

# Discriminator-Guided Multi-step Reasoning with Language Models

Muhammad Khalifa\*, Lajanugen Logeswaran<sup>†</sup>, Moontae Lee<sup>†‡</sup>,  
Honglak Lee<sup>\*†</sup>, Lu Wang\*

University of Michigan\*, LG AI Research<sup>†</sup>, University of Illinois at Chicago<sup>‡</sup>

## Abstract

In the context of multi-step reasoning, language models (LMs) probabilities are often miscalibrated—solutions with high probabilities are not always correct. Therefore, greedy decoding, which is the standard decoding method for reasoning tasks, often yields incorrect solutions. In addition, methods such as self-consistency and verifiers rely on sampling from the LM distribution and do not tackle the underlying issue. To address this, we introduce **Guiding Multi-step Reasoning with a Correctness Discriminator (GRACE)**, a stepwise decoding approach that nudges the model towards producing correct reasoning steps. GRACE employs a discriminator model, which is trained to differentiate correct steps from invalid ones, to adjust decoding preferences based on the correctness of each reasoning step. Importantly, GRACE does not require fine-tuning or re-training the LMs. When compared with conventional decoding strategies over four popular math reasoning benchmarks, GRACE exhibits significant improvements in both final answer accuracy and step correctness, outperforming both greedy decoding and self-consistency.<sup>1</sup>

## 1 Introduction

Multi-step reasoning spans a set of tasks where a question is answered via a sequence of reasoning steps until a final answer is reached (Creswell and Shanahan, 2022; Wei et al., 2022b). While pre-trained language models (LMs) have shown impressive performance on a variety of QA tasks, they still struggle with problems that require complex multi-step reasoning (Cobbe et al., 2021; Creswell et al., 2022; Ni et al., 2023). One reason is that the next-word prediction objective used for pre-training does not explicitly encourage the LM toward correct step-by-step reasoning. To boost the

Question	Fred had 212 sheets of paper. He received another 307 sheets of paper from Jane and gave Charles 156 sheets of paper. How many sheets of paper does Fred have left?	Log Prob
	A total of $212 + 307 + 156 = 673$ sheets of paper. Fred has $673 - 156 = 297$ sheets of paper left. ✗	-0.17
	Fred has $212 + 307 = 507$ sheets of paper. Fred gave Charles $156 = 156$ sheets of paper. Fred has $507 - 156 = 429$ sheets of paper left. ✗	-0.29
	Fred has $212 + 156 = 368$ sheets of paper. After giving Charles 156 sheets of paper, Fred has $368 - 156 = 212$ sheets of paper left. ✗	-0.71
	He had 212 sheets and received 307 more for a total of $212+307 = 519$ sheets. He gave out 156 so he has $519-156 = 363$ sheets. ✓	-1.39
Solutions		

Figure 1: A math question from GSM8K (Cobbe et al., 2021) and multi-step solutions sorted in descending order by their average log probability over tokens according to a fine-tuned FLAN-T5-Large (Chung et al., 2022). The model probabilities are miscalibrated with respect to the solution correctness as the incorrect solutions have higher likelihood over the correct one.

reasoning abilities of LMs, supervised fine-tuning (SFT) has been performed on gold step-by-step solutions (Uesato et al., 2022; Ho et al., 2022; Fu et al., 2023). However, SFT can easily lead to model overfitting of the gold solutions seen during training, resulting with an LM that assigns low probabilities to alternative but correct solutions (Ni et al., 2023). These issues with training objectives result in a *miscalibration* of the LMs’ probabilities with respect to the correctness of their outputs: high likelihoods are assigned to incorrect solutions and vice versa (Holtzman et al., 2021). For instance, when prompting a fine-tuned FLAN-T5-Large (Chung et al., 2022) with an unseen problem from GSM8K (Cobbe et al., 2021), the model easily assigns a higher average likelihood to incorrect solutions than correct ones (shown in Figure 1).

To elicit correct multi-step reasoning directly from LMs, *prompting* techniques have been proposed, with the scratchpad or chain-of-thought methods being particularly successful (Nye et al., 2021; Wei et al., 2022b; Wang et al., 2022). However, prompting methods do not directly address

\*Correspondence to khalifam@umich.edu

<sup>1</sup> Our code can be found at <https://github.com/mukhal/grace>.

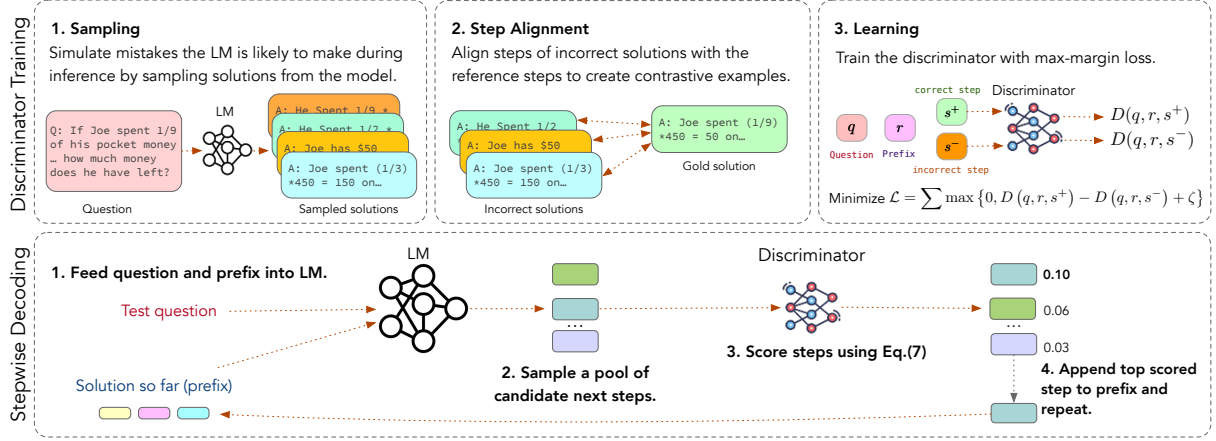


Figure 2: **Top:** The three-step process to train the discriminator. (1) **Sampling** solutions from a given language model with different mistakes by keeping the solutions with the incorrect final answers only. (2) **Alignment** involves aligning the sampled solutions with the reference solutions to identify incorrect steps. (3) **Learning** the discriminator with a max-margin loss to assign high scores to correct steps and low scores to incorrect steps. **Bottom:** The guided stepwise decoding process using the trained discriminator. Given the question and the prefix, we sample a pool of candidate next steps and use the discriminator to score steps as in eq. (7). Then the top-scored step is selected and added to the prefix. This process repeats until a final answer is generated.

the miscalibration issue and are only effective when the LM reaches a certain scale so that it can effectively utilize the prompt (Wei et al., 2022a). In parallel, *oversampling*-based techniques have been proposed to leverage information from multiple plausible solutions. For instance, the sample-then-rank approach scores a set of randomly sampled solutions based on their correctness, using a verifier model, and selects the one with the highest score (Cobbe et al., 2021; Li et al., 2022). Self-consistency has also been investigated (Wang et al., 2022) to aggregate multiple random samples by majority vote. These approaches, however, do not address the underlying problem as sampling is still performed based on the miscalibrated LM distribution.

To mitigate the miscalibration issue, this paper brings the insight that we can sample more correct solutions by *steering the decoding process towards generating correct reasoning steps*, and therefore leading to more accurate answers. Inspired by discriminator-guided controlled generation methods (Yang and Klein, 2021; Dathathri et al., 2020; Krause et al., 2021), we propose **GRACE**, which is a guided-decoding method that relies on a correctness discriminator model to nudge the decoding process towards correct steps. Our discriminator is trained at the step level; therefore providing a finer-grained control over the sampling process compared to self-consistency and sample-then-rank methods that act on complete samples. While recent work (Uesato et al., 2022) rely on hu-

man annotations to build a step-level reward model, human annotations are expensive and do not scale well. We work around this limitation and propose a 3-step approach to train the correctness discriminator based on access to the correct solutions only, without any step-level human annotations.

