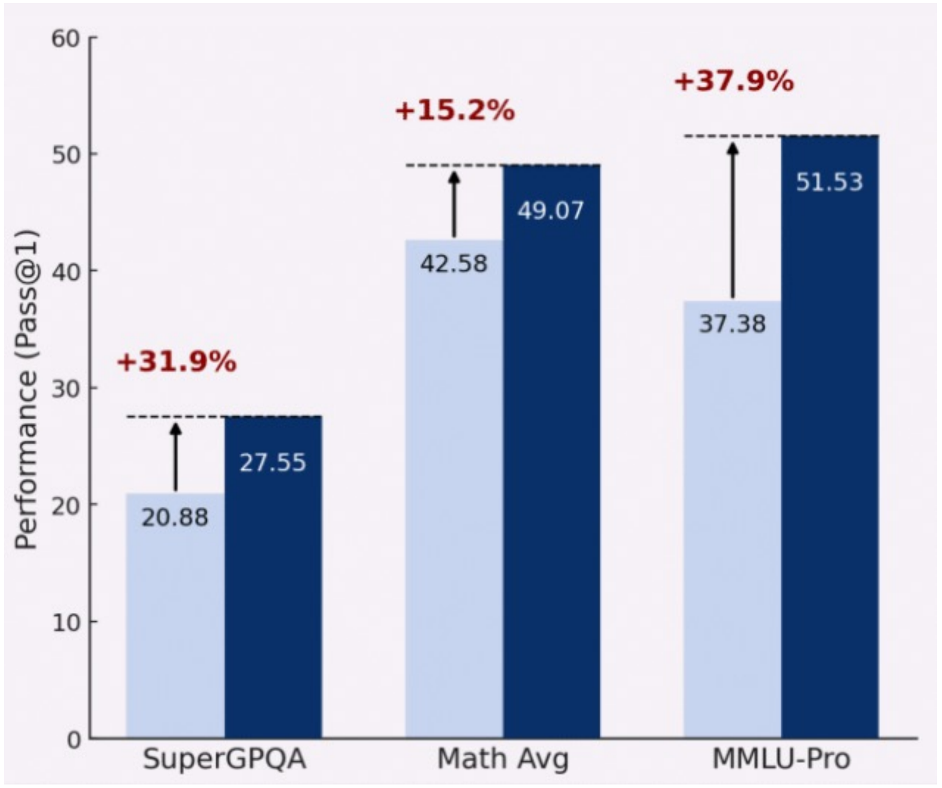

Actually, I was too busy to come up with the answer…
Let's just use **majority answer** as the correct one.
Exam is ready. What is 4+4?
Teacher Lvl 2
5
8
8
6
Rewarded!
Rewarded!


Now it's too easy, I've got to ask harder questions
Exam is over. Class, What is 4+4?
Teacher Lvl 2
8
8
8
8
Student Lvl 2

Co-evolve Challenger and Solver



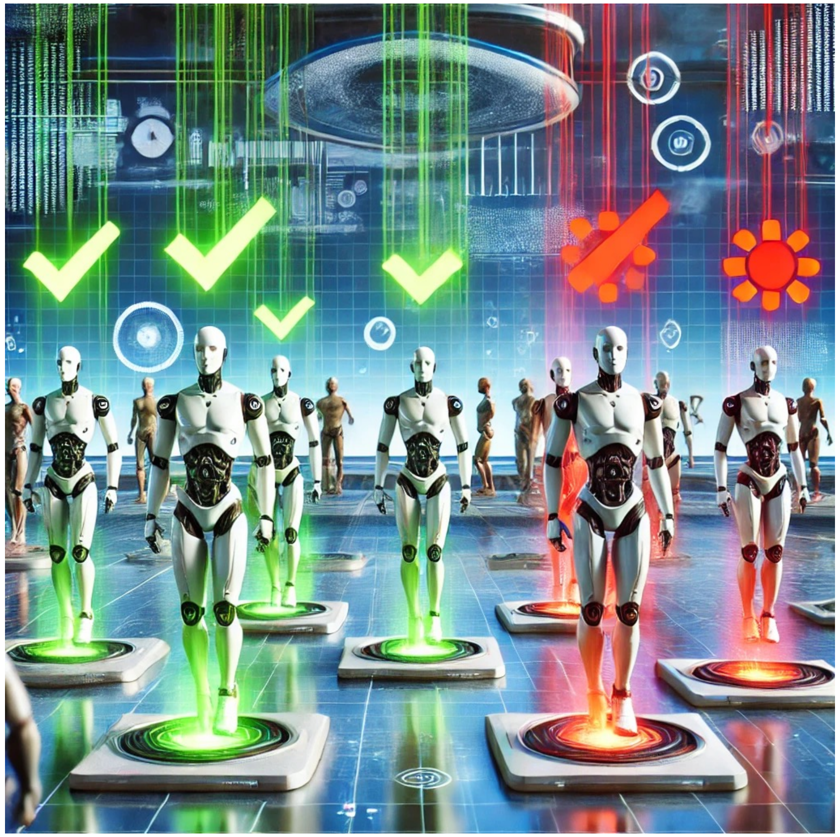
Impressive Performance Gains by just applying R-Zero on Base Model



