Grounding Natural Language to SQL Translation with Data-Based Self-Explanations

Yuankai Fan, Tonghui Ren, Can Huang, Zhenying He, X.Sean Wang
Fudan University, TeleAI

Background

Natural Language Interfaces for Databases empower non-technical users to interact with data using natural language (NL). Advanced approaches, utilizing either sequence-to-sequence or more recent sophisticated large-scale language models (LLM), typically implement NL to SQL (NL2SQL) translation in an **end-to-end fashion**. However, like humans, these end-to-end translation models may not always generate the best SQL output on their first try.

Unsatisfied Results: The plateauing accuracy (below 80%) observed when the beam size or the number of chat completions is set to 1, they may **fail** to generate best-quality translations in their **initial attempts**.

However, expanding the search space through wider beams or more chat completions steadily improves accuracy, without necessitating modifications to the underlying model architectures, which could be the potential.

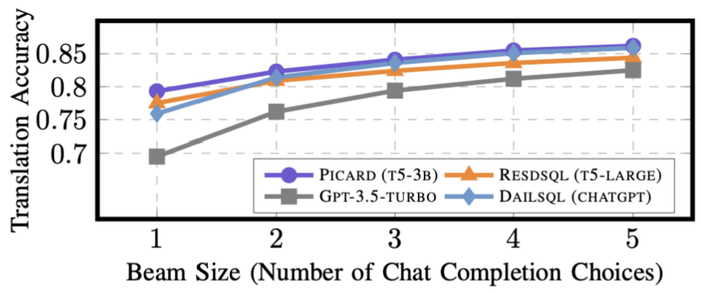
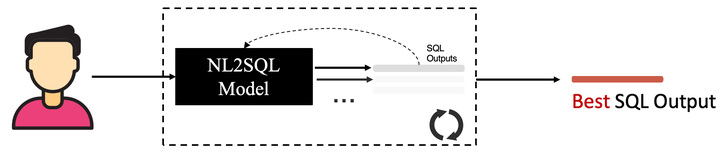


Fig. 1: Translation accuracy on SPIDER validation set with varied beam sizes (or chat completion choices). Accuracy for beam sizes (or chat completions) > 1 is evaluated by matching any beam result.

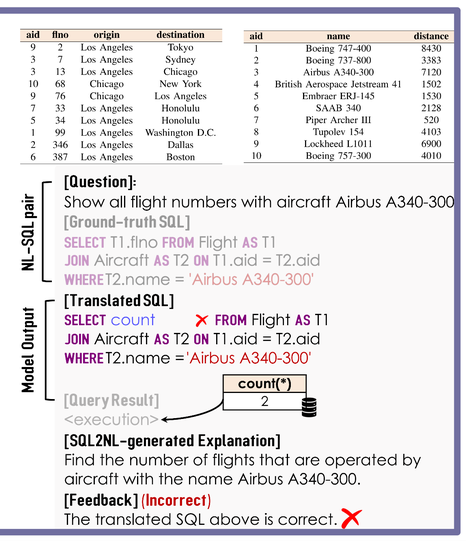
Motivation: Can we build a **self-provided feedback loop** along with NL2SQL, to improve end-to-end performance?



Question: Is **SQL2NL Back-Translation** Sufficient to establish an NL-to-SQL-to-NL translation lifecycle?

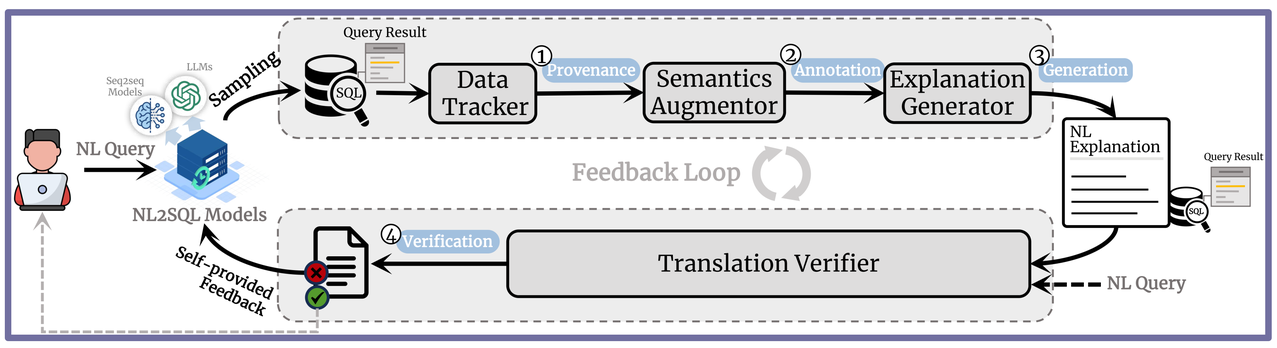
Answer: ✗ NOT reliable!!!