We compare GRACE to greedy decoding, self-consistency, and sample-then-rank and show strong improvements over all of them on four different math reasoning benchmarks with two different language models, namely FLAN-T5 (Chung et al., 2022) and LLaMA (Touvron et al., 2023). For instance, GRACE outperforms greedy decoding on GSM8K (Cobbe et al., 2021) by 7.4% accuracy points with FLAN-T5-Large and by 5.4% with LLaMA-7B. In addition, when further combining our approach with self-consistency, we outperform the vanilla self-consistency by 10.2% points on GSM8K and by 15.7% on MultiArith (Roy and Roth, 2015).

In summary, our contributions are:

- We propose a stepwise decoding strategy that guides the model towards correct multi-step solutions that rely on a step-level correctness discriminator. GRACE *does not necessitate any training* of the LM and only needs access to samples from the LM distribution.
- We propose a novel alignment algorithm to align incorrect solutions with correct ones, to automatically create step-level (in)correctness labels. This algorithm avoids the requirement of large amounts of human annotations for

reasoning steps (Uesato et al., 2022).

- GRACE shows significant improvements over greedy decoding on four different math reasoning benchmarks. We also perform human evaluation and LLM-based evaluation and show that our approach produces 7% more correct steps compared to greedy decoding and 3.8% compared to self-consistency. According to human evaluation, GRACE reduces the solution error rate from 9.0% with greedy to 5.0% (about 44% reduction).

## 2 Related Work

### Discriminator-guided Controlled Generation

Previous work in controlled generation has employed discriminators during decoding to guide generation towards specific attributes, such as sentiment, topic, or lexical constraints (Holtzman et al., 2018; Dathathri et al., 2020; Yang and Klein, 2021; Krause et al., 2021). These discriminators can either update the hidden states of the language model in real-time (Dathathri et al., 2020) or adjust token probabilities (Holtzman et al., 2018; Yang and Klein, 2021). Our research takes inspiration from these practices but extends them to multi-step reasoning in two key aspects: **control granularity** and **discriminator training**. We direct the decoding of multi-step solutions at the level of reasoning steps to promote their correctness, instead of individual tokens as correctness is often not meaningfully defined at the token level. In terms of discriminator training, we note that training a correctness discriminator is more challenging than training a topic or sentiment discriminator since judging correctness requires checking the given step for logical, mathematical, or factual inconsistencies with respect to the context i.e., the question and the prefix. To address this challenge, we design a novel 3-step process for training discriminators without requiring step-level annotations.

**Multi-step reasoning.** Two main types of approaches have been explored: inference-time methods, which do not require additional language model (LM) training, and training-based methods, which require either labeled samples or rewards. Popular inference-time techniques include model prompting such as chain-of-thought prompting (Nye et al., 2021; Wei et al., 2021) and its variants (Zhou et al., 2022; Zhang et al., 2022; Fu et al., 2022). While these input-based techniques modify the input by LM, other methods target the output

side, e.g., self-consistency (Wang et al., 2022) employs majority voting on multiple sampled solutions to determine the final answer. An alternative output-based method involves training a verifier model to rank sampled solutions according to correctness. As demonstrated by Cobbe et al. (2021), it is feasible to enhance GPT-3’s math reasoning performance by training a verifier model to predict the correctness of sampled solutions, using labels based on known final answer correctness. However, verifiers exhibit no control over solution sampling. We also show in this paper (see section 5) that verifiers trained on samples from smaller LMs perform very poorly. Training-based methods, on the other hand, focus on crafting learning objectives to teach the LM to reason correctly. For instance, Uesato et al. (2022) trained a reward model to assess the correctness of the entire reasoning chain, beyond the final answer and then used it to train the LM via Reinforcement Learning. Human annotations were used to provide step-level labels for training the step-level reward model. Ni et al. (2022) proposed training LMs on sampled partially correct solutions to enhance mathematical reasoning. The work most relevant to ours is by Li et al. (2022), who introduced a step-aware verifier to score sampled solutions. Despite the demonstrated benefits of including step-level information, their technique only applies to fully sampled solutions, unlike our approach which actively guides the decoding process. Yang et al. (2022) use a stepwise verifier to guide the search process for proof generation. They use heuristics to generate negative examples while, on the other hand, we sample incorrect solutions from the model and create examples using an alignment process with the reference solutions. Also, their setting is limited to the task of proof generation while we focus on chain-of-thought style math reasoning.

## 3 Method

**Overview.** Our setup follows chain-of-thought reasoning (Nye et al., 2021; Wei et al., 2021), where given a question  $q$  (e.g., a math word problem), our goal is to predict a step-by-step solution or chain of  $T$  intermediate reasoning steps that end with the answer  $s_1, s_2, \dots, s_T$ , where  $s_T$  is the final answer. This is typically done using a pretrained language model (LM) that is either fine-tuned or prompted in a few-shot manner. Typical decoding approaches designed for text generation

prioritize sequence likelihood and can easily generate invalid steps that ultimately lead to an incorrect answer. Our goal is to improve the reasoning ability of the LM by guiding the solution generation process via a discriminator  $D$  that models the correctness of a given reasoning step with the goal of preventing the LM from generating incorrect steps and ultimately sampling high-quality reasoning chains.

We will start by formalizing our approach in section 3.1. We then present a three-step procedure to train the discriminator (section 3.2) and then explain how it is used to guide the stepwise decoding process described in section 3.3. A detailed overview of GRACE is shown in fig. 2.

### 3.1 Formalization

Given a problem  $q$  and a correct solution prefix  $s_1, s_2, \dots, s_{t-1}$  we want to sample a correct next step  $s_t$  towards the final answer.<sup>2</sup> We assume access to a judge or a discriminator model  $D$  that takes in the problem  $q$ , the prefix  $s_1, s_2, \dots, s_{t-1}$  and a candidate next step  $s_t$  and outputs a real-valued score  $D(q, s_{1:t-1}, s_t)$  that indicates whether  $s_t$  is a correct candidate step at time-step  $t$ . We also assume access to language model distribution  $p_{\text{LM}}(\cdot|q, s_{1:t-1})$  that is either trained or to be used in a few-shot prompting manner to generate  $s_t$ .

Formally, let  $c$  be a binary variable that indicates the correctness of the generated step with respect to the question and prefix, where we want to sample the next step  $s_t \sim p(\cdot|s_{1:t-1}, c, q)$ . A Bayesian factorization of  $p(s_t|s_{1:t-1}, c, q)$  is:

$$p(s_t|s_{1:t-1}, c, q) \quad (1)$$

$$\propto p(s_t|s_{1:t-1}, q) \cdot p(c|s_t, s_{1:t-1}, q) \quad (2)$$

$$= p(s_t|s_{1:t-1}, q) \cdot p(c|s_{1:t}, q) \quad (3)$$

$$= p_{\text{LM}}(s_t|q, s_{1:t-1}) \cdot p(c|s_{1:t}, q) \quad (4)$$

$$\propto p_{\text{LM}}(s_t|q, s_{1:t-1}) \cdot \exp(D(q, s_{1:t-1}, s_t)) \quad (5)$$

As depicted in eq. (4), we substitute the probability of the next step without correctness  $p(s_t|s_{1:t-1})$  with  $p_{\text{LM}}(s_t|q, s_{1:t-1})$ . Similarly, in eq. (5),  $p(c|s_{1:t}, q)$  is replaced with  $\exp(D(q, s_{1:t-1}, s_t))$ . This substitution is justified as, in accordance with our discriminator’s definition,  $\exp(D(q, s_{1:t-1}, s_t))$  is proportionate to  $p(c|s_{1:t}, q)$ . Finally, as we assume that the pre-

fix  $s_{1:t-1}$  is correct,  $p(c|s_{1:t}, q)$  becomes dependent only on the correctness of  $s_t$ , modeled by  $D(q, s_{1:t-1}, s_t)$ .

