

Reasoning Optimization through Checkpoint DPO

2026. 01. 22.

박찬희

바야흐로 Reasoning LM의 시대

o1의 등장

- 자체적으로 CoT를 수행하는 reasoning language model 개념 등장
- Context length가 증가할수록 성능이 증가하는 inference-time scaling의 개념
- 수학, 코딩 등 논리 연산에서 압도적인 성능 기록

그러나 그 이면에는?

- 너무나 큰 연산 비용... overthinking 이슈
- 추론 불안정성 => 재생산성이 떨어짐
- 모델 성능이 Pass@64, Pass@128 등 천차만별

We suggest using Temperature=0.6, TopP=0.95, TopK=20, and MinP=0.

“RLM은 최적화가 덜 된 게 아닌가?”

Reasoning Optimization의 방향성

1. Token Efficiency (depth scaling)

- Overthinking => 추론 탐색 시 쉬운 문제도 너무 길게 생성하는 현상
- 모든 문제를 어렵게 풀 필요는 없다 => 빨리 답할 수 있는 문제는 빠르게 풀기
- Limitation : 길게 고민해야 하는 문제도 짧게 끝내버리면?

2. Trajectory Optimization (breadth scaling)

- 어려운 문제를 풀려면 실제로 여러 가지 시도(확장)를 해봐야 한다.
- GRPO : rollout 후 최적 경로로 학습
- 짧은 쿼리에 비해 trajectory의 길이는 천차만별
- Limitation : 정답만 맞추면 그만? 길이 몰타기로 인한 reward hacking

Reasoning Optimization의 방향성

3. Mid-reasoning Optimization (중간 타협점)

- Step DPO류 : reasoning step 하나하나 판단해서 필요한 step만 생성하도록 학습
- 이 과정에서 바로 답을 낼 수 있는 문제는 빠르게 추론하도록 최적화
- Limitation : 높은 annotation 비용, Instruct 모델 위주의 실험

Reasoning Optimization도 막바지인가?

- 이미 관련 연구는 포화 상태
- 그러나 아직 RLM 패러다임은 건재함...
- 다음 세대의 모델이 나오기 전까지 무엇을 해야 하는가?

Don't Think Longer, Think Wisely: Optimizing Thinking Dynamics for Large Reasoning Models

Sohyun An¹, Ruochen Wang¹, Tianyi Zhou², Cho-Jui Hsieh¹

¹University of California, Los Angeles

²University of Maryland, College Park

sohyun0423@cs.ucla.edu tianyi@umd.edu chohsieh@cs.ucla.edu

Tackling Overthinking Issue...

- 불필요하게 과도한 추론 경로 생성 => 연산 낭비 및 성능 저하
- 가설 : 추론 전략을 동적으로 채택하는 능력이 부재
- **DTO** (Dynamic Thinking pattern Optimization)
 - 평균 5,000 => 3,000 token의 토큰 수 절약
 - 약 12%의 성능 향상

=> 토큰 수도 줄이고, 성능도 높이고 ... **NIPS2025 Poster**

DTO: Constructing Optimal Reasoning Trajectories

(1) Identify the appropriate point at which the reasoning should be finalized

- 먼저 추론 path를 "Wait", "Alternatively"와 같은 언어적 표지로 truncate
- 각 thinking step마다
 - "... Wait, I suddenly got the final answer to the whole problem. Final Answer: \boxed{"**
로 정답 유도 프롬프트 주입 후 생성되는 정답의 확률을 측정
- 해당 시점에서 정답을 맞출 확률이 충분히 높다면 exit point임.
- 기준을 만족하는 첫 지점까지의 추론 path에 더해,
"Hmm, I think this is enough to derive the final answer."
를 더해 완결된 형태의 reasoning path를 만들어줌.

DTO: Constructing Optimal Reasoning Trajectories

(2) Prune intermediate thinking patterns that do not meaningfully contribute to the reasoning objective

- 완결된 thinking path 내에서도 정답을 추론하는 데 도움이 되지 않는 정보를 제거하는 과정
- 별개 LLM에게 (Llama-3.3-70B-Instruct)
 - 1) 틀린 정보를 포함하는지
 - 2) 해당 path가 추가적인 의미를 제공하는지
 - 3) Off-topic이거나 중복인지판단 후, 그렇지 않은 path는 모두 제거.
- 마지막으로 이렇게 정제된 path로 정답을 유도했을 때 정답인 샘플만 학습에 활용

Post-hoc Trajectory Optimization

Model : DeepSeek-R1-Distill-Qwen-1.5B / DeepScaleR-1.5B-Preview

Data : Math training set 5,000, 4 responses per query

Result

- 각각 attention FLOPs 47%, 40% 감소
- 이 중 오답 샘플의 경우, 각각 15.6%, 7.8%가 정답으로 전환됨

그러나 지금까지의 과정은 post-hoc으로 reasoning path를 수정한 결과임.

