Uncovering the Impact of Chain-of-Thought Reasoning for Direct Preference Optimization: Lessons from Text-to-SQL

ACL 2025

Joint Work With
Haoyang Li*[1], Xiaokang Zhang[1], Ruotong Chen[1],
Haiyong Xu[2], Tian Tian[2], Qi Qi[1], and Jing Zhang†[1]

[1]Renmin University of China, [2]China Mobile Information Technology Center

Hanbing Liu, RUC

Motivation

INPUT

Database prompt:
table **movie** , columns = [movie.mid (int | primary key | comment : movie id | values : 101 , 102) , movie.title (text | values : Gone with the Wind , Star Wars) , movie.year (int | values : 1939 , 1977) , movie.director (text | values : Victor Fleming , George Lucas)]
table **reviewer** , columns = [reviewer.rid (int | primary key | comment : reviewer id | values : 201 , 202) , reviewer.name (text | values : Sarah Martinez , Daniel Lewis)]
table **rating** , columns = [rating.rid (int | comment : reviewer id | values : 201 , 202) , rating.mid (int | comment : movie id | values : 101 ,106) , rating.stars (int | comment : rating stars | values : 2 , 4) , rating.ratingdate (date | values : 2011-01-22 , 2011-01-27)]
foreign keys :
rating.rid = reviewer.rid
rating.mid = movie.mid
matched values :
reviewer.name (Sarah Martinez)

Question:
What are the names of all directors whose movies have been reviewed by Sarah Martinez?

OUTPUT

```
SELECT DISTINCT movie.director FROM rating JOIN movie ON rating.mid = movie.mid
JOIN reviewer ON rating.rid = reviewer.rid WHERE reviewer.name = 'Sarah Martinez'
```

Figure 4: A text-to-SQL sample in Spider's training set, consisting of a triplet of <database prompt, question, SQL query>. The database prompt is crafted by our proposed method.

Credit: Li et al. (SIGMOD 2024)