This form of factorization echoes the controlled generation method used by FUDGE (Yang and Klein, 2021), but with two notable distinctions. First, we model the next step as opposed to the next token probability, which is ill-defined. Second, unlike FUDGE’s discriminator, which predicts *future* attribute satisfaction, our discriminator focuses on the *present*, evaluating whether the current step  $s_t$  will result in a correct prefix  $s_{1:t}$ . That is, the discriminator is guiding decoding at the step level as opposed to verifiers and self-consistency, which operate on complete solutions. To summarize, eq. (5) shows that we want to sample  $s_t$  (i) with high likelihood  $p_{\text{LM}}(s_t|q, s_{1:t-1})$  according to the language model and (ii) is correct with respect to the question and the prefix. Intuitively, this implies the utilization of the reasoning capabilities of the LM while maintaining correctness. Throughout the rest of the paper, we will refer to the correct prefix  $s_{1:t-1}$  as  $r$  and the next step  $s_t$  as  $s$  for simplicity.

### 3.2 Discriminator Learning

We use three steps to learn the discriminator function  $D(q, r, s)$ , which are shown in Figure 2 (top).

- **Step 1–Negative sampling:** We collect a set of solutions with at least one incorrect step.
- **Step 2–Alignment:** We align these solutions with the ground truth to create examples with positive and negative steps to train the discriminator.
- **Step 3–Learning:** We train the discriminator with a contrastive objective to distinguish between correct and incorrect steps.

**Negative Sampling.** This step aims to collect a set of solutions with incorrect steps. For each question in the training set, we sample multiple solutions via top- $k$  sampling and only keep solutions with an incorrect final answer (to make sure the solution has at least one incorrect step). We refer to this set of solutions as the *negative pool*. Although negative examples can be constructed by introducing perturbations in reference steps with a predefined set of edit operations (e.g., Golovneva et al. (2023)), we found that this does not benefit discriminator training as the perturbations produce easy negatives with artifacts that do not resemble the type of mistakes an LM will make.

<sup>2</sup>We assume the prefix so far is correct to focus on modeling the next step prediction. An empty prefix is trivially correct.



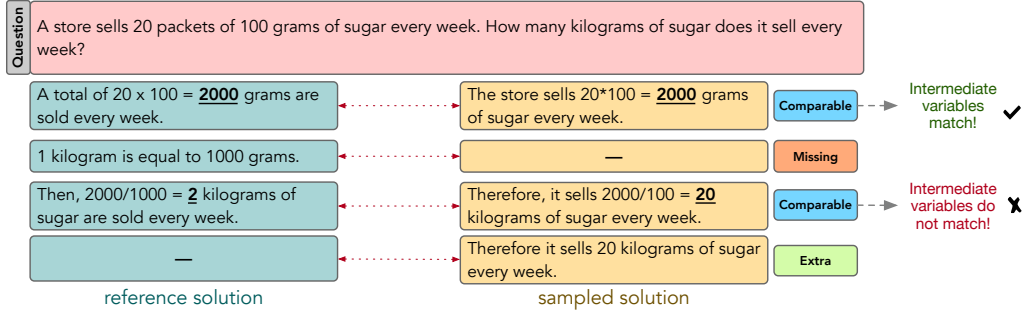


Figure 3: An example of the alignment produced by our alignment algorithm (described in Algorithm 2). The question and the reference solutions come from GSM8K (Cobbe et al., 2021). The “-” designates a step placeholder. There are three possible cases when aligning a reference solution with a sampled solution: **extra step**, **missing step**, and **aligned step**. In the aligned case, the intermediate variables (underlined) are compared to determine the correctness of the sampled step. Algorithm 1 describes how each case is handled when constructing the discriminator training data.

**Alignment.** Our objective is to train the discriminator  $D$  to effectively differentiate between correct and incorrect steps. To achieve this, we require a dataset that consists of both correct and incorrect step examples. To acquire such dataset without step-level annotations of any kind, we align sampled incorrect solutions with the reference solution via dynamic programming using the Needleman-Wunsch (NW) algorithm (Likic, 2008) commonly employed in bioinformatics applications. As the NW algorithm works on sequences with *different* lengths, it allows us to model both missing and extra steps which prior work does not take into account (Li et al., 2022; Ni et al., 2022). The standard NW algorithm aims to find a minimum-cost alignment between two character sequences and is not directly applicable to our case without defining a notion of step equivalence. For that purpose, we use *cosine distance* between step embeddings to compute alignment cost and introduce a similarity threshold to determine step equivalence. We compute step embeddings using ROSCOE (Golovneva et al., 2023), a RoBERTa-base (Liu et al., 2019) model based on SimCSE (Gao et al., 2021) and fine-tuned on positive and negative pairs of questions and reference reasoning steps. Our initial experiments found ROSCOE to produce better alignment compared to the vanilla SimCSE. The detailed alignment algorithm is shown in algorithm 2 in Appendix B. We then use the alignment to obtain pairwise examples in the form of  $(q, r, s^+, s^-)$  where  $s^+$  is the correct next step and  $s^-$  is the incorrect next step after prefix  $r$ . We do that by iterating over pairs of aligned steps and handling the three following cases: missing, extra, and comparable steps i.e., a step that can be directly comparable to its counterpart from the reference solution. In

---

**Algorithm 1** Discriminator training data construction.

---

**Input:** Question  $q$ , aligned sampled solution  $\tilde{t}$ , aligned gold solution  $\tilde{g}$ .

**Output:** Pairwise examples for discriminator training  $E$ .

```

 $P, E \leftarrow \emptyset, \emptyset$ 
 $m \leftarrow |\tilde{t}|$ 
for  $i \in \{1, \dots, m\}$  do
  if  $\tilde{t}_i = \_$  then
     $P \leftarrow P \cup \{\tilde{g}_i\}$ 
  else if  $\tilde{g}_i = \_$  then
     $\tilde{g}_j \leftarrow \text{next\_gold\_step}(\tilde{g}, i)$ 
    if  $\text{exists}(\tilde{g}_j)$  then
       $E \leftarrow E \cup \{(q, P, \tilde{g}_j, \tilde{t}_i)\}$ 
  else
    if  $\text{steps\_match}(\tilde{t}_i, \tilde{g}_i)$  then
       $P \leftarrow P \cup \{\tilde{t}_i\}$ 
    else
       $E \leftarrow E \cup \{(q, P, \tilde{g}_i, \tilde{t}_i)\}$ 
    exit
return  $E$ 

```

---

the comparable step case, we compare the intermediate variables computed at each step following prior work (Ni et al., 2022; Li et al., 2022). An example of all three cases is shown in fig. 3. Algorithm 1 details the process we use to construct the discriminator pairwise examples.

**Learning.** For a set of  $M$  pairwise examples  $\{(q_i, r_i, s_i^+, s_i^-)\}_{i=1}^M$ , the training objective for the  $i$ -th example is to maximize the difference  $\delta_i = D(q_i, r_i, s_i^+) - D(q_i, r_i, s_i^-)$ . We utilize the max-margin loss objective  $\mathcal{L}_D$  in eq. (6) (Rosasco et al., 2004; Li et al., 2020), where  $\zeta > 0$  is a hyperparameter. We found that the max-margin loss performs better than other alternatives (see section 6 for an ablation).

$$\mathcal{L}_D = \sum_{i=1}^M \left[ \max\{0, -\delta_i + \zeta\} \right] \quad (6)$$

### 3.3 Guided Stepwise Decoding

After  $D$  is trained, we can then use it to guide solution sampling. At each time  $t$ , we use nucleus sampling to sample a pool of  $J$  candidates for the next steps  $\mathcal{S} = \{s_t^{(1)}, s_t^{(2)}, \dots, s_t^{(J)}\}$  from  $p_{\text{LM}}(\cdot|q, r)$ .<sup>3</sup> These candidates represent multiple possible choices for the next step. Each candidate step  $s_t^{(i)}$  is then scored using:

$$(1 - \beta) \log p_{\text{LM}}(s_t^{(i)}|q, r) + \beta \bar{D}(q, r, s_t^{(i)}) \quad (7)$$

where  $\beta$  is a hyperparameter and  $\bar{D}(q, r, \hat{s})$  is the softmax-normalized discriminator score across the candidate steps:

$$\bar{D}(q, r, s_t^{(i)}) = \frac{\exp(D(q, r, s_t^{(i)}))}{\sum_{j=1}^J \exp(D(q, r, s_t^{(j)}))}$$

The discriminator score is normalized so as to make sure we are adding two log probabilities in when computing the score. The process continues until a final answer is generated or until a maximum number of steps is reached. The guided decoding process is shown in Figure 2 (bottom).

