Reform reading group

Date: 2025-04-23

Binghui Peng

Content

Solving math word problems with process- and outcome-based feedback
http://arxiv.org/abs/2211.14275

Let's Verify Step by Step

https://arxiv.org/abs/2305.20050

DeepMind

2022-11-28

Solving math word problems with process- and outcome-based feedback

Jonathan Uesato[1*], Nate Kushman[1*], Ramana Kumar[1*], Francis Song[1], Noah Siegel[1], Lisa Wang[1], Antonia Creswell[1], Geoffrey Irving[1] and Irina Higgins[1]

[1]DeepMind, [*]Equal contributions

Summary

1. Extensive experiments over GSM8k dataset, comparing

few-shot learning/fine-tuning $\times$ majority vote/reward model (PRM/ORM) $\times$ RL

2. RL (expert iteration) $\approx$ SFT >> few shot (if you only care about error rate)

   PRM $\approx$ ORM > final answer (if you care about trace error)

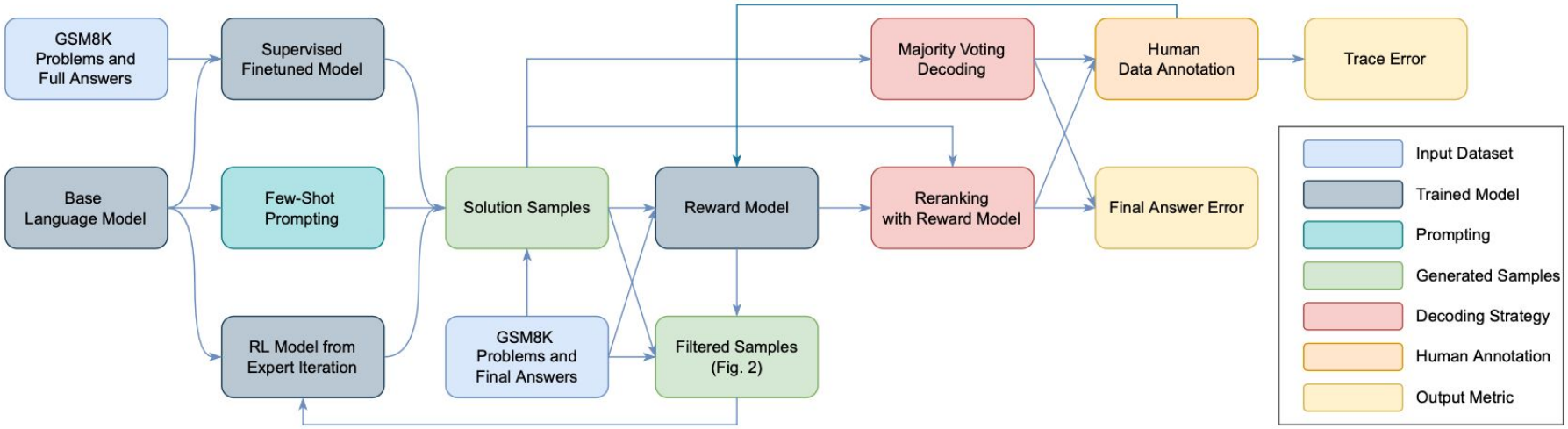


Figure 1 | **Method Overview.** This schematic provides an overview of the various modeling and training components considered and how they fit together. Some details (covered in the text) are omitted for readability.

	Approach	Base model	Error rate (%) Trace	Final-answer
	Few-shot (Wang et al., 2022; Wei et al., 2022)	PaLM-540B	14.0	25.6
	Few-shot (Lewkowycz et al., 2022)	Minerva-540B	-	21.5
	Few-shot+Final-Answer RL (Zelikman, 2022)	GPT-J-6B	-	89.3
	Few-shot, ORM reranking (Li et al., 2022)	Codex-175B	-	16.8
	Zero-shot (Kojima et al., 2022)	InstructGPT-175B	-	59.3
	SFT, ORM reranking (Cobbe et al., 2021)	GPT-175B	-	45.0
3.1	Few-shot, Majority Voting	Our Base-70B	-	41.5
3.1	Few-shot+Final-Answer RL, Majority Voting	Our Base-70B	19.8 (7.9-31.7)	23.5
3.1	SFT+Final-Answer RL, Majority Voting	Our Base-70B	12.1 (4.6-19.6)	20.2
3.1	SFT, Majority Voting	Our Base-70B	11.4 (4.8-18.0)	22.3
3.2	Few-shot, ORM reranking	Our Base-70B	-	27.8
3.2	Few-shot+Final-Answer RL, ORM reranking	Our Base-70B	12.4 (2.1-22.8)	16.6
3.2	SFT+Final-Answer RL, ORM reranking	Our Base-70B	3.7 (0.5-6.9)	14.2
3.2	SFT, ORM reranking	Our Base-70B	4.4 (0.6-8.3)	14.8
3.2	SFT, PRM reranking	Our Base-70B	3.5 (0.5-6.5)	14.1
3.3	Few-shot+ORM-RL, ORM reranking	Our Base-70B	5.5 (2.6-8.4)	13.8
3.3	SFT+ORM-RL, ORM reranking	Our Base-70B	3.4 (0.0-6.8)	12.7
3.3	SFT+PRM-RL, PRM reranking	Our Base-70B	3.8 (0.5-7.1)	12.9