A typical fine-tune based Text-to-SQL method

1. Construct question-aware database prompt
2. Fine-tune LM to directly return SQL

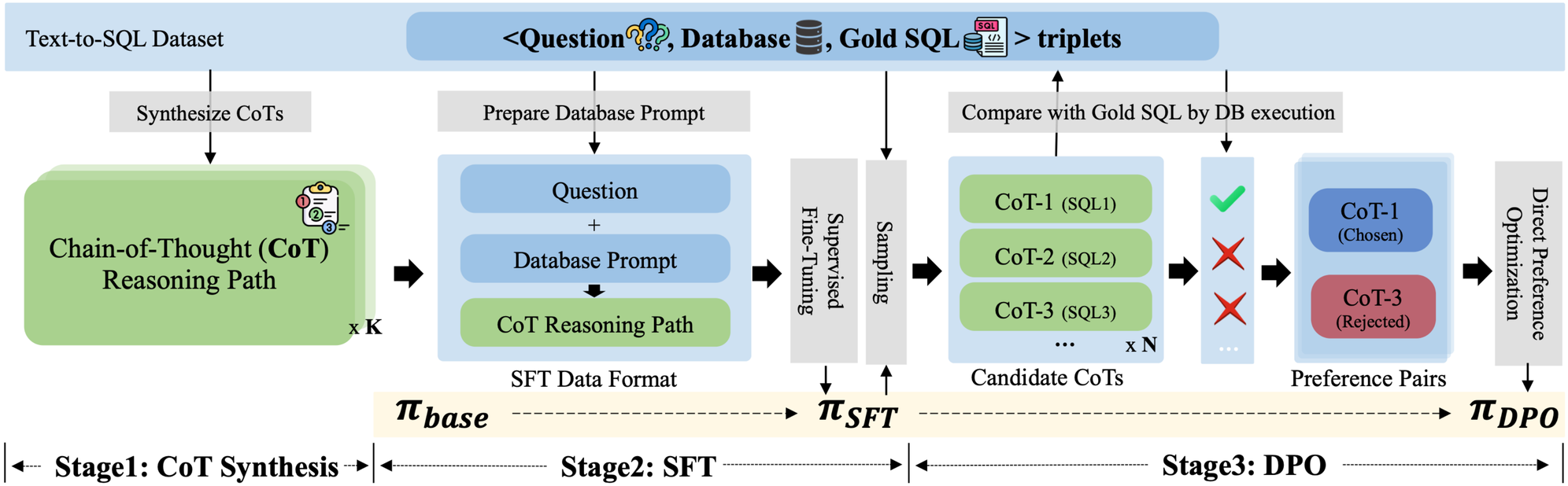
Shortcomings

1. No utilization of **step-by-step** reasoning
2. No **preference learning** phase after SFT

However, direct implementation just doesn't work!



Pipeline


Text-to-SQL Dataset
<Question, Database, Gold SQL > triplets
Synthesize CoTs
Prepare Database Prompt
Compare with Gold SQL by DB execution
Chain-of-Thought (CoT) Reasoning Path
x K
Question
+
Database Prompt
CoT Reasoning Path
SFT Data Format
Supervised Fine-Tuning
Sampling
CoT-1 (SQL1)
CoT-2 (SQL2)
CoT-3 (SQL3)
... x N
Candidate CoTs
CoT-1 (Chosen)
CoT-3 (Rejected)
Preference Pairs
Direct Preference Optimization
π_{base}
π_{SFT}
π_{DPO}
Stage1: CoT Synthesis
Stage2: SFT
Stage3: DPO

Main Results

	Model	Bird Dev						ΔEX
		Greedy		Pass@1		Maj@16		
		SFT	DPO	SFT	DPO	SFT	DPO	
General Models								
Vanilla	Deepseek-llm-7b-chat	51.8	51.2 (-0.6)	47.9	49.1 (+1.3)	54.5	54.3 (-0.3)	-
	Llama-3.1-instruct-7b	59.0	59.8 (+0.8)	56.1	57.2 (+1.1)	61.4	60.8 (-0.6)	-
	Qwen2.5-7B-Instruct	58.8	59.0 (+0.2)	55.1	55.7 (+0.6)	61.4	60.6 (-0.8)	-
	Qwen2.5-14B-Instruct	64.3	63.5 (-0.8)	62.3	62.6 (+0.3)	64.6	65.1 (+0.5)	-
Syn CoT	Deepseek-llm-7b-chat	54.3	55.9 (+1.6)	51.9	54.8 (+2.9)	59.1	61.0 (+1.9)	54.5 → 61.0 (**+6.5**)
	Llama-3.1-instruct-7b	56.8	61.2 (+4.4)	57.5	59.0 (+1.5)	60.2	61.9 (+1.7)	61.4 → 61.9 (**+0.5**)
	Qwen2.5-7B-Instruct	57.4	61.9 (+4.5)	54.8	59.2 (+4.4)	63.0	64.9 (+1.9)	61.4 → 64.9 (**+3.5**)
	Qwen2.5-14B-Instruct	63.2	65.3 (+2.1)	61.8	64.7 (+2.9)	65.4	67.1 (+1.7)	64.6 → 67.1 (**+2.5**)
Coder Models								
Vanilla	Deepseek-coder-6.7b-instruct	60.6	60.9 (0.3)	56.9	58.8 (+1.9)	59.8	61.0 (+1.2)	-
	CodeLlama-7b-Instruct-hf	57.0	55.7 (-1.3)	54.3	55.5 (+1.2)	59.1	58.5 (-0.6)	-
	CodeLlama-13b-Instruct-hf	60.0	60.2 (+0.2)	56.7	57.9 (+1.2)	61.9	62.0 (+0.1)	-
	Qwen2.5-Coder-7B-Instruct	61.6	61.3 (-0.3)	59.4	60.6 (+1.2)	61.3	62.7 (+1.4)	-
Syn CoT	Deepseek-coder-6.7b-instruct	61.5	63.8 (+2.3)	59.9	62.3 (+4.5)	64.3	65.4 (+1.1)	59.8 → 65.4 (**+5.6**)
	CodeLlama-7b-Instruct-hf	58.2	61.3 (+3.1)	56.9	60.4 (+3.5)	60.2	61.9 (+1.7)	59.1 → 61.9 (**+2.8**)
	CodeLlama-13b-Instruct-hf	62.0	63.9 (+1.9)	59.8	62.5 (+2.7)	63.6	65.8 (+2.2)	61.9 → 65.8 (**+3.9**)
	Qwen2.5-Coder-7B-Instruct	60.8	63.4 (+2.6)	59.1	62.8 (+3.7)	62.5	64.1 (+1.6)	61.3 → 64.1 (**+2.8**)
SQL-Specialized Models								
Vanilla	CodeS-7b	56.8	56.6 (-0.2)	53.7	54.6 (+0.9)	58.1	58.0 (-0.1)	-
	CodeS-15b	58.3	58.2 (-0.1)	55.6	56.2 (+0.6)	60.2	59.1 (-1.1)	-
Syn CoT	CodeS-7b	56.7	57.5 (+0.8)	54.2	55.3 (+1.1)	60.2	61.7 (+1.5)	58.1 → 61.7 (**+2.6**)
	CodeS-15b	58.6	61.1 (+2.5)	56.6	60.5 (+3.9)	62.4	63.2 (+0.8)	60.2 → 63.2 (**+3.0**)

