MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Discriminator-Guided Chain-of-Thought Reasoning

Muhammad Khalifa

Deep Learning Classics and Trends

March 1st, 2024

Outline

- Introducing chain-of-thought reasoning with LLMs.
- Why LLMs make reasoning errors?
- Post-hoc methods to improve reasoning.
- Introducing Guided Decoding for reasoning (GRACE).
- Results and some analysis.
- Connection to recent search-based techniques.
- Limitations and next steps.

Feel free to interrupt with questions!

Why Reasoning Matters

- A definition I like is *"the ability to construct models from perception, description, and knowledge, to formulate novel but parsimonious conclusions from these models."* [1]
- Complex reasoning is a hallmark of human intelligence.
- We reason about almost everything from grocery shopping to planning a vacation.
- We can’t expect to reach AGI without solving reasoning.

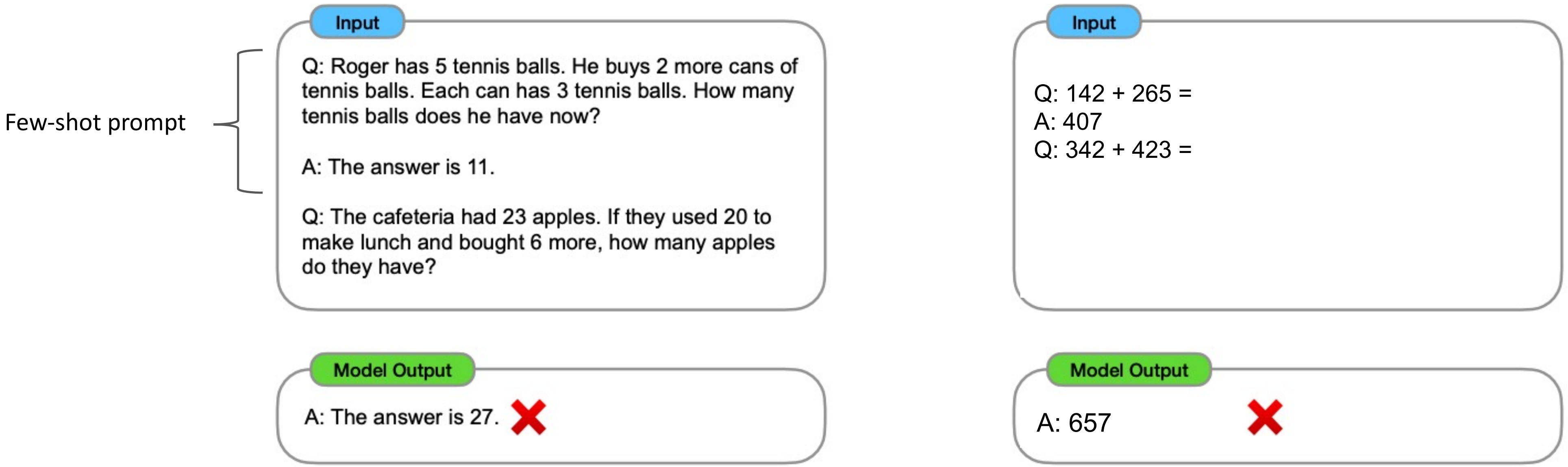
[1] Johnson-Laird, Philip. *How we reason*. Oxford University Press, 2008.

Our Scope

- This talk focuses on a slightly narrow scope of reasoning tasks:
  - The input and output are in natural language.
  - There is a single correct final answer for each given input.
- Many reasoning problems fall under this scope:
  - Mathematical reasoning (e.g., math word problem solving)
  - Arithmetic reasoning (e.g., N-digit addition, division, etc.)
  - Symbolic reasoning (e.g., sorting, theorem proving, etc.)
  - Puzzle solving and game playing (e.g., game of 24).

Few-shot reasoning

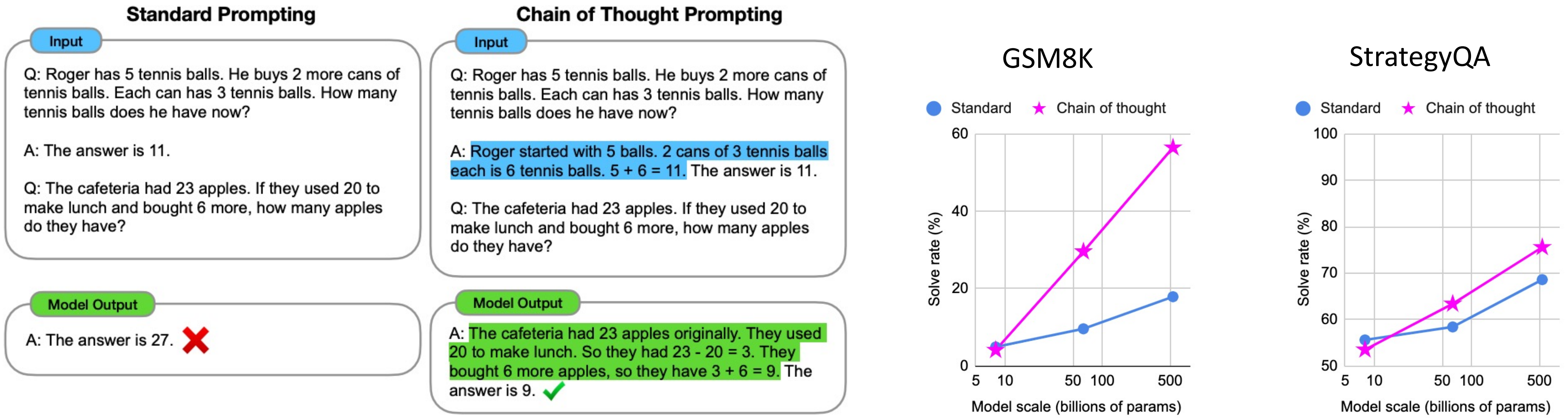
- LLMs were shown to struggle with few-shot reasoning.
- They could not solve tasks such as n-digit addition or math word problems solving in a *single* pass.