Preliminary

Dataset: GSM8k

Metric: error rate, trace error rate (correct answer & incorrect reasoning)

Method:

- Few shot / SFT
- Reward model: ORM or PRM
- RL: supervised via expert iteration (policy improvement + distillation)

SFT

2.3. Supervised finetuning

In supervised finetuning (SFT), we finetune an LM to maximize the log-likelihood of a sequence of target tokens, given a sequence of input tokens. In our paper, we use SFT as a process-based approach by taking the reasoning traces provided in the GSM8K dataset as the target tokens (as opposed to the outcome-based approach of using only the final answer as the target), with the problem statement as the input tokens.

Training details We finetune using AdamW (Loshchilov and Hutter, 2017) with a learning rate of 2×10^{-6} and a batch size of 256. We stop finetuning once the language modeling loss begins to increase on the validation set. For our SFT model, this happens after 70 steps, amounting to slightly more than 2 training set epochs.

Reward model

2.4. Reward models

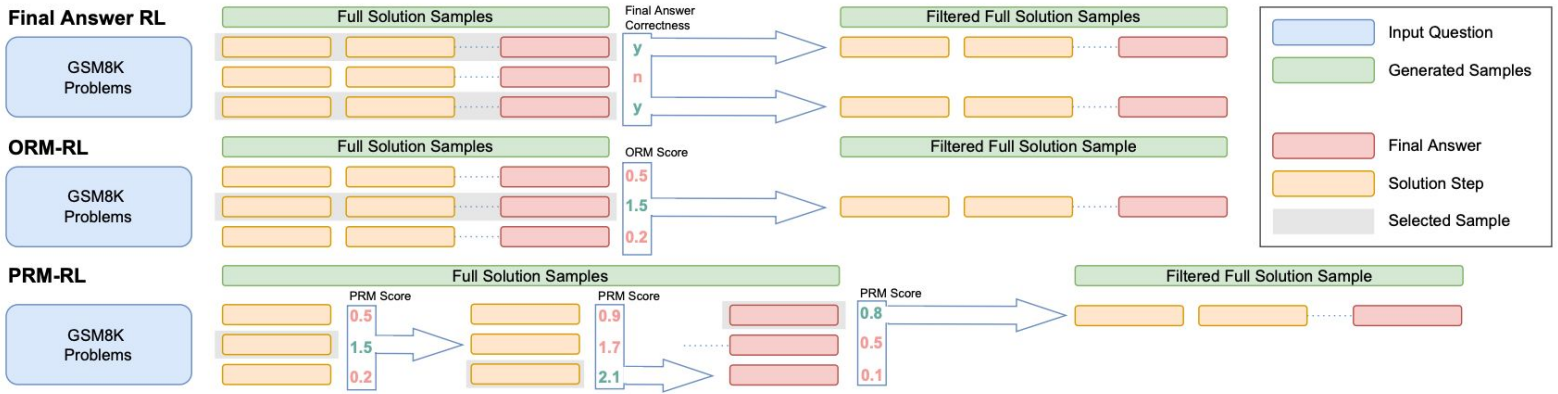
We evaluate two main approaches to training reward models (RMs) (Christiano et al., 2017; Ziegler et al., 2019; Menick et al., 2022), also known as verifiers (Cobbe et al., 2021). In both approaches, we implement the RM as a LM, trained to predict a binary label as either a 'correct' or 'incorrect' token after each step. In the *outcome-supervised RM* (ORM), the binary label for each step indicates whether the resulting final answer of that full sample matched the reference final answer, as proposed by Cobbe et al. (2021). A policy which maximizes the ORM score at each step thus maximizes the RM-estimated probability at each step of eventually reaching the correct final answer. For the *process-supervised RM* (PRM), the binary label after each step indicates whether the steps so far are correct. Because we lack reliable programmatic means for determining the correctness of intermediate steps, we use human annotations for these labels, as described in Section 2.7. A policy which maximizes the PRM score thus selects each step to maximize the RM-estimated probability of the steps so far being correct. If the steps so far are correct, this typically means such a policy minimizes the probability of introducing a mistake on the current step. As reported in Section 3.2, we find this outperforms the approach from Li et al. (2022), which is similar to our PRM but replaces human evaluations with a heuristic based on string matching the results of the intermediate calculations.