그렇다면 학습은?

Preference Optimization Towards Optimal Reasoning Behaviors

- (1) Chosen : 쿼리별 4개 응답 수집 후 가장 짧은 correct sample 채택 후 DTO 수행
- (2) Reject : 최적화되지 않은 가장 긴 trajectory 채택
- (3) SimPO로 학습

$$\mathcal{L}_{\text{SimPO}}(\pi_{\theta}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}'} \left[\log \sigma \left(\frac{\beta}{|y_w|} \log \pi_{\theta}(y_w | x) - \frac{\beta}{|y_l|} \log \pi_{\theta}(y_l | x) - \gamma \right) \right],$$

=> $|y_w|$ 에 대한 정규화 => 길이 조정에 더욱 초점을 맞춘 학습

Preference Optimization Towards Optimal Reasoning Behaviors

Table 1: **Comparison with other methods on the DeepSeek-R1-Distill-Qwen-1.5B model.** We evaluate the effectiveness of our method using a dynamic optimization framework and a preference optimization technique applied to DeepSeek-R1-Distill-Qwen-1.5B by comparing it against existing methods.

Method	MATH			GSM8K			Gaokao		
	Acc. (↑)	#Tokens (↓)	Eff. (↑)	Acc. (↑)	#Tokens (↓)	Eff. (↑)	Acc. (↑)	#Tokens (↓)	Eff. (↑)
Instruct ver. [37]	76.36	555.16	N/A	85.37	315.44	N/A	65.13	575.86	N/A
Baseline	79.80	3543.44	1.000	82.13	1382.99	1.000	66.62	3725.16	1.000
Fast Prompt	81.17	3354.99	1.074	85.14	1894.73	0.757	69.68	3634.30	1.072
SFT	81.28	3180.10	1.135	80.12	933.89	1.445	67.34	3245.37	1.160
O1-Pruner [17]	82.31	2593.06	1.409	80.67	669.41	2.029	66.69	2827.81	1.319
DAST [27]	83.35	2817.94	1.313	84.02	1174.89	1.204	69.42	3058.96	1.269
FCS + Ref. [4]	84.72	2548.55	1.476	84.29	1080.19	1.314	71.30	2750.35	1.450
DTO (Ours)	85.48	1936.19	1.960	83.91	844.18	1.674	72.66	2137.59	1.901

Method	AMC2023			AIME2024			AIME2025		
	Acc. (↑)	#Tokens (↓)	Eff. (↑)	Acc. (↑)	#Tokens (↓)	Eff. (↑)	Acc. (↑)	#Tokens (↓)	Eff. (↑)
Instruct ver. [37]	54.75	786.43	N/A	11.22	956.11	N/A	8.11	887.40	N/A
Baseline	58.25	5338.54	1.000	21.44	7359.24	1.000	18.89	7236.66	1.000
Fast Prompt	61.00	5073.15	1.102	21.56	7261.19	1.019	20.33	7137.90	1.091
SFT	61.08	5030.76	1.113	23.78	7151.84	1.141	18.44	7122.97	0.992
O1-Pruner [17]	65.50	4370.83	1.373	21.78	7015.30	1.066	17.67	6742.34	1.004
DAST [27]	66.58	4590.91	1.329	24.00	7077.45	1.164	19.78	6846.85	1.107
FCS + Ref. [4]	68.92	4166.60	1.516	24.44	6698.77	1.252	20.67	6545.62	1.210
DTO (Ours)	70.25	3376.98	1.907	28.00	5877.44	1.635	21.11	5689.38	1.421

$$\eta = \frac{\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta^*}(x)}[\mathcal{P}(y)]}{\mathbb{E}_{x \sim \mathcal{D}, y_0 \sim \pi_{\theta}(x)}[\mathcal{P}(y_0)]} \cdot \frac{\mathbb{E}_{x \sim \mathcal{D}, y_0 \sim \pi_{\theta}(x)}[\mathcal{C}(y_0)]}{\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta^*}(x)}[\mathcal{C}(y)]}$$