Chain-of-thought Reasoning

- The trick was simple: Allow the LM to reason step-by-step!
- Prompt [1,2] or finetune LLMs [3] with chain-of-thought (CoT) solutions.

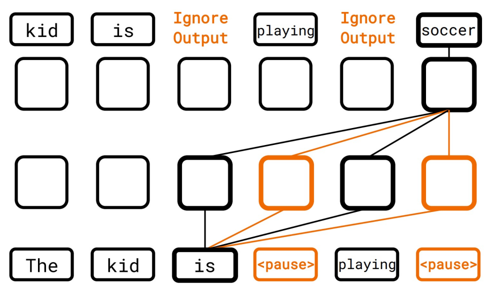
Wei et al., 2022



[1] Nye, Maxwell, et al. "Show your work: Scratchpads for intermediate computation with language models." arXiv preprint arXiv:2112.00114 (2021).
[2] Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." NeurIPS 2022.
[3] Ho, Namgyu, Laura Schmid, and Se-Young Yun. "Large language models are reasoning teachers." ACL 2023.

Why does CoT help?

- It is not immediately clear why CoT reasoning helps.
- Possibly:
  - It allows to spend extra computation per problem.
    - Recent work showed that training with pause tokens helps [1].
  - Decomposes the input problem into easier sub problems.
    - Each step can be thought of as solving a subproblem.
  - Serves as a working memory for the LLM.
    - LM does not need to store intermediate variables internally.
- Likely a combination of all these, but more analysis is needed.



Sachin, et al., 2023

[1] Goyal, Sachin, et al. "Think before you speak: Training Language Models With Pause Tokens." *arXiv preprint arXiv:2310.02226* (2023).

Defining correctness

- The final answer could be correct while the reasoning is wrong!

Input

Q: Given a 7-day week, how much does Alex charge for 2 weeks of tutoring if she charges $12 per day?

FLAN-T5 Output

A: There are 7 days in a week and Alex charges $12 per day so she charges 12*7 = $ 84 for one week of tutoring.
There are 2 weeks in a month and she charges $84 per week so she charges 2*84 = $ 168 for two weeks of tutoring.
The answer is 168.

- CoT exposes the model reasoning allowing to verify correctness.
- We say that a model has reasoned correctly if:
  - Each of the intermediate steps does not include errors (e.g., logical, factual or computational.)
  - The final answer is correct.
- Not a formal definition but should do for our purposes.

CoT Errors

- LLMs still produce incorrect steps along the chain.
- Why do reasoning errors happen?
- An artifact of both LLM *training* and *inference.*
- Inference: common decoding strategies for CoT are greedy/beam search.
  - They optimize for sequence likelihood according to the LM.
- But LLMs are poorly calibrated [1,2].
  - Training: internet data is typically of low quality.
- Thus, the most probable reasoning step is not necessarily correct.

[1] Holtzman, Ari, et al. "Surface form competition: Why the highest probability answer isn't always right." *EMNLP 2021.*
[2] Zhao, Zihao, et al. "Calibrate before use: Improving few-shot performance of language models." *International Conference on Machine Learning*. PMLR, 2021.

High Probability != Correct

- LLMs can assign a *high* probability to *incorrect* steps and vice versa.
- Decoding CoT solutions with standard decoding strategies will produce incorrect chains.

CoT prompting with LLaMA-13B

Question

I have 10 liters of orange drink that are two-thirds water and I wish to add it to 15 liters of pineapple drink that is three-fifths water. As I pour it, I spill one liter of the orange drink. How much water is in the remaining 24 liters?

Incorrect steps are assigned higher avg. token prob than the correct one!