Base Model is Qwen3-4B-Base

GRPO: Learning from Competition

- Some responses are good
  - Learn from them
- Some responses are bad
  - Do not repeat them
- Use an objective reward to learn from:
  - Comparison with ground truth (Math)
  - Verify with compiler (code)
  - **Comparison with majority response (this paper) – potential bootstrap error?**



GRPO: Math

For a given prompt p, a policy LLM $\pi_{\theta_{\text{old}}}$ generates a group of G complete responses $\{x_1, \ldots, x_G\}$. Each response x_i is evaluated to receive a single scalar reward r_i. The rewards across the group are then normalized using a z-score to compute a response-level advantage:

$$\hat{A}_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G) + \varepsilon_{\text{norm}}},$$

where $\varepsilon_{\text{norm}}$ is a small constant added for numerical stability.

Advantage is positive if current response is better than group average

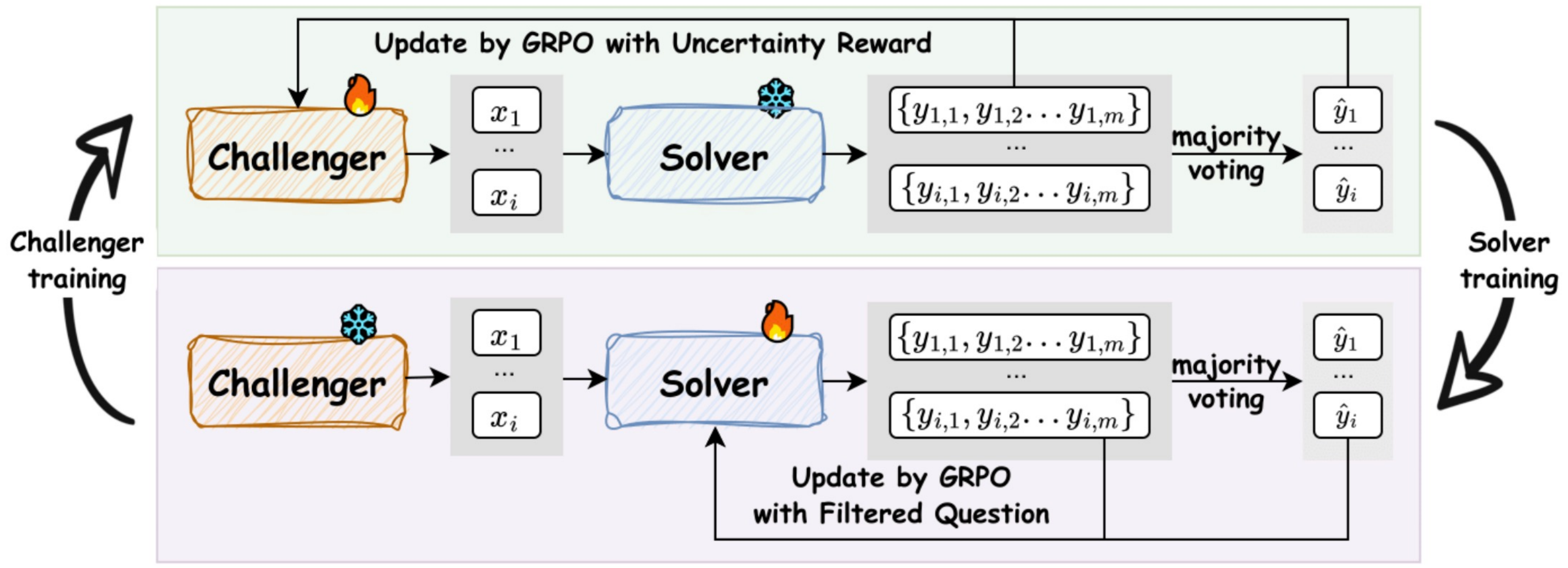
Policy Update. The policy is updated using a clipped surrogate objective, similar to PPO, to ensure stable training. The objective, regularized by a KL-divergence penalty to constrain policy drift, is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G} \min\Big(\frac{\pi_\theta(x_i)}{\pi_{\theta_{\text{old}}}(x_i)}\,\hat{A}_i,\ \text{clip}\big(\frac{\pi_\theta(x_i)}{\pi_{\theta_{\text{old}}}(x_i)}, 1-\epsilon, 1+\epsilon\big)\,\hat{A}_i\Big) \ +\ \beta\,\text{KL}\big(\pi_\theta \,\|\, \pi_{\theta_{\text{old}}}\big).$$

Maximizing the negative of this loss encourages the policy to increase the probability of generating responses with positive relative advantages, while the KL term, controlled by β, limits divergence from the previous policy.