Decoding

Best of N sampling (with reward model) or majority vote

We use two approaches to select the best sample. When no RM is available, we use *majority voting*. For this, we first select the most common final answer from the K samples, then select a random sample from among those yielding this selected final answer. This is called self-consistency by Wang et al. (2022), and is similar to more general techniques like Minimum Bayes Risk decoding (Kumar and Byrne, 2004). Otherwise, we use *RM-weighted* decoding, also called verifier-voting by Li et al. (2022). Here, we weight each sample according to the RM-estimated correctness probability, select the final answer with the largest total weight, and then select the sample with the highest RM score from those yielding the selected final answer. More formally, we select the final answer $f^* = \arg\max_f \sum_{y_i:\text{final\_ans}(y_i)=f} \text{rm\_prob}(y_i)$, where $y_1, \ldots, y_K$ are the model samples, then select the

RL


Final Answer RL
GSM8K Problems
Full Solution Samples
Final Answer Correctness
y
n
y
Filtered Full Solution Samples
ORM-RL
GSM8K Problems
Full Solution Samples
ORM Score
0.5
1.5
0.2
Filtered Full Solution Sample
PRM-RL
GSM8K Problems
Full Solution Samples
PRM Score
0.5
1.5
0.2
PRM Score
0.9
1.7
2.1
PRM Score
0.8
0.5
0.1
Filtered Full Solution Sample
Input Question
Generated Samples
Final Answer
Solution Step
Selected Sample

Main result 1: Error rate

Supervising final-answer correctness alone suffices for low final-answer error rate. The SFT and Few-shot+Final-Answer RL models attain similar final-answer error rates both without an RM (22.3% vs. 23.5%) and with an ORM (14.8% vs. 16.6%). This is notable, as Few-shot+Final-Answer RL only requires demonstrators to provide a final answer, rather than a full reasoning trace. Put another way, Few-shot+Final-Answer RL uses 1-4 tokens of label supervision per question, while SFT uses hundreds. This suggests that in cases where final-answer correctness is sufficient, outcome-based approaches can provide a label-efficient approach with competitive performance.

Main result 2: process reward

ORM-supervised reward models approximate PRM labels. Despite the fact that ORMs are only trained to predict whether the final answer is correct, we can see in Fig. 4 that ORM predictions tend to agree more with the PRM labels than with the ORM labels themselves (85% vs. 77% averaged over all steps).[1] We suspect this is because it is simpler for the ORM to learn to recognize when steps are correct, than it is to check the answer by internally computing the final answer itself. This is further supported by that fact that, even though trace error is measured only on samples with the correct final answer, RM reranking significantly improves trace error relative to SFT alone (4.4% vs. 11.4%). This suggests that RMs are checking the reasoning steps, and not just the final answers. However, we caution against over-generalizing: the fact that the ORM model approximates the PRM labels may be

Main result 3: trace error

Low trace error requires either process-based feedback or a reward model that emulates it

Fig. 3 shows that despite similar final-answer error rates, there is a significantly higher trace error rate for the outcome-based Few-shot+Final-Answer RL vs. the process-based SFT model (19.8% vs. 11.4%). This discrepancy persists with RM reranking: Few-shot+Final-Answer RL with ORM reranking underperforms SFT with ORM/PRM reranking (12.4% vs. 4.4%/3.5%). However, we find that when we train the few-shot RL model using an ORM (Few-shot+ORM-RL) rather than training directly against final-answer correctness, the trace error drops significantly from 12.4% to 5.5% closing much of this gap. We believe this results from the previous finding, i.e. that the ORM is basically learning to emulate the PRM allowing the model to learn from emulated process-based

LET'S VERIFY STEP BY STEP

Hunter Lightman*, **Vineet Kosaraju***, **Yura Burda***, **Harri Edwards**, **Bowen Baker**, **Teddy Lee**, **Jan Leike**, **John Schulman**, **Ilya Sutskever** & **Karl Cobbe***
OpenAI
San Francisco, CA, USA
karl@openai.com