Post-hoc methods

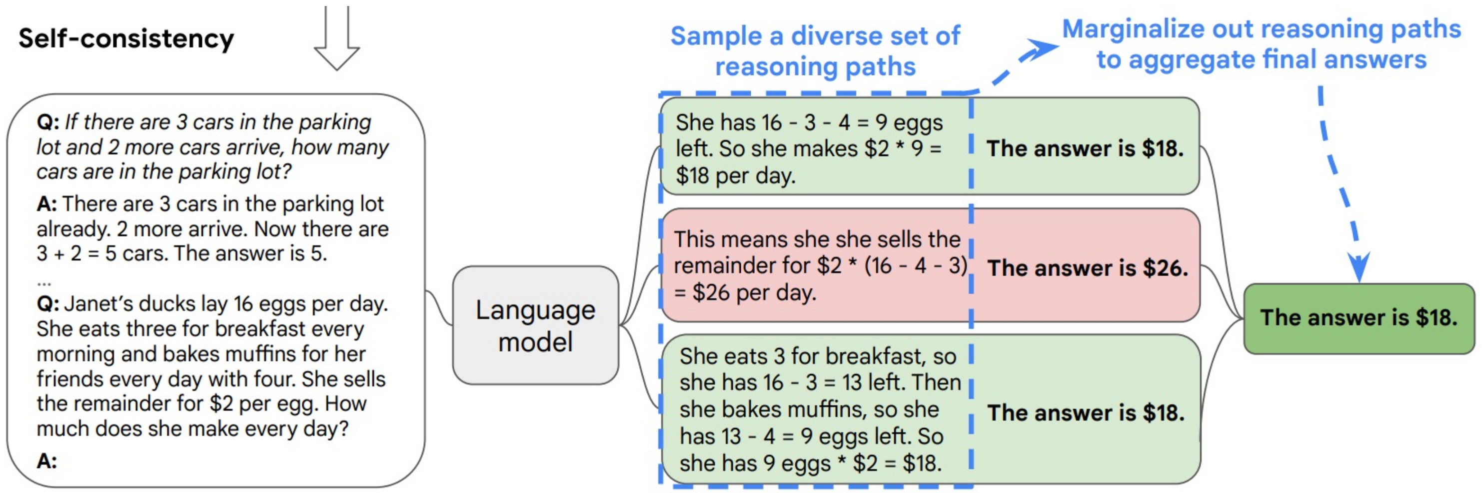
- Techniques to workaround this issue rely on sampling *multiple* full chains then ranking them based on correctness.
- Two main techniques: self-consistency [1] and verifiers [2].
- **Ranking criterion:**
  - Final answer frequency: chains with more frequent answers are more likely to be correct.
  - Chain "correctness": Train a model to differentiate correct/incorrect chains labeled based on final answer [2].

[1] Wang, Xuezhi, et al. "Self-consistency improves chain of thought reasoning in language models." *ICLR 2023.*
[2] Cobbe, Karl, et al. "Training verifiers to solve math word problems." *arXiv preprint arXiv:2110.14168* (2021).

Issues with post-hoc methods

- There are at least two issues with post-hoc methods:
  - **Miscalibration:** They rely on sampling from underlying miscalibrated LM distribution.

Wang et al., 2023

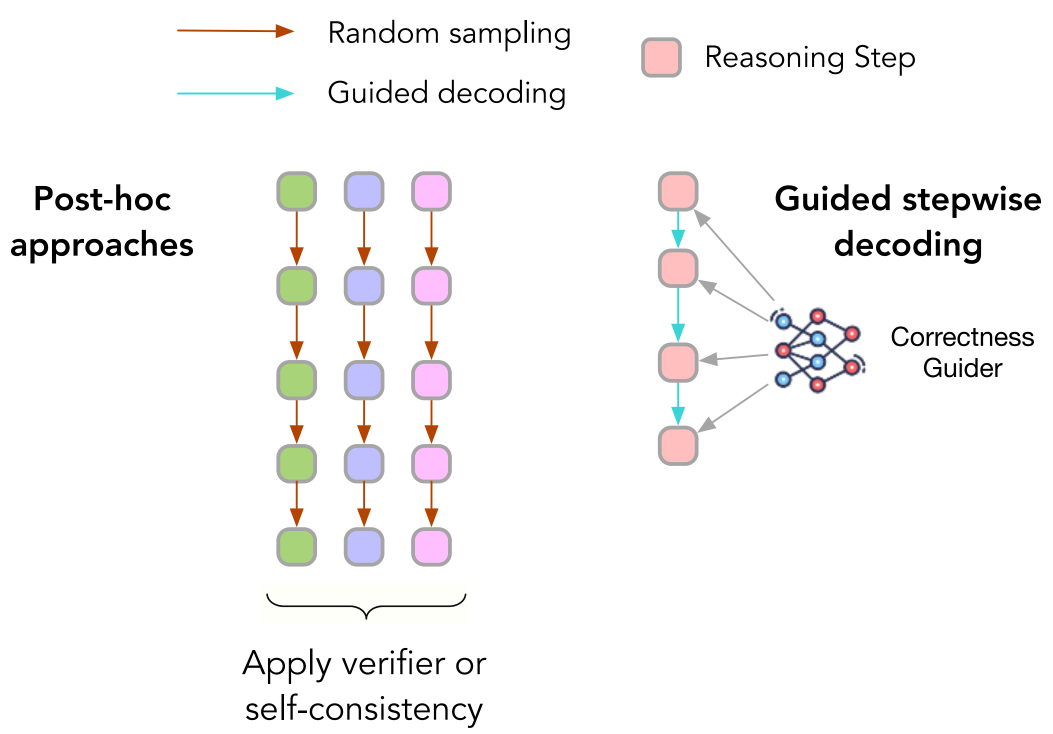


  - **No control over the decoding:** Applied on top of full chains after decoding is finished.

→ They still make mistakes and are highly sample-inefficient:
- Many samples are needed to *run into* correct chains (if at all)

Guided decoding can help

- We propose guided stepwise decoding to address issues with post-hoc methods:
  - **Guided:** Recalibrate step scores based on correctness.
  - **Stepwise:** finer-grained control at the step rather than the chain level.