Conclusion

(1) Truncation 방법

- Reasoning 모델의 독특한 특징으로 Wait, Alternatively와 같은 "사고 토큰"을 생성한다는 점
- 이를 기반으로 trajectory를 분절하고 조절하는 학습 및 추론 방법론이 높은 효과성을 보여줌

(2) 효과적으로 길이를 줄이면서 성능도 높은 점

- Accuracy가 오른 것은 추론 길이가 짧아진 것 + 헤매는 지점이 사라진 것
- 즉, token efficiency와 performance가 별개의 항목이 아니라는 것을 실험으로 보여줌

(3) 아쉬운 점

- 학습을 8k 토큰 기준으로 수행
- A6000 4장으로 모든 학습을 진행할만큼 가성비가 좋긴 하지만,
AIME24/25 같은 벤치마크는 long-trajectory가 실제로 유효한 전략일 수 있음.

=> 쉬운 문제만 타겟한 거 아님?

STEP-DPO: STEP-WISE PREFERENCE OPTIMIZATION FOR LONG-CHAIN REASONING OF LLMs

Xin Lai¹ **Zhuotao Tian²** **Yukang Chen¹** **Senqiao Yang²** **Xiangru Peng²** **Jiaya Jia^{1,3}**

¹The Chinese University of Hong Kong ²Harbin Institute of Technology (Shenzhen) ³SmartMore

Background

DPO의 한계

- long-chain으로 갈수록 이점이 줄어든다 => 하나의 path를 통째로 보기 때문임
Models fine-tuned with vanilla DPO cannot pinpoint detailed errors in incorrect answers, hindering the improvement of reasoning abilities
- Step-DPO는 자체 생성한 데이터로 (in-distribution) GPT4나 human 학습보다 뛰어난 결과 (ood)를 보임
- 10K 데이터쌍으로 500 Step만 해도 MATH가 3% 올랐다

Step DPO

- 학습 샘플을 단일 trajectory가 아니라 각각의 개별적인 step으로 인식하여 매 생성 기로에서 오답이 무엇 인지를 학습할 수 있음.

10K Dataset with annotated steps

prompt:

Determine the least positive integer x for which the sum of x and 4609 gives a remainder of 2104 when divided by 12.

initial_reasoning_steps:

Let's think step by step.

Step 1: We know that $x + 4609 \equiv 2104 \pmod{12}$.

Step 2: We can rewrite the given congruence as $x \equiv 2104 - 4609 \pmod{12}$.

Step 3: Calculating $2104 - 4609 = -2505$.

Step 4:

chosen_step:

Now we need to find the equivalent positive integer for $-2505 \pmod{12}$.

To do this, we divide -2505 by 12 and find the remainder. $-2505 \div 12 = -208$ remainder -9 .

However, we want a positive remainder, so we add 12 to -9 to get a positive remainder. $-9 + 12 = 3$.

So, $-2505 \equiv 3 \pmod{12}$.

rejected_step:

We want to find the positive integer x , so we add 12 until we get a positive number. $-2505 + 12 \cdot 210 = 30$.

Figure 5: An example of preference data sample for Step-DPO.

In-Distribution Data Construction

모델이 학습하려면 스스로 생성한 패턴에서 출발해야 한다!

1. Error Collection

- "Let's think step by step. Step 1:"로 단계별 생성 유도
- 처음 오답이 발생하는 지점의 데이터 수집

2. Step Localization

- 각 step을 verify하는 단계. 여기서는 사람이 보거나 GPT-4를 사용.
- 처음으로 error가 발생하는 step을 찾아 표기

3. Rectification

- error 이전 스텝부터 응답을 샘플링해서 맞추는 응답을 수집

Experiment

Table 3: Performance comparison between DPO and Step-DPO. We use only 5K data for training in this ablation study.

Model	Qwen2-7B-SFT	Qwen2-7B-SFT + DPO (5K)	Qwen2-7B-SFT + Step-DPO (5K)
MATH (%)	54.8	55.0	55.8
Model	Qwen2-72B-SFT	Qwen2-72B-SFT + DPO (5K)	Qwen2-72B-SFT + Step-DPO (5K)
MATH (%)	61.7	62.5	64.1

Conclusion

(1) In-distribution Data Generation

- 상위 모델의 생성 패턴을 따라가는 Distillation과 다르게, in-distribution 기반 RL은 (이론상) 모델의 포텐셜을 더욱 잘 이끌어낼 수 있음.