Table 1: Model performance on the Bird development set. **Vanilla**: SFT and DPO on the original Bird training set; **Syn CoT**: SFT and DPO on the CoT-enhanced training set; **ΔEX**: The performance difference in EX between "Syn CoT + DPO" and "Vanilla + SFT" when using the same base model.

What Does DPO Benefit From CoT?

1 More accurate reward model

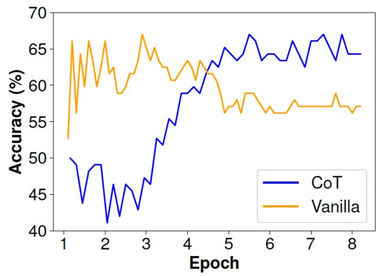
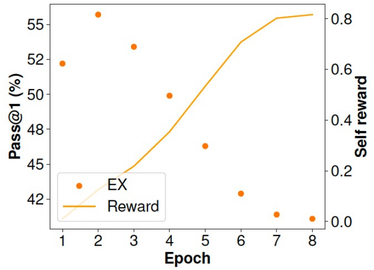
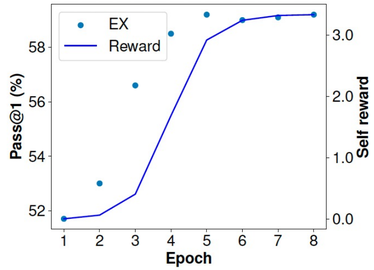


Figure 2: Comparison of model's discriminative ability during DPO (measured by classification accuracy on curated evaluation set).



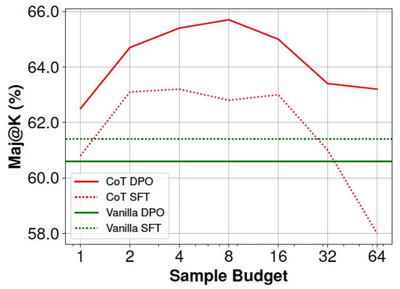
(a) Vanilla



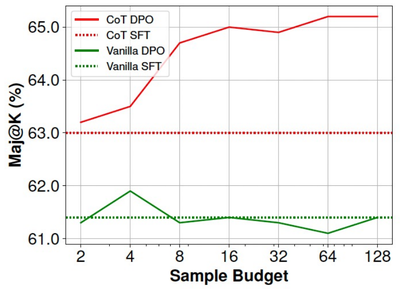
(b) Syn CoT

Figure 3: Comparison of model's self-assessed performance (average implicit reward policy model given to its own roll-outs) and real performance (EX) on Bird development set (Pass@1) during DPO training.