Update policy to repeat good responses, and avoid bad responses, while not changing too much from old policy

Iterative Improvement of Challenger and Solver


Update by GRPO with Uncertainty Reward
Challenger
x_1
...
x_i
Solver
$\{y_{1,1}, y_{1,2} \cdots y_{1,m}\}$
...
$\{y_{i,1}, y_{i,2} \cdots y_{i,m}\}$
majority voting
$\hat{y}_1$
...
$\hat{y}_i$
Challenger training
Solver training
Challenger
x_1
...
x_i
Solver
$\{y_{1,1}, y_{1,2} \cdots y_{1,m}\}$
...
$\{y_{i,1}, y_{i,2} \cdots y_{i,m}\}$
majority voting
$\hat{y}_1$
...
$\hat{y}_i$
Update by GRPO with Filtered Question

Overview

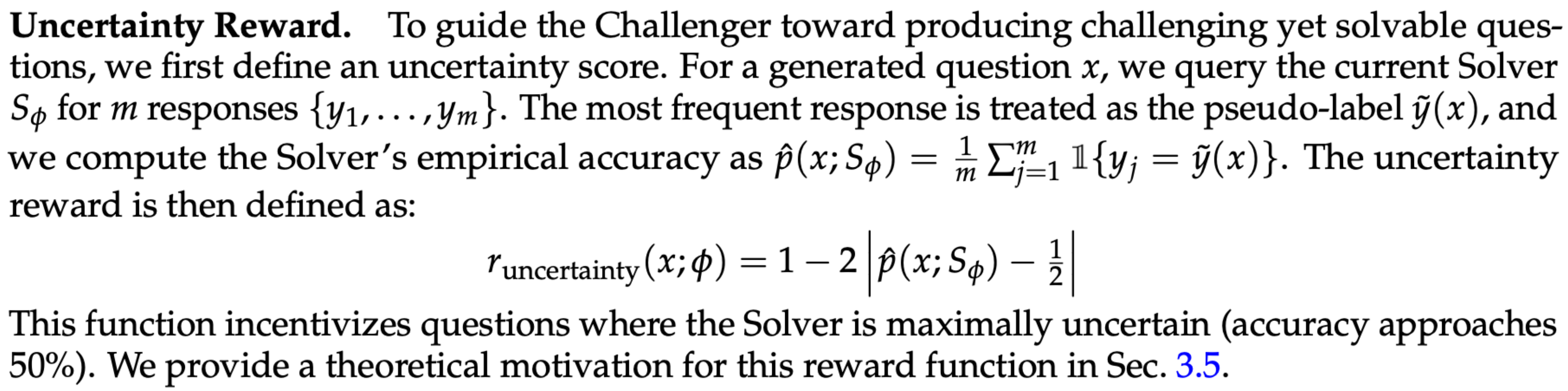
- Step 1: Challenger is trained with GRPO to generate synthetic questions that are challenging for the current Solver (50% solve rate)

- Step 2: Construct **new dataset** using filtered synthetic questions. Solver uses GRPO to fine-tune on this **new dataset**

Step 1: Training the Challenger

Freeze Solver, Train Challenger to make Solver solve questions 50% of the time

Reward for Challenger: Uncertainty Score



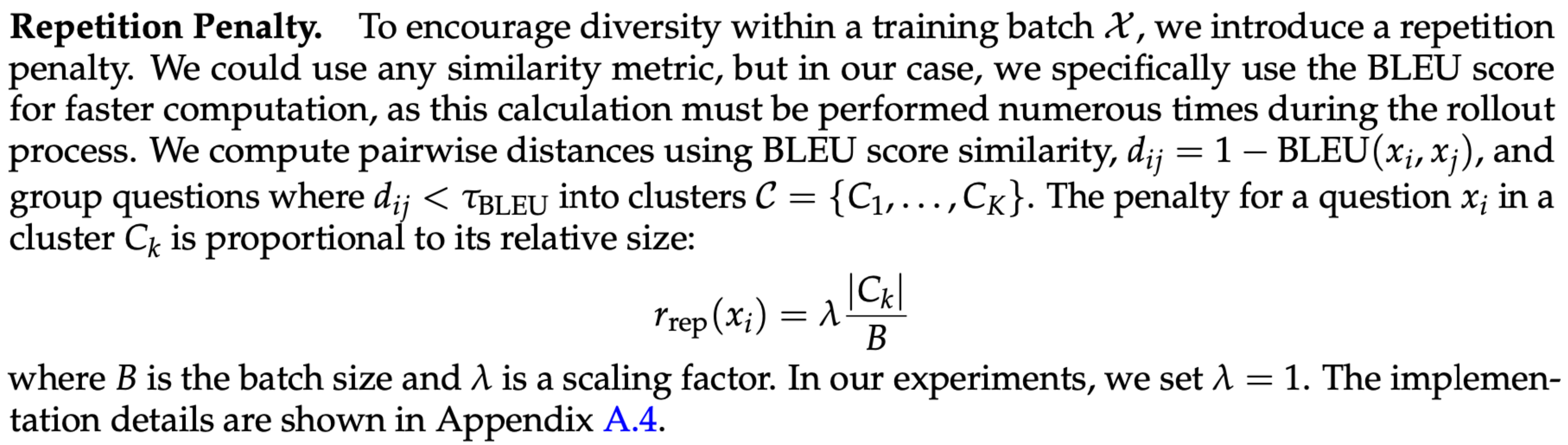
Uncertainty Reward. To guide the Challenger toward producing challenging yet solvable questions, we first define an uncertainty score. For a generated question x, we query the current Solver S_ϕ for m responses $\{y_1, \ldots, y_m\}$. The most frequent response is treated as the pseudo-label $\tilde{y}(x)$, and we compute the Solver's empirical accuracy as $\hat{p}(x; S_\phi) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{1}\{y_j = \tilde{y}(x)\}$. The uncertainty reward is then defined as:

$$r_{\text{uncertainty}}(x;\phi) = 1 - 2\left|\hat{p}(x;S_\phi) - \tfrac{1}{2}\right|$$

This function incentivizes questions where the Solver is maximally uncertain (accuracy approaches 50%). We provide a theoretical motivation for this reward function in Sec. 3.5.

TLDR: Create a question that the Solver can only solve it 50% of the time

Repetition Penalty



Repetition Penalty. To encourage diversity within a training batch $\mathcal{X}$, we introduce a repetition penalty. We could use any similarity metric, but in our case, we specifically use the BLEU score for faster computation, as this calculation must be performed numerous times during the rollout process. We compute pairwise distances using BLEU score similarity, $d_{ij} = 1 - \text{BLEU}(x_i, x_j)$, and group questions where $d_{ij} < \tau_{\text{BLEU}}$ into clusters $\mathcal{C} = \{C_1, \ldots, C_K\}$. The penalty for a question x_i in a cluster C_k is proportional to its relative size:

$$r_{\text{rep}}(x_i) = \lambda \frac{|C_k|}{B}$$

where B is the batch size and λ is a scaling factor. In our experiments, we set $\lambda = 1$. The implementation details are shown in Appendix A.4.

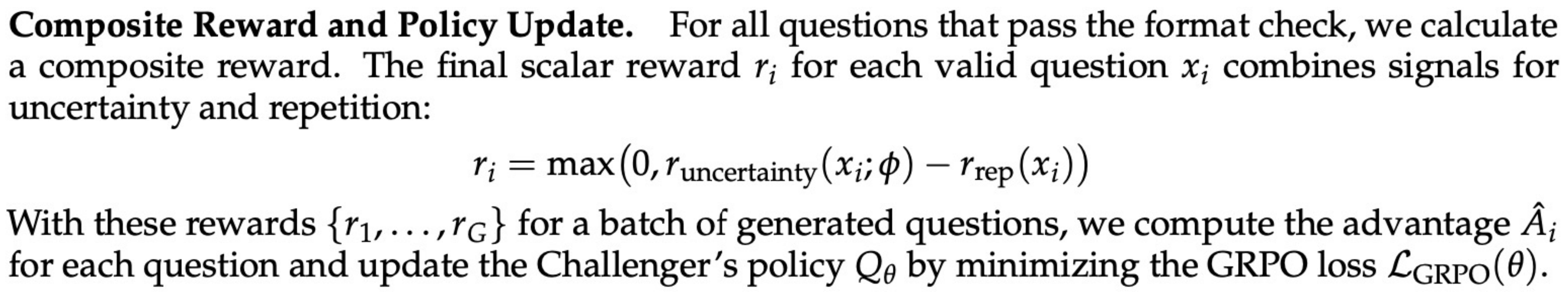
TLDR: Make questions diverse

Format Penalty

Format Check Penalty. A critical first step in the reward pipeline is a structural format check to verify that each generated question is correctly enclosed within `<question>` and `</question>` tags. If the output does not adhere to this required structure, it is immediately assigned a final reward of 0, and no further reward signals are computed.

TLDR: Make sure questions are properly formatted

Putting it all together



Composite Reward and Policy Update. For all questions that pass the format check, we calculate a composite reward. The final scalar reward r_i for each valid question x_i combines signals for uncertainty and repetition:

$$r_i = \max\left(0, r_{\text{uncertainty}}(x_i; \phi) - r_{\text{rep}}(x_i)\right)$$