(2) Step-wise truncation

- Chosen/reject를 데이터 샘플 단위가 아니라, 동일 path 내의 특정 부분으로 삼은 것.
- 즉, [Query / Step 1 / Step 2 ...] 는 학습하지 않고, [Step k_w ~ / Step k_l ~]을 학습

(3) 아쉬운 점

- 어쨌든 틀린 지점을 찾는 공수가 필요함. Instruct 모델이야 이마저도 Reasoning 모델에 적용하기는 쉽지 않다.

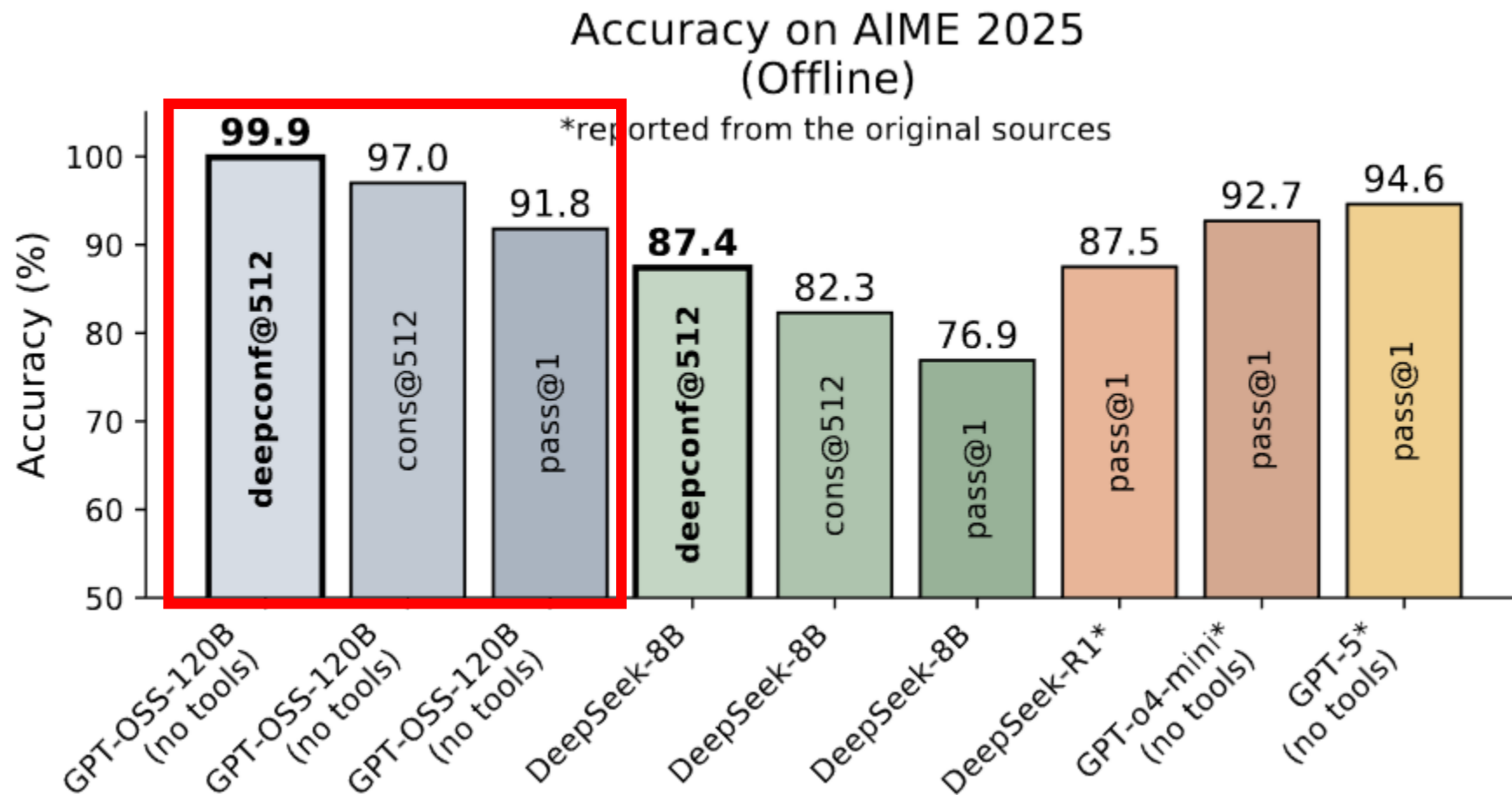
실험 설계: DTO + StepDPO

Motivation

- 현재 RLM의 optimization은 token efficiency에 상당히 치중되어 있음.
- 그러나 RLM의 강점은 넓은 space의 탐색임.
- 그런데 여기서 파생되는 단점이 높은 temperature에 따른 들쭉날쭉한 성능임.
- 최적의 성능을 내기 위해 32k full-trajectory를 64번 뽑아야 하는 모델은 사용 가치가 떨어짐.