## 4 Experimental Setup

**Tasks.** We evaluate our approach on four popular multi-step math reasoning tasks. The **GSM8K** dataset (Cobbe et al., 2021) is one of the most commonly used benchmarks for complex multi-step reasoning. It consists of math word problems for 8th graders, each containing the corresponding step-by-step solution. We use the original split by Cobbe et al. (2021) and use 1K solutions from the training set as the development set. **MathQA-Gain** is a subset of MathQA (Amini et al., 2019) which includes math word problems about gain/loss. Each problem is accompanied by a step-by-step Python program. **SVAMP** (Patel et al., 2021) and **Multi-Arith** (Roy and Roth, 2015) consist of elementary-level math word problems. For MathQA-Gain, SVAMP, and MultiArith, we use the splits included in the LILA benchmark (Mishra et al., 2022).

As SVAMP and MultiArith do not include reference step-by-step solutions (only the final answer is included for each question) we follow recent work on chain of thought distillation (Ho et al., 2022; Fu et al., 2023; Hsieh et al., 2023) and prompt GPT-3.5 to generate step-by-step solutions. We sample

<sup>3</sup>We make sure each sample will contain only one step by halting when a special end-of-step token is reached.

20 solutions for each question and only keep the questions for which GPT-3.5 was able to reach the correct final answer. More details on this process and exact data statistics are in Appendix E.1.

**Metrics.** We evaluate GRACE in terms of final answer accuracy as in prior work in addition to the step correctness as measured by both a LLM and human evaluation.

**Baselines.** We compare GRACE to **greedy decoding** at the token level, which is the standard decoding method for reasoning tasks (Wei et al., 2022b; Li et al., 2022; Fu et al., 2022; Zhou et al., 2022). We also compare to **self-consistency** (Wang et al., 2022), where multiple solutions are sampled with a temperature of  $T = 0.7$  and we pick the most frequent answer as the final answer. We sample 40 solutions for experiments with FLAN-T5 and 20 with LLaMA. Lastly, we compare to **sample-then-rank** or verifiers (Cobbe et al., 2021; Uesato et al., 2022; Li et al., 2022), where a classifier is trained to read the question and the full solution and then predict a binary label of correct or incorrect. The labels are based on the final answer’s correctness. At inference, we sample multiple solutions with temperature  $T = 0.7$  and we pick the solution with the highest correctness probability according to the verifier. For a fair comparison with GRACE, the verifier is also based on a T5-Large encoder as our discriminator and trained on the same incorrect samples, along with the correct counterparts. We use the verifier checkpoint that achieves the best macro F1 on a held-out set. It is worth noting that while we compare to self-consistency and verifiers, they are not necessarily competitors to our technique and can indeed be combined with GRACE i.e., we can sample complete solution using our guided decoding approach then rerank or apply majority voting over the sampled solutions. Indeed, we show in the next section that applying self-consistency on top of samples from GRACE performs consistently better across the board than either the vanilla self-consistency or GRACE separately.

**Models.** We verify the effectiveness of GRACE on two language models from different families and with different sizes, namely FLAN-T5-Large (778M) (Chung et al., 2022) and LLaMA (7B) (Touvron et al., 2023). We fine-tune FLAN-T5 over the training set of each task while LLaMA is used in a few-shot setting with chain-of-thought

	FLAN-T5-LARGE			LLAMA-7B		
	GSM8K	SVAMP	MathQA-Gain	GSM8K	MultiArith	SVAMP
Greedy decoding	26.9	54.5	76.5	12.9	54.0	32.8
Self-consistency	33.3	61.8	78.9	20.7	78.9	52.4
Sample-then-rank	20.5	45.9	83.7	9.60	46.4	26.1
GRACE	34.3 (+7.4)	66.2 (+11.7)	84.1 (+6.0)	16.2 (+3.30)	84.9 (+30.9)	49.7 (+17.3)
GRACE w/ self-consistency	<b>36.3</b> (+3.0)	<b>68.6</b> (+6.80)	84.4 (+0.7)	<b>30.9</b> (+10.2)	<b>94.6</b> (+15.7)	<b>55.6</b> (+3.20)

Table 1: Final answer accuracy on four multi-step reasoning tasks. Self-consistency and verifier results use 40 samples for FLAN-T5-Large experiments and 20 samples for LLaMA. The discriminator used with GRACE is T5-Large encoder. FLAN-T5-Large results are aggregated over 5 runs and LLaMA over 3 runs. The absolute improvement by GRACE w.r.t to the corresponding baseline is shown in parentheses. GRACE with self-consistency outperforms the baselines on all tasks.

prompting (Wei et al., 2022b) with 6 demonstrations.

**Sampling and Discriminator Training.** For each task, we sample roughly 80K incorrect solutions for discriminator training with top- $k$  sampling with  $k = 50$  and temperature  $T = 1.3$  for FLAN-T5 and  $T = 0.7$  for LLaMA. The discriminator used in all our experiments is a FLAN-T5-Large encoder (~340M). The step score is computed by applying max-pooling over the hidden states followed by a two-layer MLP with a ReLU and tanh non-linearities. The tanh is applied to constrain the scores in the range  $[-1, 1]$ . We train the discriminator for 10 epochs with a batch size of 32. We use the Adam optimizer with a learning rate of  $1e-4$  for GSM8K and  $6e-5$  for other tasks. We use  $\zeta = 1.0$  as the margin hyperparameter. We monitor the loss on a held-out development set and choose the best checkpoint.

**Decoding.** For step-wise decoding, we sample reasoning steps using nucleus sampling to form the pool of candidate next steps. We continue decoding steps until a final answer is generated or until a maximum number of steps is reached. Table 3 provides concrete hyperparameters used for stepwise decoding for each task.

**Calculator.** All our results rely on a calculator during decoding. That is, whenever a formula is encountered, a calculator module is invoked to compute the result, which is then given back to the LM to continue sampling. This is to relieve the LMs from having to do simple math operations and to let them focus on the actual reasoning process.

## 5 Results and Discussion

### Can GRACE improve final answer correctness?

Here, we focus on comparing the accuracy of the final answer (also known as solve rate). We

first discuss the results with FLAN-T5-Large on GSM8K, SVAMP, and MathQA-Gain (shown in Table 1).<sup>4</sup> We see that GRACE outperforms the baselines on all tasks. For instance, GRACE is outperforming greedy decoding by 7.4% and 11.7% points over GSM8K and SVAMP, respectively. Interestingly, combining our approach with self-consistency, where sampling is done using GRACE and then majority voting is applied on the samples, further boosts the accuracy improving on vanilla self-consistency by as large as 6.8 points on SVAMP.

Moving to the results on LLaMA-7B, we see a similar trend where GRACE outperforms greedy decode and self-consistency on MultiArith and SVAMP. GRACE with self-consistency outperforms self-consistency with random sampling by 10.2% and 15.7% points on GSM8K and MultiArith, respectively. Ultimately, our results show that GRACE is indeed able to boost both FLAN-T5 and LLaMA’s final answer correctness on all tasks. Interestingly, in the case of LLaMA-7b, we observe such improvements (i) without having to train the LM *at all* and (ii) with a discriminator that has 20X fewer parameters than the LM. This points to a promising direction of our approach in steering the generations of huge LLMs using significantly smaller discriminator models.

One final observation is that the verifier approach performs extremely poorly over all tasks except for MathQA-Gain. This is likely due to the fact that the training examples of the verifier include positive examples (i.e., solutions with correct final answers) but have incorrect or invalid reasoning steps. These correspond to cases where the model reached the correct answer spuriously without correct reason-

<sup>4</sup>We do not show FLAN-T5 results on MultiArith as it already achieves  $> 90\%$  accuracy. We do not show results of LLaMA on MathQA-Gain since it performs extremely poorly ( $< 2\%$ ).

ing. Training the verifier on these prevents the verifier from identifying correct from incorrect reasoning. To test this hypothesis, we ran an experiment where we only included the gold trajectories as positive examples and found the verifier performance to improve significantly (although it still underperformed self-consistency and GRACE). That explains why the verifier helps with MathQA-Gain since for a solution to have a correct final answer, it must correspond to a correct runnable Python program. These findings may initially seem to conflict with previous findings, where the verifier approach was shown to be indeed beneficial (Cobbe et al., 2021; Li et al., 2022). However, one should note that in these works, the verifier is trained over examples sampled from much larger LMs (e.g., GPT-3). In this case, it is certainly expected that reaching the correct final answer correlates more strongly with correct reasoning (which has been shown in Uesato et al. (2022)) compared to our setting with FLAN-T5-Large, and therefore the verifier model encounters the above issue much less frequently.