Formalization

Given a problem q and a correct solution prefix $s_1, s_2, \ldots, s_{t-1}$, let's assume access to a discriminator model D that outputs a real-valued correctness score $D(q, s_{1:t-1}, s_t)$.

Let c be a binary variable indicating correctness of the generated step.

We want to sample next step $s_t \sim p(.|s_{1:t-1}, c, q)$

We can write

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

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

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

Replace with probability according to LM

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

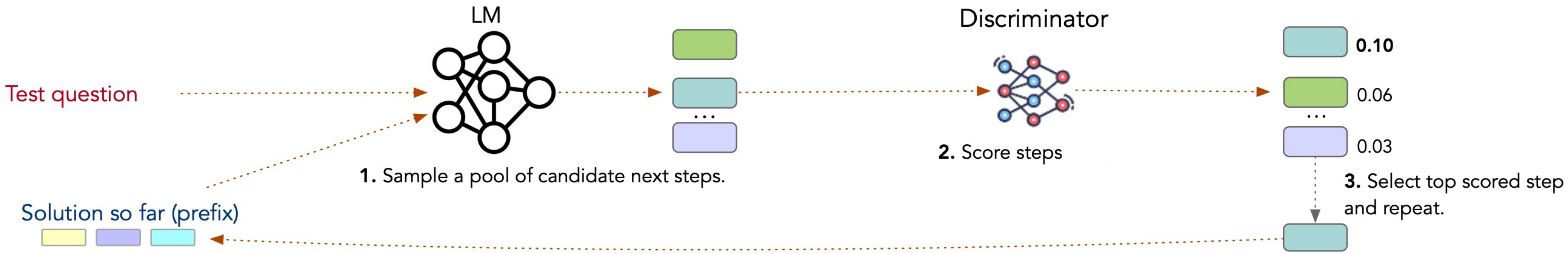
Based on our definition of $D(q, s_{1:t-1}, s_t)$ and since we assume the prefix to be correct

GRACE Decoding

- At each time step:
  - Sample a set of J candidate next steps from the LM $\{s_t^{(1)}, s_t^{(2)}, \dots, s_t^{(J)}\}$.
  - Score each step $s^{(i)}$ using:

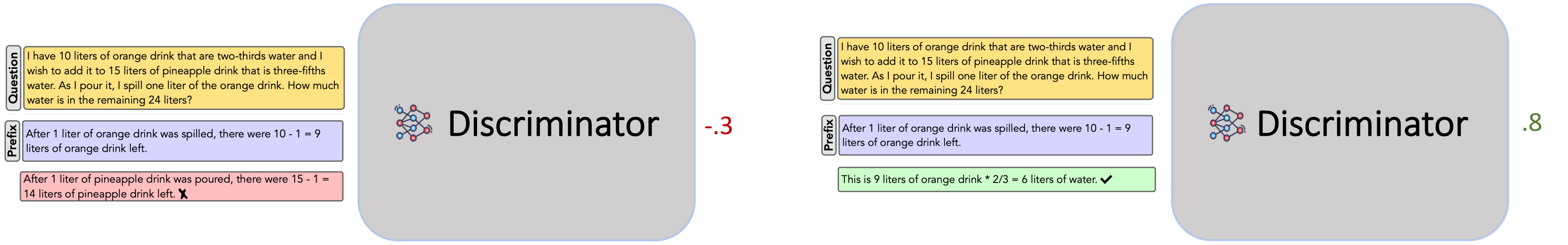
$$(1-\beta)\log p_{\mathrm{LM}}\left(s^{(i)} \mid q, s_{1:t-1}\right) + \beta D\left(q, s_{1:t-1}, s^{(i)}\right)$$

  - Select top scored step.



Discriminator Learning

- $\mathrm{D}(\mathrm{q}, \mathrm{r}, \mathrm{s_t})$ should be high for correct steps and low for incorrect ones (r is the prefix for brevity).
- How do we learn $\mathrm{D}(\mathrm{q}, \mathrm{r}, \mathrm{s_t})$?
- Goal: $D(q, r, s^+) > D(q, r, s^-)$ for all correct and incorrect steps s^+ and s^-



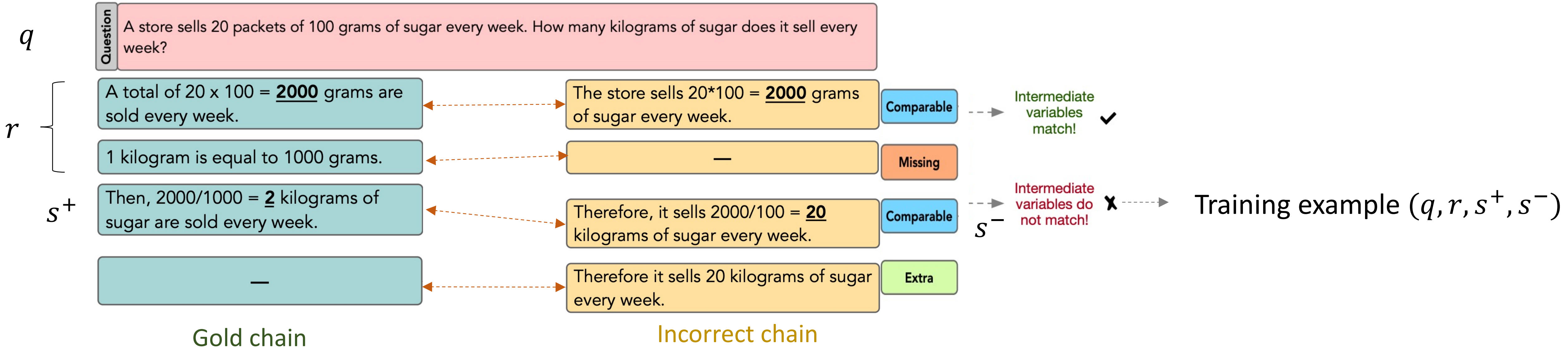
Discriminator Learning (contd.)