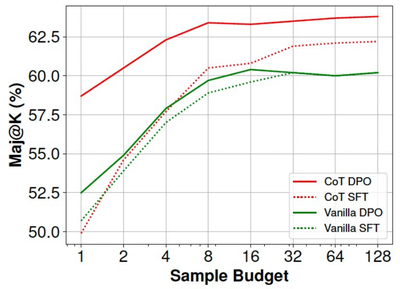
2 More stable training process



(a) Chain-of-Thought Synthesis



(b) Preference Data Collection



(c) Inference-time Computation

Figure 4: Model performance with different sample budget K in each stage (Maj@K).

3 More reliable scaling behavior

Error Analysis

Type	Vanilla DPO Fix (%)	Syn CoT DPO Fix (%)	Δ(%)↓
[E2] Column Sequence	0 (0/5)	**42.9 (3/7)**	**+42.9**
[A1] EK	0.0 (0/3)	**37.5 (3/8)**	**+37.5**
[B6] NULL/DISTINCT	9.7 (3/31)	**40.0 (12/30)**	**+30.3**
[C1] String/Number	4.5 (1/22)	21.1 (4/19)	**+16.6**
[B2] JOIN	15.6 (12/77)	**32.1 (18/56)**	**+16.5**
[D2] Aggregation	6.7 (5/75)	18.2 (12/66)	**+11.5**
[E1] Redundant/Incomplete	11.8 (4/34)	19.2 (5/26)	**+7.4**
[C2] Date	23.1 (6/26)	**30.4 (7/23)**	**+7.3**
[D3] Complex Operation	5.6 (1/18)	12.5 (3/24)	**+6.9**
[B5] Condition	16.7 (10/60)	23.2 (16/69)	**+6.5**
[B3] Column	10.3 (7/68)	16.1 (10/62)	**+5.8**
[D1] Mathematical Formula	13.3 (6/45)	18.2 (8/44)	**+4.9**
[B4] Hallucination	23.7 (14/59)	**27.2 (28/102)**	**+3.5**
[E3] ORDER BY/LIMIT	9.1 (1/11)	12.5 (1/8)	**+3.4**
[B1] Table	13.0 (12/92)	15.9 (11/69)	**+2.9**
[F1] Syntax	14.3 (2/14)	14.3 (2/14)	**-1.0**
[E4] Format	66.7 (2/3)	**33.3 (2/6)**	**-33.4**

Table 26: Comparison of Vanilla and Syn CoT DPO correction capability across error types on Bird development set (greedy), results are arranged in descending order of fix rate difference. Base model is Qwen2.5-7B-Instruct. For Syn CoT, Error types with fix rates over 25% are bolded, and error types with the five lowest fix rates are underlined.

DPO excels at correcting errors caused by **ignoring details**

Q: What are author names and IDs?
A: `SELECT id, name FROM author`



Error Analysis

Type	Vanilla DPO Fix (%)	Syn CoT DPO Fix (%)	Δ(%)↓
[E2] Column Sequence	0 (0/5)	**42.9 (3/7)**	**+42.9**
[A1] EK	0.0 (0/3)	**37.5 (3/8)**	**+37.5**
[B6] NULL/DISTINCT	9.7 (3/31)	**40.0 (12/30)**	**+30.3**
[C1] String/Number	4.5 (1/22)	21.1 (4/19)	**+16.6**
[B2] JOIN	15.6 (12/77)	**32.1 (18/56)**	**+16.5**
[D2] Aggregation	6.7 (5/75)	18.2 (12/66)	**+11.5**
[E1] Redundant/Incomplete	11.8 (4/34)	19.2 (5/26)	**+7.4**
[C2] Date	23.1 (6/26)	**30.4 (7/23)**	**+7.3**
[D3] Complex Operation	5.6 (1/18)	12.5 (3/24)	**+6.9**
[B5] Condition	16.7 (10/60)	23.2 (16/69)	**+6.5**
[B3] Column	10.3 (7/68)	16.1 (10/62)	**+5.8**
[D1] Mathematical Formula	13.3 (6/45)	18.2 (8/44)	**+4.9**
[B4] Hallucination	23.7 (14/59)	**27.2 (28/102)**	**+3.5**
[E3] ORDER BY/LIMIT	9.1 (1/11)	12.5 (1/8)	**+3.4**
[B1] Table	13.0 (12/92)	15.9 (11/69)	**+2.9**
[F1] Syntax	14.3 (2/14)	14.3 (2/14)	**-1.0**
[E4] Format	66.7 (2/3)	**33.3 (2/6)**	**-33.4**