**Does GRACE produce more correct steps compared to baselines?** Reaching a correct final answer does not always correspond to correct reasoning; The model can reach the correct answer spuriously (Golovneva et al., 2023; Uesato et al., 2022). Here, we want to measure if GRACE yields more correct steps compared to the baselines. To do that, we use *prefix correctness* (PC) following Uesato et al. (2022), which measures whether the steps so far are correct. Inspired by recent work showing that using LLMs for evaluation correlates highly with human judgment (Wang et al., 2023; Liu et al., 2023; Luo et al., 2023), we measure prefix correctness using LLMs in addition to human evaluation. For LLM evaluation, we use GPT-3.5 with a few-shot prompt that lets the model predict a binary label of correct or incorrect after each prefix. Details on LLM evaluation are in Appendix C.

In addition to PC, which is computed over all solutions regardless of the final answer we also evaluate the *trace error*, which is computed exclusively on solutions with correct final answer and measures the percentage of these solutions that have at least one major mistake, which is defined as “A step where the information expressed is incorrect, or it would no longer be possible to reach the correct solution without undoing that step” following Uesato et al. (2022). We evaluate trace error using both human and LLM evaluation. As for trace er-

ror evaluation with LLM compute the percentage of correct solutions with at least one incorrect prefix. As for human evaluation, we ask annotators to label each solution as to whether it has such a major mistake and to mark the step where the mistake happened and a justification of their decision. Concrete details on the human evaluation are in Appendix D. We conduct this evaluation on GSM8K test set since the reasoning required to answer its questions is considered more complex compared to other tasks.

Table 2 shows the LLM and human evaluation results comparing GRACE to greedy decoding and self-consistency. GRACE scores higher than both greedy decoding and self-consistency by 7.0 and 3.8 points respectively. We observe significant improvements in terms of the trace error as well with GRACE. For instance, GRACE reduces trace error from 9.0% with greedy decoding to 5.0% (44% reduction), and a similar improvement is seen in the LLM-computed trace error. Our results clearly suggest that guiding the decoding process with GRACE not only improves the final answer correctness but also the correctness of the intermediate steps generated by the LM.

	<b>Prefix Correctness- LLM (<math>\uparrow</math>)</b>	<b>Error- LLM (<math>\downarrow</math>)</b>	<b>Error- Human (<math>\downarrow</math>)</b>
Greedy decode	46.5	7.0	9.0
S.C	51.0	9.8	-
GRACE	53.5 (+7.0)	5.2 (-1.8)	5.0 (-4.0)
GRACE w/ S.C	54.8 (+3.8)	6.6 (-3.2)	-

Table 2: Solution prefix correctness computed over GSM8K. GRACE and self-consistency (S.C) metrics are averaged over 3 runs. Prefix correctness is computed over 1.3K questions. Trace error-LLM is computed over ~300 questions and trace error-human over 200 questions. All solutions come from GSM8K test set. Details on the evaluations in Appendix C and Appendix D

## 6 Analysis

**Alignment.** Our hypothesis is that aligning sampled solutions with reference solutions using the Needleman-Wunsch (NW) algorithm allows us to leverage solutions with different lengths than the reference, thereby capturing extra and missing steps. To validate this hypothesis, we compare our alignment approach to a naive version where steps in the sampled solutions are aligned with the corresponding steps in the reference solutions. However, the naive approach only considers samples with the



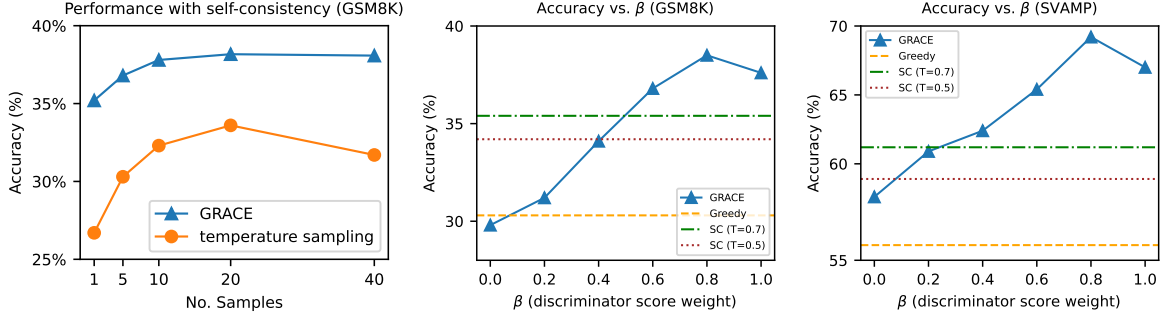


Figure 4: **Left:** Self-consistency performance on 400 examples from GSM8K dev set with GRACE compared to temperature sampling (Wang et al., 2022). GRACE exhibits better sample efficiency and does not incur a performance drop when using more samples compared to temperature sampling. **Mid and Right:** Final answer accuracy on GSM8K and SVAMP dev sets as we vary the discriminator score weight  $\beta$  in eq. (7). The model used is FLAN-T5-Large and the discriminator is FLAN-T5-Large encoder. All results except for greedy are averaged over 3 runs. Increasing  $\beta$  improves the final answer accuracy, suggesting the benefit given by steering the decoding process via the discriminator.

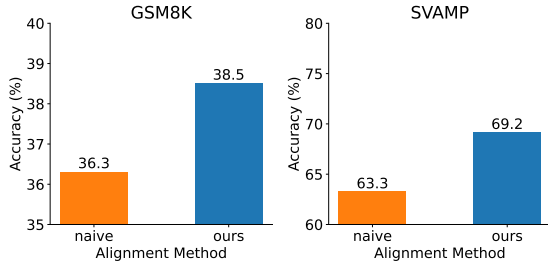


Figure 5: Comparison of final answer accuracy on GSM8K and SVAMP between our alignment method and the naive approach. Our alignment method, leveraging the Needleman-Wunsch algorithm, outperforms the naive approach by 2.2 and 5.9 on GSM8K and SVAMP, respectively, demonstrating the effectiveness of our proposed alignment method in improving discriminator training.

same number of steps as the reference solution, as there is no clear way to align samples with different lengths. Figure 5 presents the final answer accuracy on GSM8K and SVAMP when the discriminator is trained using both alignment methods. We observe a significant gap between our alignment method and the naive approach, with our method achieving better performance by 2.2% and 5.9% points on GSM8K and SVAMP, respectively. These results highlight the advantages of our proposed alignment method in improving discriminator training.

**Sampling Efficiency.** A primary motivation for GRACE is to provide more control over the solution sampling process compared to standard temperature sampling. To verify whether GRACE samples more correct solutions, we compare it to temperature sampling when used for self-consistency with different numbers of samples. Figure 4 (left) shows a plot of the number of samples against final answer accuracy over GSM8K. We observe GRACE is indeed more sample-efficient and yields better

accuracy with the same number of samples as temperature sampling. We also observe a drop in the accuracy with  $N = 40$  with temperature sampling, which we do not observe with GRACE.

**Step Score.** We study the effect of the discriminator score weight  $\beta$  in eq. (7) when computing the score of a candidate step on the reasoning performance. Figure 4 (mid and right) shows final answer accuracy on GSM8K and SVAMP as we vary  $\beta$  from 0.0 to 1.0. We can observe the accuracy improving as  $\beta$  is increased until 0.8, and then drops slightly at 1.0. This emphasizes the benefit brought by integrating  $D(q, r, s)$  into the step scores while also showing that we should not completely omit  $p_{\text{LM}}(s|q, r)$ , which represents the LM’s learned reasoning abilities. We observe a similar trend for  $\beta$  over all remaining tasks.