Summary

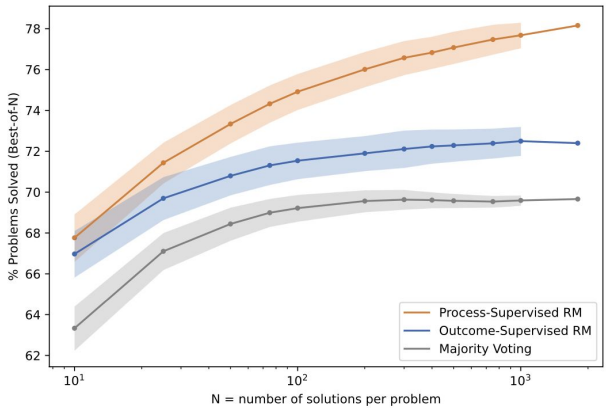
PRM out-performs ORM over MATH dataset, at large scale (large annotation) with more capable base model

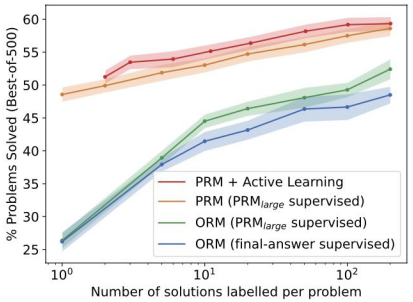
Despite these advantages, Uesato et al. (2022) found that outcome supervision and process supervision led to similar final performance in the domain of grade school math. We conduct our own detailed comparison of outcome and process supervision, with three main differences: we use a more capable base model, we use significantly more human feedback, and we train and test on the more challenging MATH dataset (Hendrycks et al., 2021).

Two scales

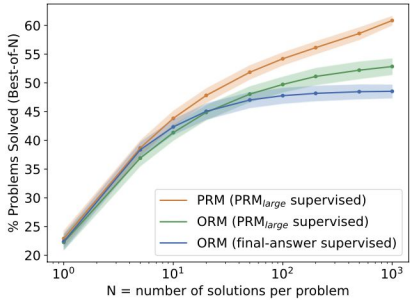
Large scale: Use human annotation data

	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	**78.2**	69.6





(a) Four series of reward models trained using different data collection strategies, compared across training sets of varying sizes.



(b) Three reward models trained on 200 samples/problem using different forms of supervision, compared across many test-time compute budgets.

Small scale: Use PRM_{large} as annotation, oblation study

Data annotation

During data collection, we must decide which solutions to surface to data-labelers. The most straightforward strategy is to uniformly surface solutions produced by the generator. However, if we surface solutions that make obvious errors, the human feedback we get is less valuable. We would prefer to surface solutions that are more likely to fool our best reward model. To that end, we attempt to strategically select which solutions to show data-labelers. Specifically, we choose to surface *convincing wrong-answer* solutions. We use the term *convincing* to refer to solutions that are rated highly by our current best PRM, and we use *wrong-answer* to refer to solutions that reach an incorrect final answer. We use this slightly verbose phrasing to emphasize the fact that correctness is determined solely by checking the final answer, a process which occasionally leads to misgraded solutions. We expect to gain more information from labeling convincing wrong-answer solutions, since we know the PRM is mistaken about at least one step in each such solution.

When we provide process supervision, we deliberately choose to supervise only up to the first incorrect step. This makes the comparison between outcome and process supervision more straightforward. For correct solutions, both methods provide the same information, namely that every step is correct. For incorrect solutions, both methods reveal the existence of at least one mistake, and process supervision additionally reveals the precise location of that mistake. If we were to provide additional process supervision beyond the first mistake, then process supervision would have an even greater information advantage. This decision also keeps the labelling cost similar for humans: without relying on an easy-to-check final answer, determining the correctness of a solution is equivalent to identifying its first mistake. While most MATH problems do have easy-to-check final answers, we expect this to not remain true in more complex domains.

ORM vs. PRM

2.5 Outcome-supervised Reward Models (ORMs)

We train ORMs following a similar methodology to Cobbe et al. (2021). We uniformly sample a fixed number of solutions per problem from the generator, and we train the ORM to predict whether each solution is correct or incorrect. In practice, we usually determine correctness by automatically checking the final answer, but in principle these labels could be provided by humans. At test time, we use the ORM's prediction at the final token as the overall score for the solution. We note the automatic grading used to determine ORM targets is not perfectly reliable: *false positives* solutions that reach the correct answer with incorrect reasoning will be misgraded. We discuss additional ORM training details in Appendix E.