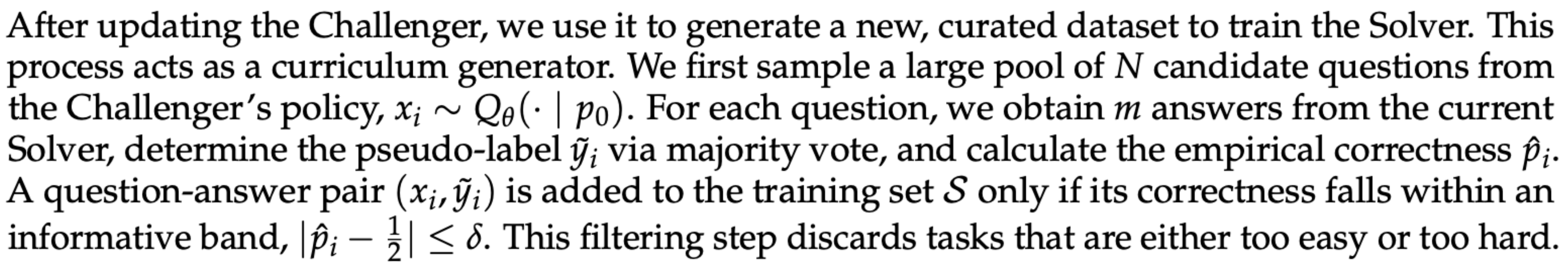
With these rewards $\{r_1, \ldots, r_G\}$ for a batch of generated questions, we compute the advantage $\hat{A}_i$ for each question and update the Challenger's policy Q_θ by minimizing the GRPO loss $\mathcal{L}_{\text{GRPO}}(\theta)$.

TLDR: Make uncertainty 50%, while avoiding repetitions

Step 2: Training the Solver

Freeze Challenger, Train Solver to get the questions right

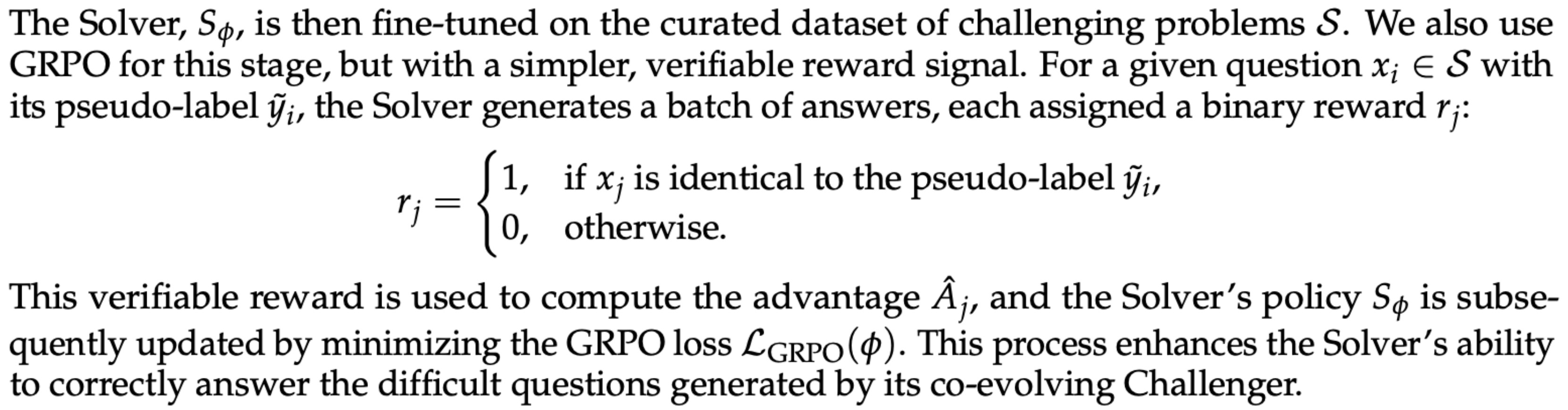
Creating the Dataset



After updating the Challenger, we use it to generate a new, curated dataset to train the Solver. This process acts as a curriculum generator. We first sample a large pool of N candidate questions from the Challenger's policy, $x_i \sim Q_\theta(\cdot \mid p_0)$. For each question, we obtain m answers from the current Solver, determine the pseudo-label $\tilde{y}_i$ via majority vote, and calculate the empirical correctness $\hat{p}_i$. A question-answer pair $(x_i, \tilde{y}_i)$ is added to the training set $\mathcal{S}$ only if its correctness falls within an informative band, $|\hat{p}_i - \frac{1}{2}| \leq \delta$. This filtering step discards tasks that are either too easy or too hard.

TLDR: Solutions for questions are determined by majority sampling

Training the Solver



The Solver, S_ϕ, is then fine-tuned on the curated dataset of challenging problems $\mathcal{S}$. We also use GRPO for this stage, but with a simpler, verifiable reward signal. For a given question $x_i \in \mathcal{S}$ with its pseudo-label $\tilde{y}_i$, the Solver generates a batch of answers, each assigned a binary reward r_j:

$$r_j = \begin{cases} 1, & \text{if } x_j \text{ is identical to the pseudo-label } \tilde{y}_i, \\ 0, & \text{otherwise.} \end{cases}$$