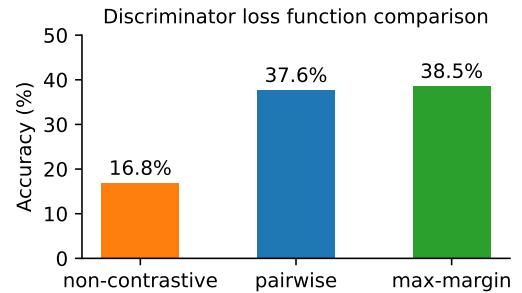


Figure 6: Guided decoding accuracy on GSM8K when the discriminator is trained with different loss functions. Our max-margin loss outperforms both the non-contrastive version (Uesato et al., 2022) and the pairwise ranking loss (Ouyang et al., 2022). Results are averaged over 3 runs.

**Discriminator Loss Function.** We compare the max-margin objective in eq. (6) to two different discriminator training objectives. The first is a non-contrastive binary objective, where the

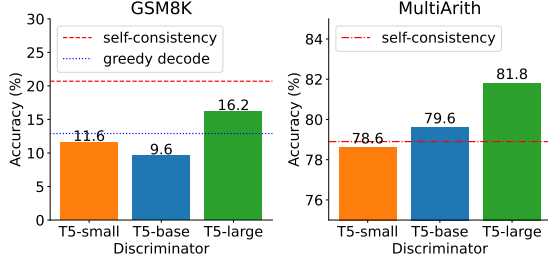


Figure 7: Accuracy on GSM8K and MultiArith with different discriminator sizes. Discriminator size matters: Larger discriminators work better than smaller ones since they have more capacity to model correctness. The complexity of the task matters: GRACE requires a larger discriminator than MultiArith to produce an observable performance boost as GSM8K involves more complex questions.

model is trained to predict correct or incorrect after each step following Uesato et al. (2022), and the probability of correctness is used as the discriminator score in eq. (7). The second is the pairwise ranking loss used to train the reward model for InstructGPT (Ouyang et al., 2022)  $\mathcal{L}_D^{\text{pairwise}} = -\sum \log [\sigma(D(q, r, s^+) - D(q, r, s^-))]$ . Figure 6 shows final answer accuracy on GSM8K when GRACE’s discriminator is trained with each of these loss functions. Notably, the non-contrastive loss exhibits the lowest accuracy, emphasizing the importance of enabling the discriminator to correct incorrect steps jointly, rather than evaluating them in isolation. Moreover, our proposed max-margin objective achieves marginally better performance than the pairwise ranking loss. We posit that this enhancement stems from the incorporation of the margin parameter, which prevents the discriminator from excessively optimizing the objective. These results highlight the efficacy of the max-margin objective in training the correctness discriminator.

**Discriminator Size.** We study how the size of the discriminator impacts the final answer accuracy. In addition to the FLAN-T5-Large encoder used so far, we run experiments with a FLAN-T5-Base encoder (110M) and a FLAN-T5-Small encoder (30M) as discriminators on GSM8K and MultiArith and with LLaMA as the backbone LM. Figure 7 shows the accuracy on both datasets with different model sizes. For MultiArith, better performance is brought by larger discriminator models, which is expected. Interestingly, using the T5-base discriminator, GRACE can already surpass self-consistency by 0.7 points, and such a boost is achieved using a discriminator that is 63X smaller than LLaMA.

As for GSM8K, we observe a very different trend, where smaller models (base and small) do not perform well. This can be understood in the light of GSM8K being a more difficult task with more complex reasoning requirements compared to MultiArith and therefore a discriminator with sufficient capacity is needed.

## Conclusion

When solving multi-step reasoning problems, language models are often miscalibrated with respect to correctness, leading to high-probability solutions that are not necessarily correct. Existing methods like self-consistency and verifiers that rely on sampling from the LM distribution do not effectively address this issue. To tackle this problem, we introduce GRACE, which utilizes a discriminator model trained to distinguish between correct and incorrect reasoning steps. The discriminator is used to steer the decoding process toward correct steps and thus preventing the language model from generating invalid ones. Experimental results on four popular math reasoning benchmarks demonstrate that GRACE significantly improves the correctness of the generated solutions at both the final answer and the intermediate steps levels. We further validate the effectiveness of different components of our method through multiple ablations.

## Limitations and Future Work

There is an overhead incurred by incorporating the discriminator model during decoding as we pause decoding at each step to compute the discriminator scores. Also, our approach requires access to reference step-by-step solutions for the alignment process. In this paper, we use an LLM to obtain these for two tasks, LLMs can make mistakes yielding incorrect reference solutions, especially for more complex reasoning tasks. As for future directions, it is possible to iterate the 3-step process to train the discriminator by sampling solutions using GRACE and then performing the alignment and re-training the discriminator and so on. We leave it to future work to explore this direction. Extending this work to logical and symbolic reasoning tasks is also one promising future direction.

## References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. [Mathqa: Towards interpretable math](#)

- word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2357–2367. Association for Computational Linguistics.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.
- Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.
- Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. [Plug and play language models: A simple approach to controlled text generation](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*.
- Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [Simcse: Simple contrastive learning of sentence embeddings](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.
- Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. [ROSCOE: A suite of metrics for scoring step-by-step reasoning](#). In *The Eleventh International Conference on Learning Representations*.
- Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.
- Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. [Learning to write with cooperative discriminators](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1638–1649. Association for Computational Linguistics.
- Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. [Surface form competition: Why the highest probability answer isn’t always right](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7038–7051. Association for Computational Linguistics.
- Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. [Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes](#).
- Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. [Gedi: Generative discriminator guided sequence generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 4929–4952. Association for Computational Linguistics.
- Jingjing Li, Zichao Li, Lili Mou, Xin Jiang, Michael Lyu, and Irwin King. 2020. Unsupervised text generation by learning from search. *Advances in Neural Information Processing Systems*, 33:10820–10831.
- Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*.
- Vladimir Ljickic. 2008. The needleman-wunsch algorithm for sequence alignment. *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, pages 1–46.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. [Gpteval: Nlg evaluation using gpt-4 with better human alignment](#). *arXiv preprint arXiv:2303.16634*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for abstractive text summarization. *arXiv preprint arXiv:2303.15621*.

- Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2022. [LILA: A unified benchmark for mathematical reasoning](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5807–5832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. 2023. [Learning math reasoning from self-sampled correct and partially-correct solutions](#). In *The Eleventh International Conference on Learning Representations*.
- Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Oleksandr Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. 2022. Learning from self-sampled correct and partially-correct programs. *arXiv preprint arXiv:2205.14318*.
- Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. [Show your work: Scratchpads for intermediate computation with language models](#). *CoRR*, abs/2112.00114.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2080–2094. Association for Computational Linguistics.
- Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. 2004. Are loss functions all the same? *Neural computation*, 16(5):1063–1076.
- Subhro Roy and Dan Roth. 2015. [Solving general arithmetic word problems](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752. The Association for Computational Linguistics.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.
- Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. [Emergent abilities of large language models](#). *CoRR*, abs/2206.07682.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. [Generating natural language proofs with verifier-guided search](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 89–105. Association for Computational Linguistics.
- Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*.
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.



## A Hyperparameters

Table 3 shows the stepwise decoding hyperparameters used for each task and language model used. These values were found through a grid search over the development set for each task.

## B Solution Alignment

Algorithm 2 shows the Needleman-Wunsch algorithm for aligning sampled solutions with the ground-truth solution for a given problem. In practice, we discard examples with alignment cost  $> 2.0$ .