⇒ 한 path 내 몇 token을 사용하는지도 중요하지만,
몇 번을 탐색해야 그 성능이 나오는지도 중요하다!

실험 설계: DTO + StepDPO



실험 설계: DTO + StepDPO

작은 모델일수록 성능이 오락가락한다?

1. Overthinking issue : 제때 사고를 끝내는 법을 배우지 못해 loop에 빠짐.
⇒ 쉬운 건 빨리 풀게
2. Error propagation : 넓은 추론 탐색 공간에서 방향을 잃고 이상한 길로 빠짐
⇒ 아는 문제도 틀리는 바보가 탄생
⇒ 탐색 space를 조금 좁혀줄 필요가 있음 (고점은 조금 낮아져도, 저점을 높이도록)
⇒ 특히 GRPO에서 breadth search 위주로 학습한 RL기반 모델들이 이런 문제가 더 심한 듯함

Method

Checkpoint 기반 데이터 생성 파이프라인

- Step-DPO와 유사하되, 외부 supervision이 전혀 없는 방법

- (1) 학습 데이터셋을 모델에게 풀게 함 (DeepScaleR 40K)

- (2) 평가 후 맞는 샘플과 틀린 샘플을 분리

- (3) 논리적인 분기점이 될 수 있는 trajectory의 checkpoint를 태깅함 ("WnWn" + "Wait"/"Alternatively"/"But")

- (4) Chosen/Reject Sampling 진행

- Right-to-Wrong : checkpoint 이후 생성 시 오답이 되는 경우를 수집

- Wrong-to-Right : checkpoint 이후 맞출 때까지 재생성

- (5) 수집한 샘플들로 DPO 기반 추가 학습 수행

=> pass@1과 pass@k 성능 좁히기 (가성비 있는 1-queue 추론 모델)

Method

왜 DPO인가?

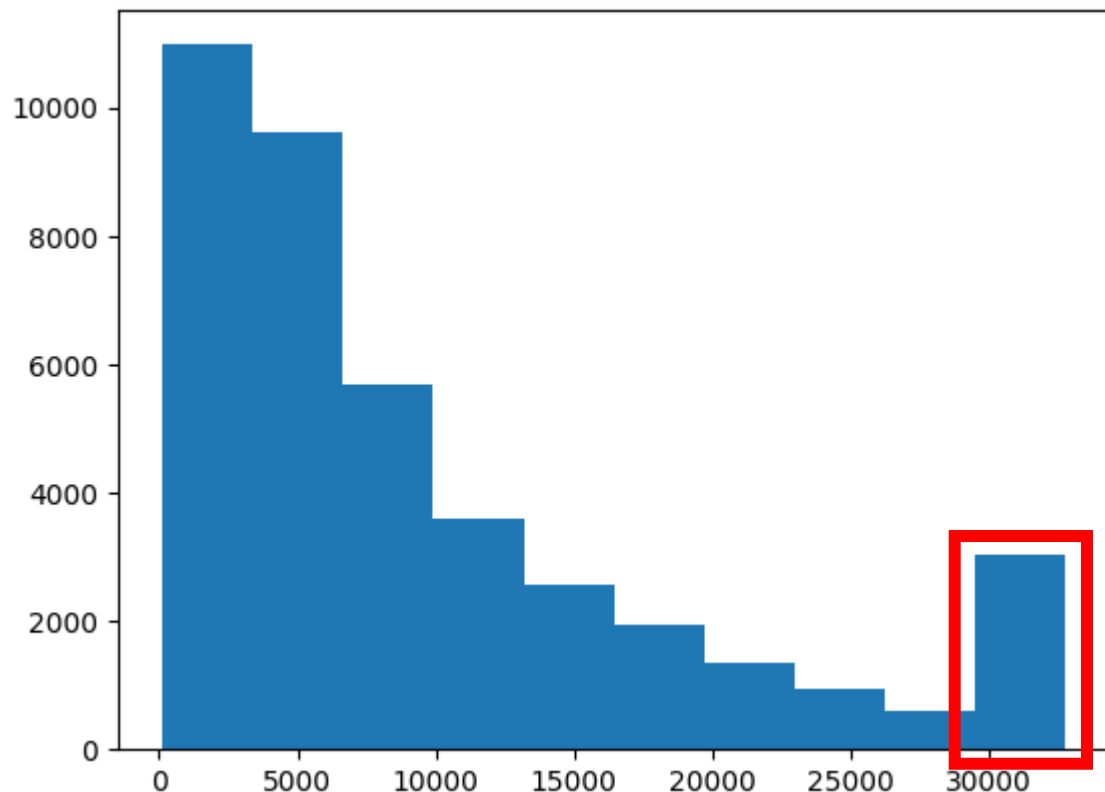
- (1) 지식 주입이 아닌 thinking 최적화 => In-distribution data
- (2) 비용.... long trajectory를 학습 데이터로 쓰기 때문에 full finetune은 어렵다
- (3) "일반화"를 향한 꿈 (패턴학습)
 - ⇒ DeepScaleR로 학습해도 AIME (수학), MBPP (코드), GPQA (일반)에서 성능이 오를 수 있을까?