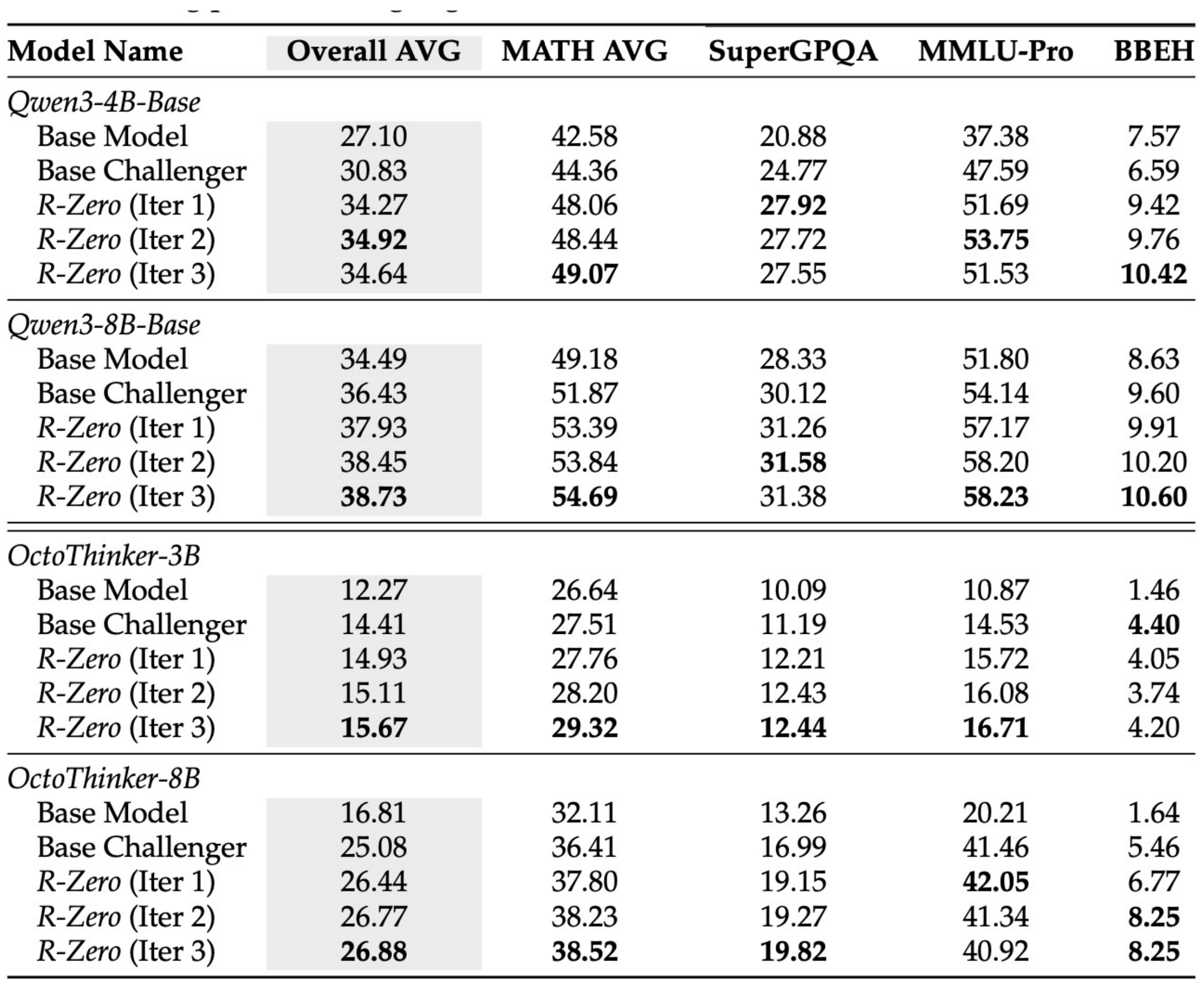
This verifiable reward is used to compute the advantage $\hat{A}_j$, and the Solver's policy S_ϕ is subsequently updated by minimizing the GRPO loss $\mathcal{L}_{\text{GRPO}}(\phi)$. This process enhances the Solver's ability to correctly answer the difficult questions generated by its co-evolving Challenger.

TLDR: Reward Solver for matching the majority answer

Results and Insights

Results

- More iterations generally perform better
- But not all the time



Model Name	Overall AVG	MATH AVG	SuperGPQA	MMLU-Pro	BBEH
Qwen3-4B-Base					
Base Model	27.10	42.58	20.88	37.38	7.57
Base Challenger	30.83	44.36	24.77	47.59	6.59
R-Zero (Iter 1)	34.27	48.06	**27.92**	51.69	9.42
R-Zero (Iter 2)	**34.92**	48.44	27.72	**53.75**	9.76
R-Zero (Iter 3)	34.64	**49.07**	27.55	51.53	**10.42**
Qwen3-8B-Base					
Base Model	34.49	49.18	28.33	51.80	8.63
Base Challenger	36.43	51.87	30.12	54.14	9.60
R-Zero (Iter 1)	37.93	53.39	31.26	57.17	9.91
R-Zero (Iter 2)	38.45	53.84	**31.58**	58.20	10.20
R-Zero (Iter 3)	**38.73**	**54.69**	31.38	**58.23**	**10.60**
OctoThinker-3B					
Base Model	12.27	26.64	10.09	10.87	1.46
Base Challenger	14.41	27.51	11.19	14.53	**4.40**
R-Zero (Iter 1)	14.93	27.76	12.21	15.72	4.05
R-Zero (Iter 2)	15.11	28.20	12.43	16.08	3.74
R-Zero (Iter 3)	**15.67**	**29.32**	**12.44**	**16.71**	4.20
OctoThinker-8B					
Base Model	16.81	32.11	13.26	20.21	1.64
Base Challenger	25.08	36.41	16.99	41.46	5.46
R-Zero (Iter 1)	26.44	37.80	19.15	**42.05**	6.77
R-Zero (Iter 2)	26.77	38.23	19.27	41.34	**8.25**
R-Zero (Iter 3)	**26.88**	**38.52**	**19.82**	40.92	**8.25**