---

**Algorithm 2** Step Alignment using Needleman-Wunsch

---

**Input:** Sampled solution  $t$ , ground-truth solution  $g$ , gap cost  $c$ , similarity threshold  $\gamma$

**Output:** aligned\_solutions  $\tilde{t}, \tilde{g}$ , min\_alignment\_cost

```
1:  $m \leftarrow \text{len}(t)$ 
2:  $n \leftarrow \text{len}(g)$ 
3:  $P \leftarrow \text{pairwise\_similarity}(t, g)$ 
4:  $i \leftarrow 0; j \leftarrow 0; L \leftarrow \text{zeros\_matrix}(m + 1, n + 1)$ 
5:  $L_{:,m+1,0} \leftarrow [i*c \text{ for } i \text{ in } 1 \dots m]$ 
6:  $L_{0,:,n+1} \leftarrow [i*c \text{ for } i \text{ in } 1 \dots n]$ 
7:  $i \leftarrow 1$ 
8: while  $i \leq m$  do
9:    $j \leftarrow 1$ 
10:  while  $j \leq n$  do
11:    if  $P_{i-1,j-1} \geq \gamma$  then
12:       $L_{i,j} \leftarrow L_{i-1,j-1}$ 
13:    else
14:       $L_{i,j} \leftarrow \min(L_{i-1,j-1} + 1 - P_{i-1,j-1},$ 
         $L_{i-1,j} + c, L_{i,j-1} + c)$ 
15:     $j \leftarrow j + 1$ 
16:   $i \leftarrow i + 1$ 
17: min_alignment_cost  $\leftarrow L_{m,n}$ 
18:  $\tilde{t}, \tilde{g} \leftarrow \text{backtrack}(L, t, g)$ 
19: return  $\tilde{t}, \tilde{g}$ , min_alignment_cost
```

---

## C LLM Evaluation Details

Before using GPT-3.5 to evaluate our model, we need to measure whether it can reliably assess the prefix correctness. To do that, we manually annotate 100 model-generated solutions from GSM8K which corresponded to 280 prefixes in total. We ask human annotators to provide a binary label for each prefix to indicate whether the solution so far will still lead to the correct final answer or not. If a prefix is found to be incorrect, then all the following prefixes in the solution are also incorrect. Interestingly, we found that the few-shot prompting GPT-3.5 could predict the prefix correctness with around 88.8% accuracy. The few-shot prompt we use is shown in Table 4. We run our evaluation on three different runs for GRACE and self-consistency results and sample 10 different demonstrations each time for the prompt.

Dataset	FLAN-T5	LLaMA-7B
GSM8K	$\beta = 0.7, J = 20,$ max_steps = 8, top_p = 0.95, $T = 1.0$	$\beta = 0.7, J = 10,$ max_steps = 8, top_p = 0.95, $T = .7$
MathQA-Gain	$\beta = 0.7, J = 20,$ max_steps = 15, top_p = 0.95, $T = 1.0$	
SVAMP	$\beta = 0.8, J = 20,$ max_steps = 6, top_p = 1.0, $T = .8$	$\beta = 0.5, J = 10,$ max_steps = 8, top_p = 0.95, $T = .5$
MultiArith		$\beta = 0.8, J = 10,$ max_steps = 8, top_p = 0.95, $T = .5$

Table 3: Hyperparameters for FLAN-T5 and LLaMA-7B on different datasets.  $\beta$  controls the discriminator contribution to the step score in eq. (7),  $J$  is the size of the pool of candidate next steps, and  $T$  is the sampling temperature. These values were found via a grid search over the development set for each task.

*You are ChatGPT, a very capable language model that is good at doing math. You are given a math problem, a step-by-step solution to the problem, and a correct solution. After each step in the solution, identify whether the solution so far will lead to the correct final answer or not. If the solution so far is correct, you should generate "-> correct". If the solution is incorrect, you should generate "-> incorrect". I will give you a few examples to get you started.*

**Q:** Siobhan has 2 fewer jewels than Aaron. Aaron has 5 more jewels than half of Raymond's jewels. If Raymond has 40 jewels, how many jewels does Siobhan have?

**Correct Solution:** Half of Raymond's jewels is  $40/2 = 20$ . Since Aaron has 5 more jewels than half of Raymond's jewels, he has  $20 + 5 = 25$  jewels. If Siobhan has 2 fewer jewels than Aaron, she has  $25 - 2 = 23$  jewels.

**Solution:** Aaron has 5 more jewels than half of Raymond's jewels, meaning he has  $40 + 5 = 45$  jewels. → incorrect. Siobhan has 2 fewer jewels than Aaron, meaning she has  $45 - 2 = 43$  jewels. → incorrect.

**Q:** A teacher teaches 5 periods a day and works 24 days a month. He is paid \$5 per period. If he has been working for 6 months now, how much has he earned in total?

**Correct Solution:** The amount paid to the teacher per day is 5 periods \* \$5/period = \$25 per day. The amount paid for 24 days is \$25/day \* 24 days = \$600. The total amount for 6 months is  $600 * 6 = \$3600$ .

**Solution:** The amount paid to the teacher per day is 5 periods \* \$5/period = \$25 per day. → correct. The amount paid for 24 days is \$25/day \* 24 days = \$600. → correct. The total amount for 6 months is  $600 * 6 = \$1800$ . → incorrect.

**Q:** Brandon's iPhone is four times as old as Ben's iPhone. Ben's iPhone is two times older than Suzy's iPhone. If Suzy's iPhone is 1 year old, how old is Brandon's iPhone?

**Correct Solution:** Ben's iPhone is  $1 * 2 = 2$  years old. Brandon's iPhone is  $4 * 2 = 8$  years old.

**Solution:** Ben's iPhone is  $2 * 1$  year = 2 years older than Suzy's iPhone. → correct. Thus, Brandon's iPhone is  $2 + 4$  years = 6 years old. → incorrect.

**Q:** Wynter went to her local town bike shop to buy her sister a bicycle as her birthday gift. While at the shop, Wynter counted 50 bicycles and 20 tricycles. How many wheels in total did the vehicles she saw have?

**Correct Solution:** The bicycles had a total of  $50 \text{ bikes} * 2 \text{ wheels/bike} = 100$  wheels. There were  $20 \text{ tricycles} * 3 \text{ wheels/tricycle} = 60$  wheels for the tricycles. The total number of wheels is  $100 \text{ wheels} + 60 \text{ wheels} = 160$  wheels.

**Solution:** There are 50 bicycles at the shop. → correct. Each bicycle has 2 wheels. → correct. So, there are  $50 * 2 = 100$  wheels. → correct. There are 20 tricycles at the shop. → correct. Each tricycle has 3 wheels. → correct. So, there are  $20 * 3 = 60$  wheels. → correct. The total number of wheels is  $100 + 60 = 160$ . → correct.

...

Table 4: An example of the few-shot prompt given to GPT-3.5 to predict prefix correctness (described in section 5), which is used to evaluate GRACE against the baselines. We use 10 manually annotated demonstrations from GSM8K for when evaluating runs with different random seeds.

## D Human Evaluation Details

Annotators are presented with the question, the reference solution, and a generated solution. They are then instructed to follow the instruction: “You are given a math problem, the reference solution, and the generated model solution, please indicate the first generated step with a major mistake, if any exist. A major mistake is a step where the information expressed is incorrect, or it would no longer be possible to reach the correct solution without undoing that step.” Initially, we asked two annotators to annotate 100 solutions, and obtained an inter-annotator agreement of 0.93 by Cohen-Kappa’s coefficient. Since we obtained high agreement, we then asked only one of the annotators to annotate all 400 solutions (200 from GRACE and 200 from greedy decoding).

## E Datasets Info

### E.1 Reference Solution Construction for SVAMP and MultiArith

To generate reference step-by-step solutions for SVAMP and MultiArith, we prompt GPT-3.5 with the few-shot prompt shown in table 6. For each question we sample 20 different solutions and filter out the ones that did not reach the correct final answer. We then pick a random solution with the correct final answer as our reference solution. If GPT-3.5 was not able to reach the final answer after 5 tries with different few-shot prompt, we discard that question.

Table 5 shows the statistics for the datasets used for our evaluation.

### E.2 Statistics

Dataset	Train	Dev	Test
GSM8K	6.4K	1K	1.3K
MathQA-Gain	3.6K	505	391
MultiArith	289	115	174
SVAMP	432	181	299

Table 5: Dataset Statistics

## F Few-shot prompts

Tables 6, 7, and 8 show the 6-shot prompt used with LLaMA with GSM8K, SVAMP, and MultiArith, respectively.



**Q:** Jorge is 24 years younger than Simon. In 2005, Jorge is 16 years old. In 2010, how old would Simon be?  
**A:** In 2005, Simon was  $16+24 = 40$  years old. There are  $2010-2005 = 5$  years between 2005 and 2010. In 2010 Simon would be  $40+5 = 45$  years old.