1차 데이터 필터링

DeepSeek-R1-Distillation-Qwen-1.5B 모델 기준

- 40K data 중 17321 정답 + 22994 오답

오답 데이터의 길이 분포



오답 샘플의 Checkpoint (문단)의 분포

```
11650th index with the maximum number of paragraphs: 2298
13th index with the minimum number of paragraphs: 0
mean: 105.65073475565103
median: 68.0
std: 112.32616340430745
```

에러 샘플 (2298개 문단)

```
'Wait, triangle AFE: same.\n'
'\n'
'Wait, triangle AFD: same.\n'
'\n'
'Wait, triangle AFE: same.\n'
'\n'
```


Data Iteration Logic

어디서 부터 재생성할 것인가?

- 추론 최적화 시
 - 너무 앞쪽을 선택할 경우, 그냥 regular DPO 형태가 되어버림.
 - 너무 뒤쪽을 선택할 경우, 이미 오류가 너무 진행되어서 개선되지 않을 가능성이 높음.
- 초반부에서 많이 선정하되, 골고루 편성할 수 있는 조합
- 문단 수가 10개 이하인 경우 (5개 기준 예시) [1, 1, 1, 1, 1, 2, 2, 3, 3, 4, 5]
- 문단 수가 10개 이상인 경우, quantile로 10%씩 할당
- Batch로 재생성 => 평가 => pair 수집 => 반복...

Example Sample

Chantal and Jean start hiking from a trailhead toward a fire tower. Jean is wearing a heavy backpack and walks slower. Chantal starts walking at 4 miles per hour. ... What was Jean's average speed, in miles per hour, until they meet?

Okay, let's try to figure out this hiking problem. So, Chantal and Jean are starting from the trailhead and going towards a fire tower. Jean is wearing a heavy backpack, so he walks slower. Chantal starts at 4 mph, but halfway to the tower, the trail gets steep, and she slows down to 2 mph.

...

So, both of them meet at the halfway point, but Chantal is on her way back, while Jean is still going towards the tower.

Wait, so Chantal goes from trailhead to tower, then back down the steep part, and meets Jean at the halfway point. So, Chantal's path is trailhead -> halfway -> tower -> halfway, and Jean is just going trailhead -> tower.

... Thus, Jean's average speed was $\frac{12}{7}$ miles per hour.

Wait, no—the problem says "she meets Jean at the halfway point." Wait, the halfway point from where? The halfway point to the tower? Let me check the problem statement again.

... The calculation is consistent and verified through multiple approaches. $\frac{12}{13}$

Data Statistics

수집한 DeepSeek R1 1.5B 모델의 학습 데이터 구성

- 총 11,339개 샘플 확보
- 초기 DPO 파이프라인 구현 완

파악된 한계점

- Reject/Chosen의 길이가 너무 들쭉날쭉함.
- 32K full-length를 학습에 사용하는 비효율성 => 16k truncation 시 절반인 5000여 샘플 남음.
- Model-specific한 데이터 생성 방법, 정확히는 운에 의존한...
- 막막한 학습법

model	timestamp	AIME24	AIME25	AMC23	GPQADian	HumanEval	LiveCodeBench	MBPP
../3.training_phase/checkpoints/20260119_204850	20260120_134811	0.24333	0.19	0.59	0.22222	0	0.2531	0.106
deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B	20260120_134309	0.33333	0.21667	0.7	0.31313	0	0.26973	0.234

Thank you