- If we have pairwise examples of the form (q, r, s^+, s^-), we can train D with a contrastive objective.
- Max margin objective:
  - Minimize $\sum_{(q,r,s^+,s^-)\in E} \max\{0, D(q,r,s^-) - D(q,r,s^+) + \zeta\}$
- How do we get these pairwise examples?
  - Human annotation:
    - Has been used to train process-reward models PRMs [1, 2].
    - Expensive!
  - Can we leverage gold chains?
    - Automatic Alignment incorrect chains with gold chains to extract pairwise examples.

[1] Lightman, Hunter, et al. "Let's Verify Step by Step." *arXiv preprint arXiv:2305.20050* (2023).
[2] Uesato, Jonathan, et al. "Solving math word problems with process-and outcome-based feedback." *arXiv preprint arXiv:2211.14275* (2022).

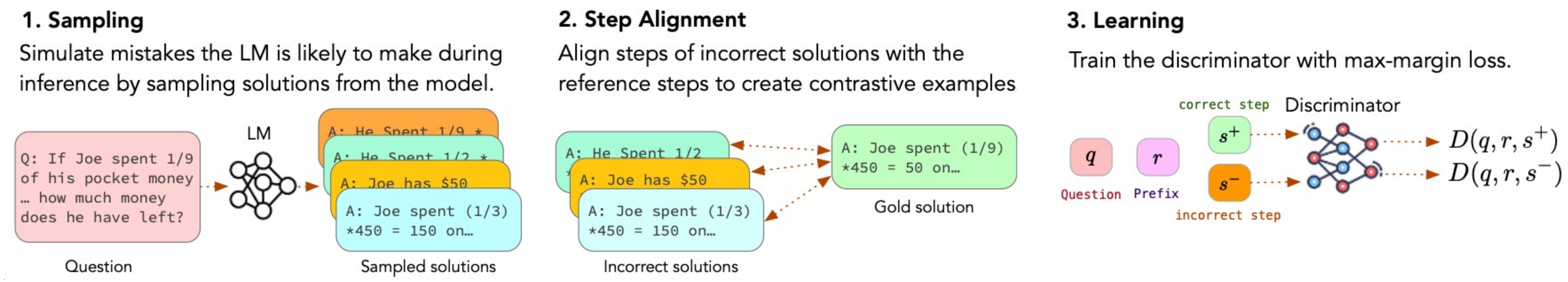
Chain Alignment

- Given an incorrect chain and a gold chain, we can find a *minimum-cost* alignment between two.
- **Cost:** Total cosine distance between aligned steps measured using ROSCOE[1].
- We use the Dynamic Programming-based Needleman-Wunsch algorithm [2].



[1] ROSCOE: A Suite of Metrics for Scoring Step-by-Step Reasoning. ICLR 2023.
[2] Likic, Vladimir. "The Needleman-Wunsch algorithm for sequence alignment." *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne* (2008): 1-46.

Summing it up



Similar reward model training in RLHF!

Experimental Setup

- **Backbone models:** FLAN-T5, LLaMA-7B, LLaMA-13B.
- **Baselines:** greedy, LM-only scoring, vanilla self-consistency (SC) and verifiers.
- For each task, we roughly sample 100K solutions for discriminator training.
- We use a T5-Large encoder as the discriminator (20X, 38X smaller than LLaMA-7B and LLaMA-13B.)
- **Tasks:**
  - **math:** GSM8K, MathQA-Gain, SVAMP, and MultiArith.
  - **symbolic:** Coin Flip and Tracking Shuffled Objects from Big-Bench Hard.

Effect on Final Answer Accuracy

	FLAN-T5$_{\text{LARGE}}$ *(Fine-tuned)*			LLaMA$_{\text{7B}}$ *(few-shot prompted)*		
	GSM8K	**SVAMP**	**MathQA-Gain**	**GSM8K**	**SVAMP**	**MultiArith**
Greedy decoding	26.9	54.5	76.5	12.9	32.8	54.0
Random sampling						
Vanilla SC	33.3	61.8	78.9	20.7	52.4	78.9
Solution verifier	20.5	45.9	83.7	9.60	26.1	46.4
LM-only score ($\beta = 0$)	27.5	53.1	52.9	12.5	39.6	57.9
Guided sampling						
GRACE	34.3 (+7.4)	66.2 (+11.7)	84.1 (+6.0)	16.2 (+3.30)	49.7 (+17.3)	84.9 (+30.9)
GRACE w/ SC	**36.3** (+3.0)	**68.6** (+6.80)	**84.4** (+0.7)	**30.9** (+10.2)	**55.6** (+3.20)	**94.6** (+15.7)

Final answer accuracy (math reasoning)

GRACE outperforms almost all baselines.