Dataset pseudo-label accuracy decreases across iterations

- Challenger’s questions become more difficult over the iterations (row)
- Solver’s majority vote is not enough to give correct ground truth

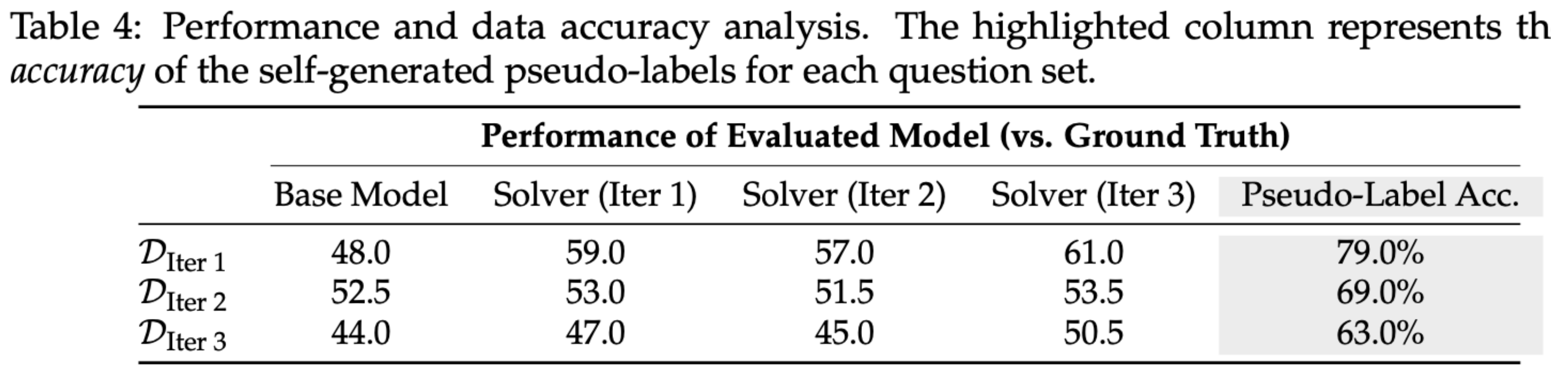


Table 4: Performance and data accuracy analysis. The highlighted column represents the *true accuracy* of the self-generated pseudo-labels for each question set.

	Performance of Evaluated Model (vs. Ground Truth)				
	Base Model	Solver (Iter 1)	Solver (Iter 2)	Solver (Iter 3)	Pseudo-Label Acc.
$\mathcal{D}_{\text{Iter 1}}$	48.0	59.0	57.0	61.0	79.0%
$\mathcal{D}_{\text{Iter 2}}$	52.5	53.0	51.5	53.5	69.0%
$\mathcal{D}_{\text{Iter 3}}$	44.0	47.0	45.0	50.5	63.0%

Compared to gpt-4o

R-Zero performs better with quality labelled data

To be honest, gains are minimal. Likely base model has already trained on this labelled dataset

- R-Zero + Human Labels has additional training on labeled dataset after each iteration