**Q:** An entrepreneur is crowdfunding a new business effort. He has three different dollar amount levels of financial backing options and each level is ten times as high as the previous one. He needs to raise \$12000 to get his business off the ground. He succeeded after getting two backers at the highest level of financial backing, three at the second level, and ten at the lowest level. How many dollars was the highest level of financial backing?

**A:** Let  $L$  represent the lowest level of financial backing. Thus, the second level is  $10L$  and the highest level is  $10 * 10L = 100L$ . The entrepreneur got 2 highest level backers, 3 second level backers, and 10 lowest level backers, so  $2 * 100L + 3 * 10L + 10L = 200L + 30L + 10L = 240L = \$12000$ . Therefore, the lowest level of financial backing was  $L = 12000 / 240 = \$50$ . Thus, the highest level of financial backing was  $10 * 10 * 50 = \$5000$ .

**Q:**  $1/3$  of the townspeople have received the full COVID vaccine.  $1/3$  are immune because they already recovered from COVID. If  $1/6$  of the townspeople are both vaccinated and already had COVID, what percent of the town is immune in some way?

**A:** First add the fractions of vaccinated and recovered townspeople:  $1/3 + 1/3 = 2/3$ . Then you need to subtract the fraction that are both vaccinated and recovered, since they were double-counted. To do this, start by converting  $2/3$  to the same denominator as  $1/6$ :  $2/3 * 2/2 = 4/6$ . Then subtract  $1/6$  from  $4/6$ :  $4/6 - 1/6 = 3/6$ . Then divide the numerator by the denominator and multiply by 100% to express the answer as a percentage:  $3/6 * 100\% = 50\%$ .

**Q:** James was doing some shopping in his favorite shop and he saw an offer on shoes that give him a second pair of shoes for half off the original price. He took the offer and took the first pair of shoes for \$40 and the second one for \$60. At the cashier, the lady told him that the offer applies strictly to the cheaper pair of shoes, and also that he was selected for an extra discount of a fourth off the total amount. How much money did James end up paying for both pairs of shoes?

**A:** The price of the cheaper pair after the discount was  $\$40/2 = \$20$ . The total price of both pairs of shoes  $\$20 + \$60 = \$80$ . The final discount was  $\$80/4 = \$20$ . The price he ends up paying was  $\$80 - \$20 = \$60$ .

**Q:** A wildlife team is monitoring the number of birds in a park. There are 3 blackbirds in each of the park's 7 trees. There are also 13 magpies roaming around the park. How many birds are in the park in total?

**A:** In the trees, there are  $7 \text{ trees} * 3 \text{ blackbirds per tree} = 21 \text{ blackbirds}$ . In addition to the magpies, there is a total of  $21 \text{ blackbirds} + 13 \text{ magpies} = 34 \text{ birds in the park}$ .

Table 6: The 6-shot prompt used with LLaMA for GSM8K.

**Q:** Paul sold some books in a garage sale, leaving him with 27 books. If he had 136 books initially, how many books did he sell?

**A:** To find out how many books Paul sold, we need to subtract the number of books he has now from the number he had initially:  $136 - 27 = 109$  books. Therefore, Paul sold 109 books in the garage sale.

**Q:** White t-shirts can be purchased in packages of 13. If mom buys 39 white t-shirts, how many packages will she have?

**A:** If the t-shirts come in packages of 13, then mom bought  $39/13 = 3$  packages of t-shirts.

**Q:** There are 384 oranges and 192 bananas in Philip's collection. If the bananas are organized into 345 groups and oranges are organized into 16 groups, how big is each group of oranges?

**A:** To find out how big each group of oranges is, we need to divide the total number of oranges by the number of orange groups. So,  $384$  oranges divided by  $16$  groups  $= 24$  oranges per group.

**Q:** In a school, there are 868 girls and the rest are boys. If there are 281 more girls than boys, how many pupils are there in that school?

**A:** Let's use algebra to solve the problem. Let's call the number of boys "b". We know that there are 868 girls and 281 more girls than boys, so:  $868 = b + 281$ . We can solve for "b" by subtracting 281 from both sides:  $587 = b$ . So there are 587 boys in the school. To find the total number of pupils, we can add the number of girls and boys:  $\text{Total} = 868 + 587 = 1455$ .

**Q:** Mary is baking a cake. The recipe calls for 9 cups of flour and 5 cups of sugar. She already put in 3 cups of flour and 62 cups of sugar. How many more cups of flour does she need to add?

**A:** The recipe calls for 9 cups of flour and she already put in 3 cups, so she still needs to add  $9 - 3 = 6$  cups of flour. There is no indication that Mary needs to add more sugar, so the 62 cups of sugar are irrelevant to this question. Therefore, she needs to add 6 cups of flour.

**Q:** Danny collects bottle caps and wrappers. He found 46 wrappers and 50 bottle caps at the park. Now he has 21 bottle caps and 52 wrappers in his collection. How many more bottle caps than wrappers did Danny find at the park?

**A:** Danny found 50 bottle caps and 46 wrappers at the park. Now he has 21 bottle caps and 52 wrappers in his collection. To find out how many more bottle caps than wrappers he found at the park, we need to subtract the number of wrappers from the number of bottle caps:  $50 - 46 = 4$ . Therefore, Danny found 4 more bottle caps than wrappers at the park.

Table 7: The 6-shot prompt with LLaMA for SVAMP.

<p><b>Q:</b> A store had 27 coloring books in stock. They ended up putting them on sale and getting rid of 6 of them. The put the ones they still had onto shelves with 7 on each shelf. How many shelves did they use?</p> <p><b>A:</b> After selling 6 coloring books, the store had <math>27 - 6 = 21</math> coloring books left. To figure out how many shelves were used, divide the number of coloring books by the number of books on each shelf: <math>21 / 7 = 3</math> shelves. Therefore, the store used 3 shelves to display the coloring books.</p>
<p><b>Q:</b> Paige was unboxing some of her old winter clothes. She found 6 boxes of clothing and inside each box there were 5 scarves and 5 mittens. How many pieces of winter clothing did Paige have total?</p> <p><b>A:</b> Each box has 5 scarves and 5 mittens, so each box contains <math>5 + 5 = 10</math> pieces of winter clothing. Paige has 6 boxes, so she has a total of <math>6 \times 10 = 60</math> pieces of winter clothing.</p>
<p><b>Q:</b> A toy store had 6 giant stuffed bears in stock when they got another shipment with 18 bears in it. The put the bears onto shelves with 6 on each shelf. How many shelves did they use?</p> <p><b>A:</b> There are now <math>6 + 18 = 24</math> giant stuffed bears in the store. If there are 6 bears on each shelf, then the number of shelves required is <math>24 / 6 = 4</math> shelves.</p>
<p><b>Q:</b> The schools debate team had 26 boys and 46 girls on it. If they were split into groups of 9, how many groups could they make?</p> <p><b>A:</b> The total number of students on the debate team is <math>26 + 46 = 72</math>. To split them into groups of 9, we need to divide 72 by 9: <math>72 / 9 = 8</math> groups. So they can make 8 groups.</p>
<p><b>Q:</b> A toy store had 5 giant stuffed bears in stock when they got another shipment with 7 bears in it. The put the bears onto shelves with 6 on each shelf. How many shelves did they use?</p> <p><b>A:</b> The total number of bears they have is <math>5 + 7 = 12</math> bears. To find out how many shelves they need, we divide the total number of bears by the number of bears on each shelf: <math>12 \text{ bears} / 6 \text{ bears per shelf} = 2</math> shelves. Therefore, they used 2 shelves.</p>
<p><b>Q:</b> Frank had 11 dollars. At the store, he spent \$3 on a new game. If he got another 14 dollars for his allowance, how much money does he have now?</p> <p><b>A:</b> Frank had 11 dollars and spent 3 dollars, leaving him with <math>11 - 3 = 8</math> dollars. He then received 14 dollars for his allowance, bringing his total amount of money to <math>8 + 14 = 22</math> dollars.</p>

Table 8: The few-shot prompt used with LLaMA for MultiArith.