GRACE + Self-consistency is best across the board → Shows the value of guided compared to random sampling.

Effect on Intermediate Reasoning

- Final answer accuracy does tell the full story.
- We evaluate whether GRACE improves correctness of generated chains.
- We measure prefix correctness via GPT-3.5-turbo.
- We measure trace error [1] (% of correct solutions with at least one major mistake) via humans and GPT-3.5-turbo.

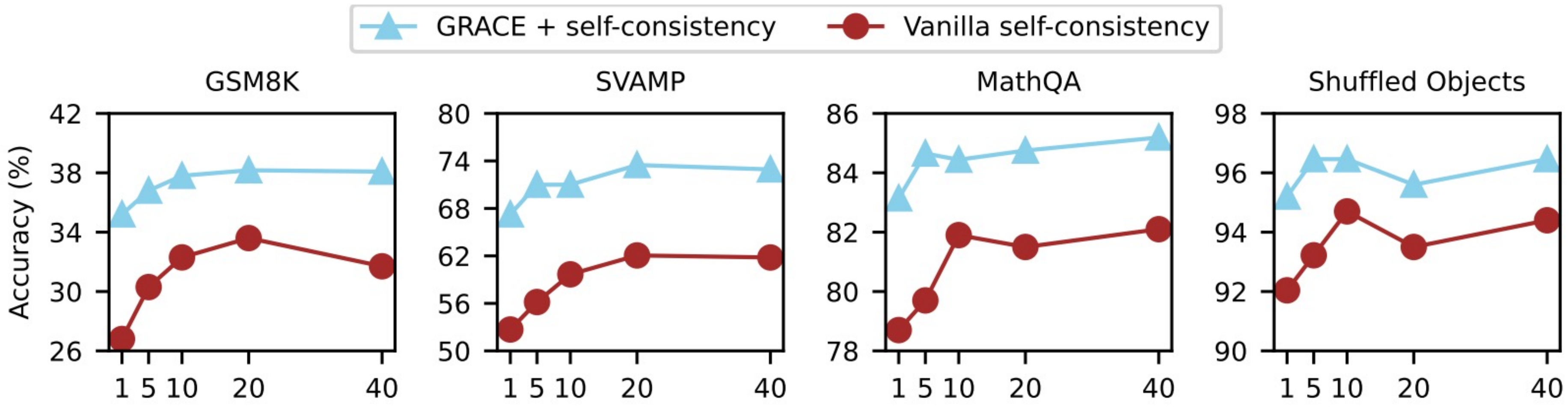
	Prefix Correctness-LLM (↑)	**LLM-TE (↓)**	**Human-TE (↓)**
Greedy decode	46.5	7.0	9.0
Vanilla SC	51.0	9.8	-
GRACE	53.5 (+7.0)	5.2 (-1.8)	5.0 (-4.0)
GRACE w/ SC	54.8 (+3.8)	6.6 (-3.2)	-

GRACE improves prefix correctness and reduces trace error by 44% compared to greedy

[1] Uesato, Jonathan, et al. "Solving math word problems with process-and outcome-based feedback." *arXiv preprint arXiv:2211.14275* (2022).

Sample Efficiency

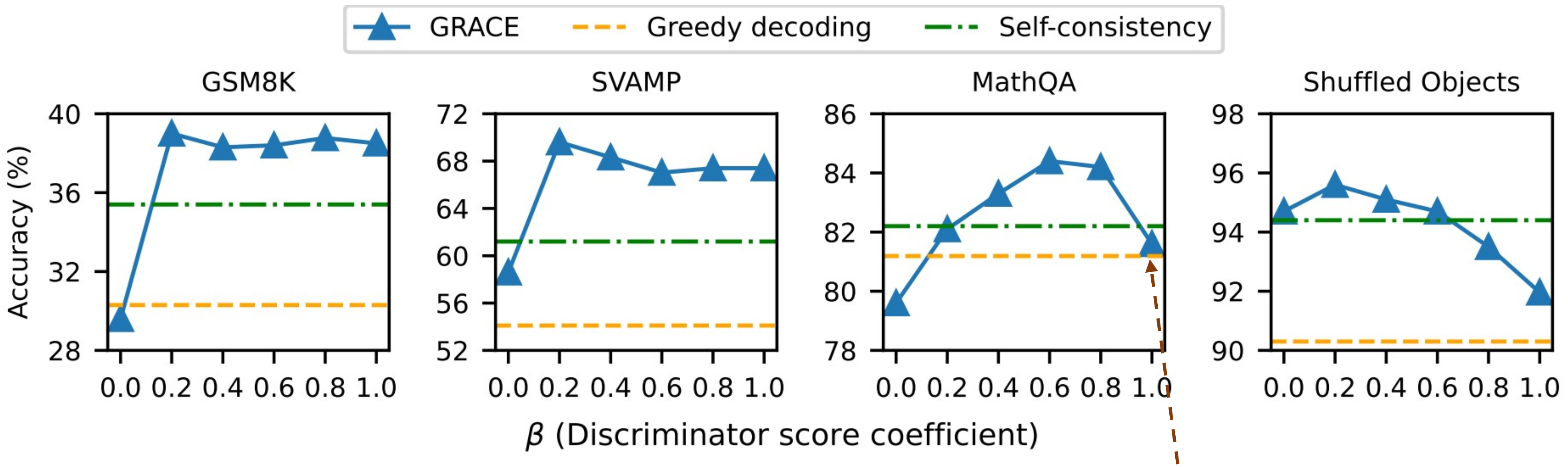
- As discussed earlier, post-hoc methods require plenty of samples to reach correct answers because of reliance on random sampling.
- We compare guided sampling + SC vs. random sampling + SC.