2.6 Process-supervised Reward Models (PRMs)

We train PRMs to predict the correctness of each step after the last token in each step. This prediction takes the form of a single token, and we maximize the log-likelihood of these target tokens during training. The PRM can therefore be trained in a standard language model pipeline without any special accommodations. To determine the step-level predictions at test time, it suffices to perform a single PRM forward pass over the whole solution. We visualize large-scale PRM scores for two different solutions in Figure 2. To compare multiple solutions, it is necessary to compute a single score for each solution. This is an important but straightforward detail: we define the PRM score for a solution to be the probability that every step is correct under the PRM. We implement this as the product of the correctness probabilities for each step. We describe other possible scoring strategies and additional PRM training details in Appendix F.

Large scale supervision

	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	**78.2**	69.6

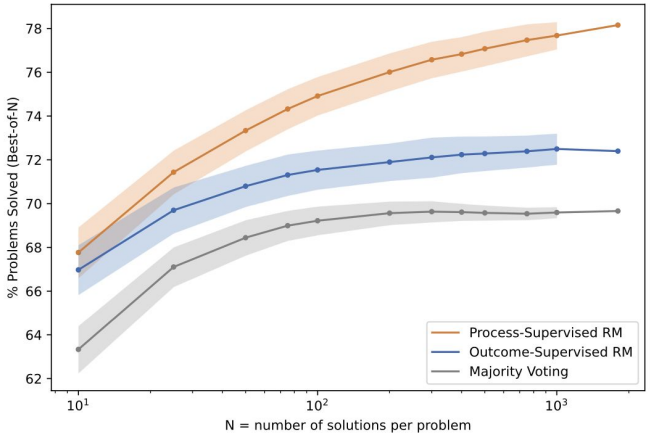
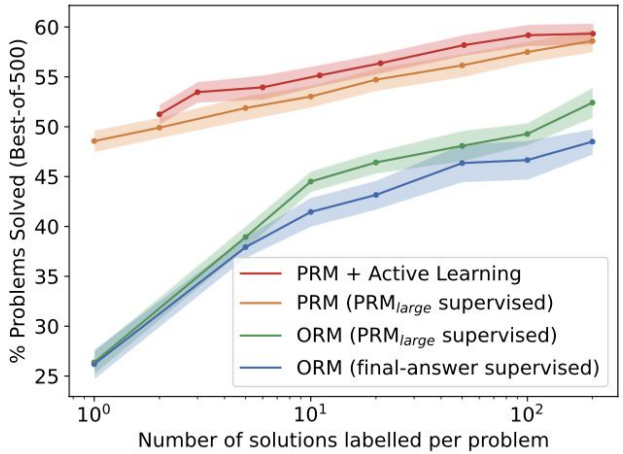
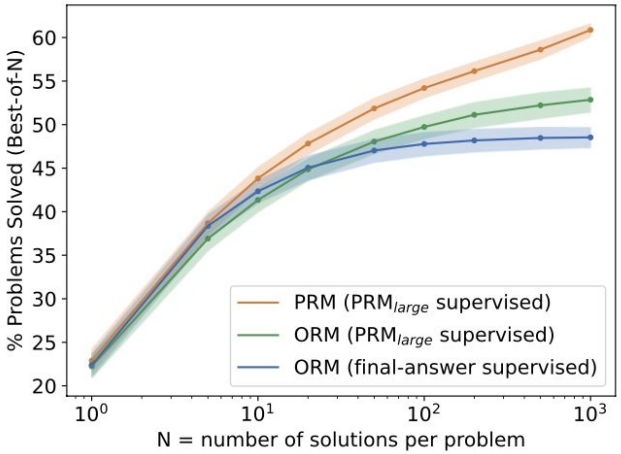


Figure 3: A comparison of outcome-supervised and process-supervised reward models, evaluated by their ability to search over many test solutions. Majority voting is shown as a strong baseline. For $N \leq 1000$, we visualize the variance across many subsamples of the 1860 solutions we generated in total per problem.

Small scale synthetic supervision



(a) Four series of reward models trained using different data collection strategies, compared across training sets of varying sizes.



(b) Three reward models trained on 200 samples/problem using different forms of supervision, compared across many test-time compute budgets.

Figure 4: A comparison of different forms of outcome and process supervision. Mean and standard deviation is shown across three seeds.