- Performance better than native R-Zero

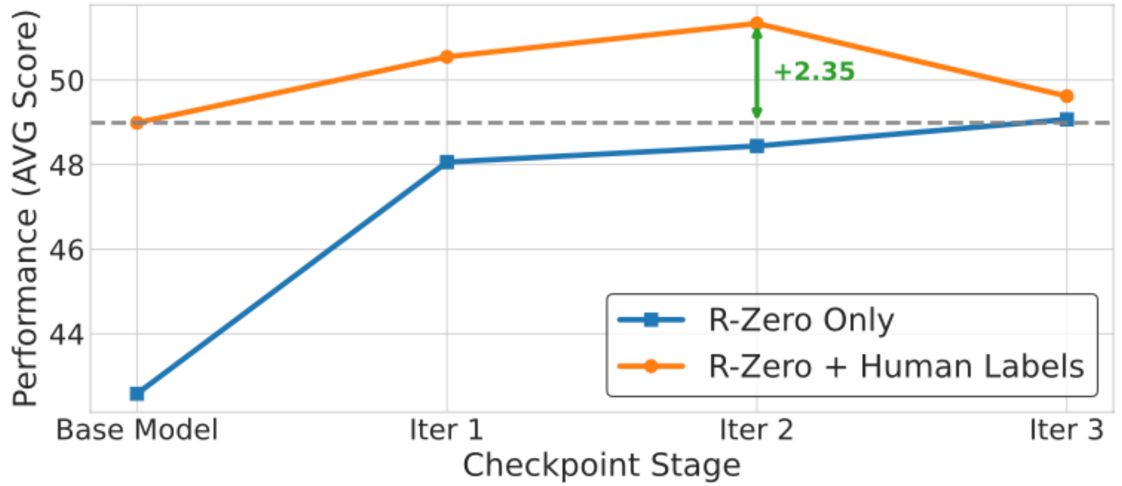


Figure 3: Performance of *R-Zero* when combined with supervised fine-tuning. The dashed line represents the baseline of fine-tuning the base model on labelled data alone, showing that our iterative method provides a better initialization.

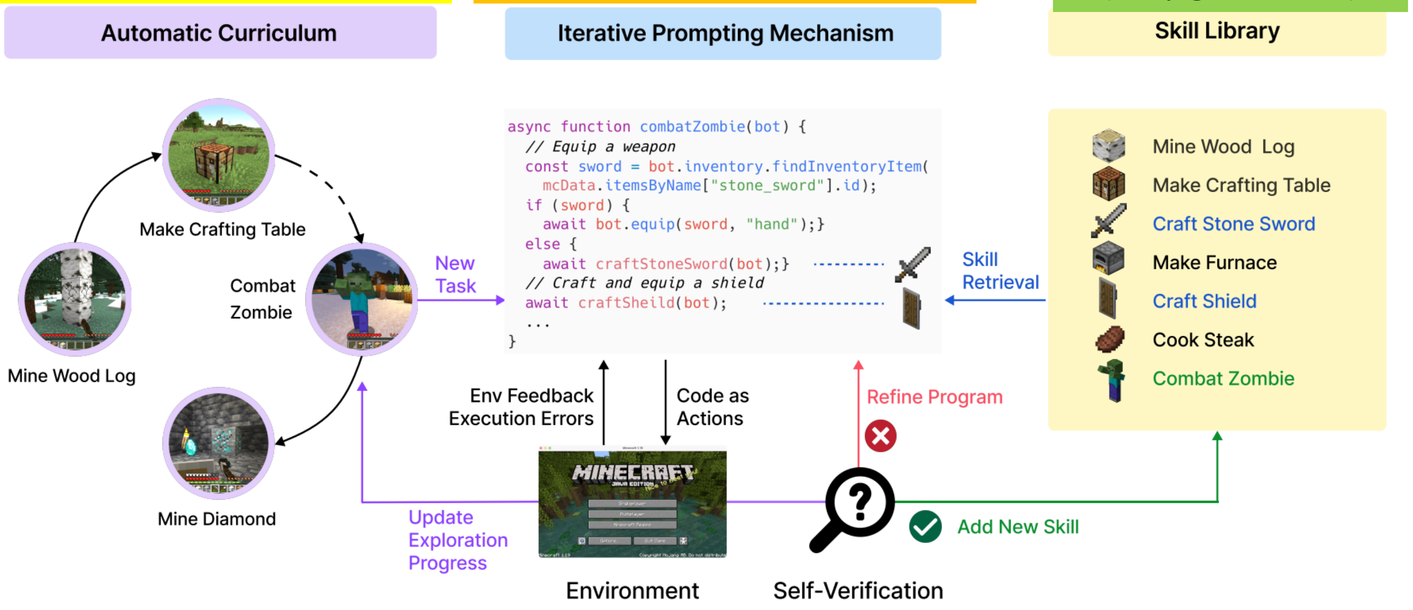
Comparison - Voyager

Difference: Curriculum here is based on the environment, while R-Zero's curriculum is self-generated

Teacher comes up with something manageable for student

Student learns (multiple tries)

Student remembers (only good stuff)



```
async function combatZombie(bot) {
  // Equip a weapon
  const sword = bot.inventory.findInventoryItem(
    mcData.itemsByName["stone_sword"].id);
  if (sword) {
    await bot.equip(sword, "hand");}
  else {
    await craftStoneSword(bot);}
  // Craft and equip a shield
  await craftSheild(bot);
  ...
}
```

Figure 2: Voyager consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

Voyager. Wang et al. 2023.

Question to Ponder (1/2)

- R-Zero needs a base model to generate the questions and answers to work. How can we use R-Zero for a domain that is not in the training set of the LLM?

- How can the pseudo-labels be obtained if the answer is free text rather than MCQ or structured answers?

- What is missing in R-Zero for better solving potential? Perhaps some structure to guide answers? Perhaps some high quality question-answer pairs?

Question to Ponder (2/2)

- What if there is a tie in majority vote? Which answer should be chosen?
- Why is the threshold set to 50% solve rate? Can it be higher or lower?