GRACE reaches substantially better accuracy with significantly fewer samples.

Analysis: step scoring coefficient

Effect of varying β In $(1-\beta)\log p_{\mathrm{LM}}\left(s^{(i)} \mid q, s_{1:t-1}\right)+\beta D\left(q, s_{1:t-1}, s^{(i)}\right)$



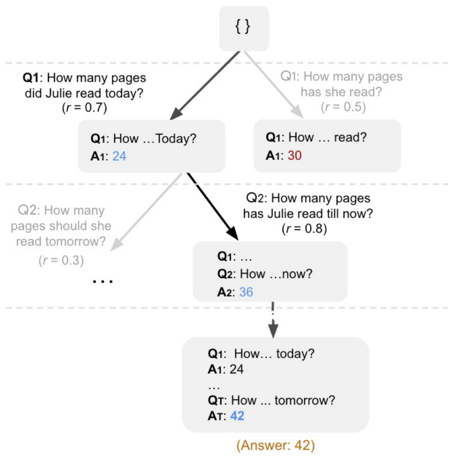
Performance drops when β=1 showing that we still want to incorporate p_{LM} representing the reasoning abilities of the LM

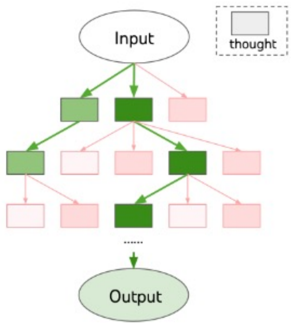
Takeaways

- LMs can easily assign high likelihood to incorrect reasoning.
- We can mitigate this by recalibrating reasoning step likelihoods based on correctness.
- Guided reasoning can improve both sample efficiency and correctness of reasoning.
- A small but specialized model can provide guidance for a much larger model.

Connection to recent work

- There's plenty of recent approaches on applying search methods on top of LLMs.
- A search method is used to traverse the solution space, guided by some scoring function or reward [1,2,3].
- Three main dimensions to inference-time techniques for reasoning:
  - **Step scoring:** learned vs. prompting-based.
  - **# of parallel chains:** 1 vs many (tree [1,2,3] or even graph [4])
  - **Search algorithm:** Greedy vs. more advanced like A* or MCTS.





(d) Tree of Thoughts (ToT)

[1] Hao, Shibo, et al. "Reasoning with language model is planning with world model." *EMNLP 2023.*
[2] Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models." NeurIPS 2023.
[3] Zhou, Andy, et al. "Language agent tree search unifies reasoning acting and planning in language models." *arXiv preprint arXiv:2310.04406* (2023).
[4] Besta, Maciej, et al. "Graph of thoughts: Solving elaborate problems with large language models." *arXiv preprint arXiv:2308.09687* (2023).

Limitations

- While inference-time techniques relief the need to train the LLM, they do not come without limitations.
- **Latency:** The search process makes inference extremely slow.
- **API cost:** LLM-based scoring requires tens of API calls per input.
- GRACE requires access to correct chains to train the discriminator.
- Inference-time methods are upper-bounded by the performance of the underlying LLM.
- The alignment algorithm is sensitive to the step order even if two steps can be done in any order.

What’s next?

- Use the learned scoring function to train the LM akin to RLAIF [1].
- Improving search efficiency:
  - Requires many API Calls (for self-evaluation and exploration)
  - Slow when the search space is large
- Training a low-shot discriminator that needs few/no gold chains:
  - Training a dedicated scoring function is still relevant.
  - LLMs were shown to fail at identifying their own errors [3].
- Introducing step order invariance to the alignment.
- Please reach out if you want to discuss more!

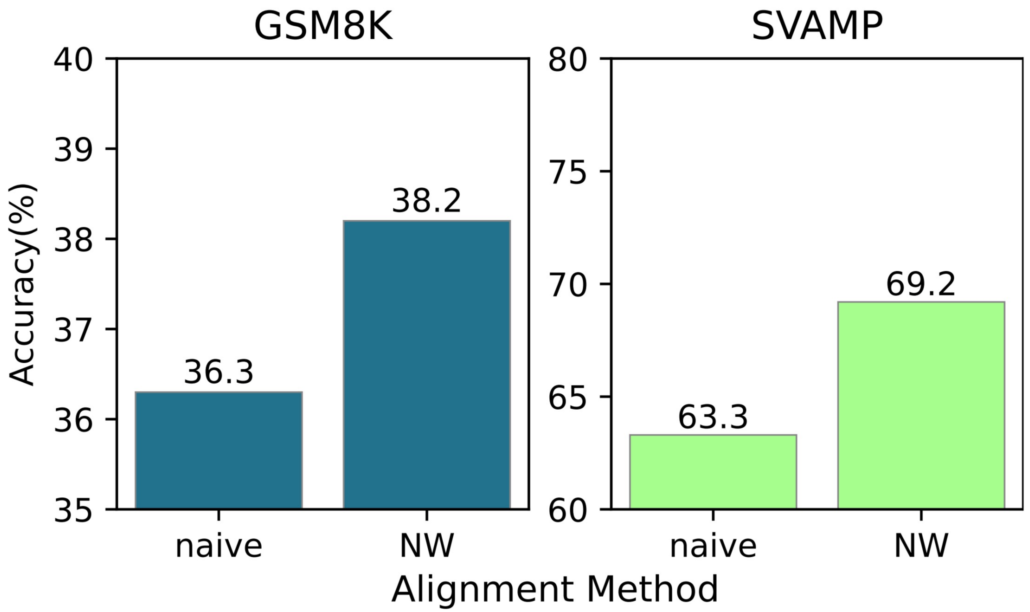
[1] Lee, Harrison, et al. "Rlaif: Scaling reinforcement learning from human feedback with ai feedback." *arXiv preprint arXiv:2309.00267* (2023).
[2] Li, Jingjing, et al. "Unsupervised text generation by learning from search." *NeurIPS* 2020.
[3] Huang, Jie, et al. "Large language models cannot self-correct reasoning yet." *arXiv preprint arXiv:2310.01798* (2023).