Table 26: Comparison of Vanilla and Syn CoT DPO correction capability across error types on Bird development set (greedy), results are arranged in descending order of fix rate difference. Base model is Qwen2.5-7B-Instruct. For Syn CoT, Error types with fix rates over 25% are bolded, and error types with the five lowest fix rates are underlined.

CoT provides **logical guidance** to enhance DPO's error correction

Q: What are venue names and expenses?
A: We need to join table venue and table expense **through table venue_expense**.

```
SELECT venue.names, expense.amount FROM venue JOIN venue_expense ON venue_id JOIN expense ON expense.id
```



Error Analysis

Type	Vanilla DPO Fix (%)	Syn CoT DPO Fix (%)	Δ(%)↓
[E2] Column Sequence	0 (0/5)	**42.9 (3/7)**	**+42.9**
[A1] EK	0.0 (0/3)	**37.5 (3/8)**	**+37.5**
[B6] NULL/DISTINCT	9.7 (3/31)	**40.0 (12/30)**	**+30.3**
[C1] String/Number	4.5 (1/22)	21.1 (4/19)	**+16.6**
[B2] JOIN	15.6 (12/77)	**32.1 (18/56)**	**+16.5**
[D2] Aggregation	6.7 (5/75)	18.2 (12/66)	**+11.5**
[E1] Redundant/Incomplete	11.8 (4/34)	19.2 (5/26)	**+7.4**
[C2] Date	23.1 (6/26)	**30.4 (7/23)**	**+7.3**
[D3] Complex Operation	5.6 (1/18)	12.5 (3/24)	**+6.9**
[B5] Condition	16.7 (10/60)	23.2 (16/69)	**+6.5**
[B3] Column	10.3 (7/68)	16.1 (10/62)	**+5.8**
[D1] Mathematical Formula	13.3 (6/45)	18.2 (8/44)	**+4.9**
[B4] Hallucination	23.7 (14/59)	**27.2 (28/102)**	**+3.5**
[E3] ORDER BY/LIMIT	9.1 (1/11)	12.5 (1/8)	**+3.4**
[B1] Table	13.0 (12/92)	15.9 (11/69)	**+2.9**
[F1] Syntax	14.3 (2/14)	14.3 (2/14)	**-1.0**
[E4] Format	66.7 (2/3)	**33.3 (2/6)**	**-33.4**

Schema Linking

Table 26: Comparison of Vanilla and Syn CoT DPO correction capability across error types on Bird development set (greedy), results are arranged in descending order of fix rate difference. Base model is Qwen2.5-7B-Instruct. For Syn CoT, Error types with fix rates over 25% are bolded, and error types with the five lowest fix rates are underlined.

Existing Text-to-SQL methods can **complement** DPO's weaknesses

Bird		Spider	
Method	**EX**	**Method**	**EX**
DTS-SQL	47.7	C3-SQL	80.5
+ CoT	52.0 (**+4.3**)	+ CoT	80.9 (**+0.4**)
+ CoT + DPO	55.9 (**+8.2**)	+ CoT + DPO	81.9 (**+1.4**)

Table 6: Performance improvements of two Text-to-SQL frameworks when incorporating our trained models.

Takeaways

- **When equipped with CoT, DPO finally work in Text-to-SQL!**
  (Which also applies to RFT and other XPOs)
- To DPO w/o CoT is prone to reward hacking, while **CoT can stabilize DPO training**, making it robust to hyper-parameter choice.
- In Text-to-SQL, **DPO works best for errors that caused by ignoring details**, owing to the explicit logical guidance provided by CoT.