Thank you!

Backup slides

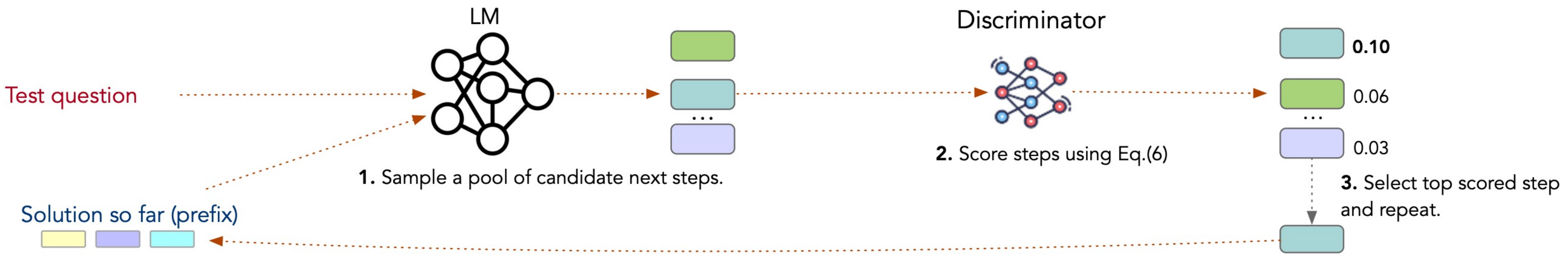
NW Alignment ablation

- How useful is our Needleman-Wunsch (NW) alignment for discriminator training?
- We compare NW alignment to naive alignment.
- The naive version simply aligns corresponding steps in correct and incorrect chains.



NW alignment outperforms the naïve version

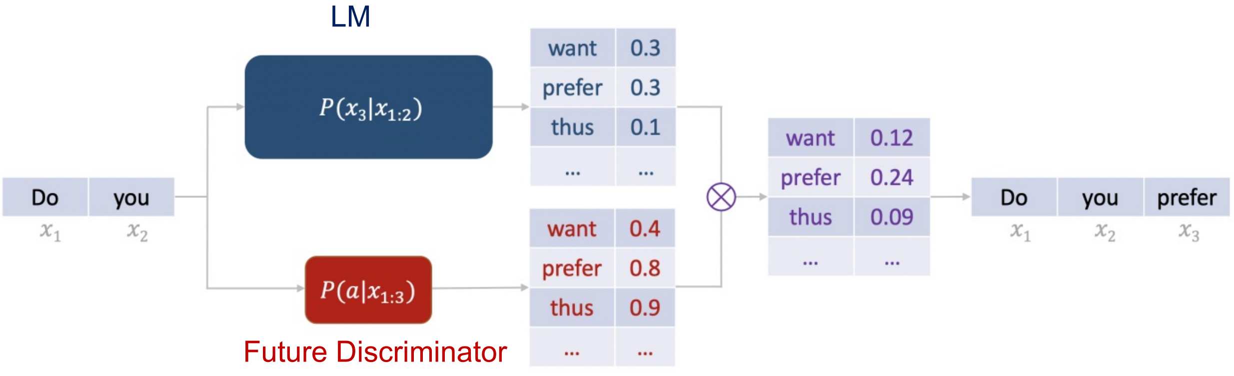
Token efficiency



- If a step is on average 20 tokens and each chain has 5 steps.
- If number of candidate steps for GRACE is J = 10.
- GRACE would sample 20 * 10 * 5 = 1000 tokens per chain.
- Self-consistency with N=40 would sample: 20 * 5 * 40 = 4000 tokens.
- GRACE requires 0.25x as many tokens as SC.

Connection to Controlled Generation

- GRACE is inspired by the controllable generation approach FUDGE [1].
- FUDGE uses a *future* discriminator to adjust token log probs.
- Two main distinctions:
  - FUDGE's discriminator looks at the future, ours looks at both present and past.
  - FUDGE operates at the token level, GRACE at the step-level (correctness of a single token is meaningless).



FUDGE (Yang et al., 2021)

[1] Yang, Kevin, and Dan Klein. "FUDGE: Controlled text generation with future discriminators." *NAACL 2021*