Fase positive feedback for NL questions is observed, which leads to incorrect translation result ultimately.



CycleSQL

OVERVIEW

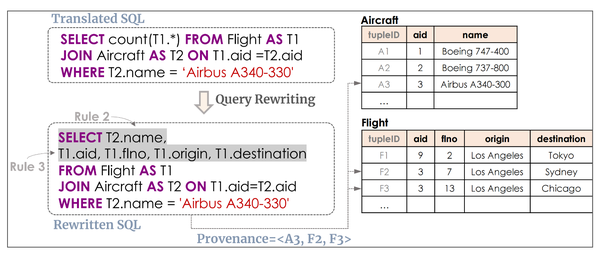


- ✓ Data Tracker: Track provenance information of the to-explained query result to retrieve **data-level information**
- ✓ Semantics Augmentor: Enhance provenance by associating it with **operation-level semantics**
- ✓ Explanation Generator: Interpret provenance to **explanation**
- ✓ Translation Verifier: Verify **correctness** of NL2SQL translation based on the generated NL explanation

METHODOLOGIES

Data Provenance

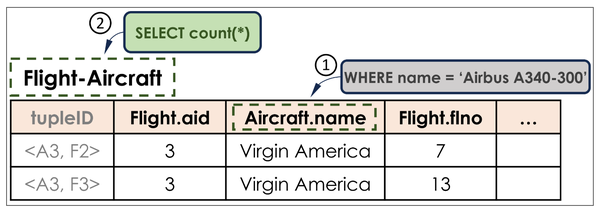
Capture provenance information using **query rewriting**, with heuristic rules:
① Result Transformation Rule: Translate query result into **WHERE clause** to specify as condition
② Projection Enhancement Rule: Extract **all columns** used in the original query, as additional projection columns
③ Aggregation Deconstructive Rule: Eliminate aggregations to unravel lineage



Semantics Enrichment

Main idea of integrating the operation-level semantics of the SQL queries is to better reflect **user query intent**.

As different parts of the query may "**contribute**" to different parts of the provenance information, we treat the SQL query as a text string and divides the string into **chunks** that correspond to each clause in the SQL query.



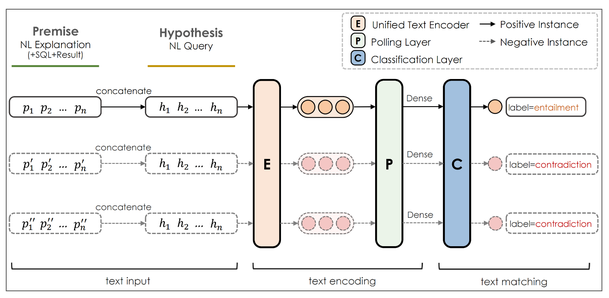
Explanation Text Genearation

To interpret the enriched provenance to a textual expression in free-text forms, we employs a **rule-based method** to synthesize the NL explanation using the enriched provenance information:

The query returns a result with one column of aggregation type (count) and one row. For flights with aircraft, named Airbus A340-300, there are 2 flights in total.

NL2SQL Translation Validation

Drawing inspiration from recent achievements in the NLP field, we thus formulate the translation validation problem as a **textual entailment task** (NLI) and leverage deep learning techs to build the **NLI model** for this purpose.



Evaluation

Overall Results

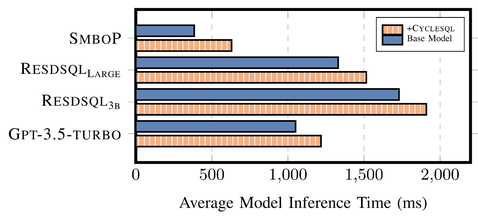
Models		Spider Validation EM	Spider Validation EX	Spider Validation TS	Spider Test EM	Spider Test EX	Spider Variants Realistic EM	Realistic EX	Realistic TS	Syn EM	Syn EX	Syn TS	Dk EM	Dk EX	ScienceBenchmark OncoMx EX	Cordis EX	Sdss EX
Seq2seq-based Models																	
SmBoP	Base	**74.5**	77.8	72.5	69.5	71.1	**60.2**	61.4	56.1	**60.2**	64.9	58.3	**54.0**	59.1	16.2	**16.0**	7.0
	+CycleSQL	74.1	**78.8**	**73.1**	-	-	59.8	**62.2**	**56.7**	59.1	**65.5**	**58.6**	53.3	**59.8**	**17.2**	**16.0**	**8.0**
PICARD$_{3B}$	Base	75.9	79.3	**72.8**	72.4	75.1	**68.9**	71.7	**64.6**	65.6	69.8	63.2	49.7	63.2	32.3	**26.0**	**8.0**
	+CycleSQL	**76.1**	**79.9**	**72.8**	-	-	68.7	**71.9**	**64.6**	**65.9**	**70.8**	**63.6**	**50.3**	**63.6**	**33.3**	24.0	**8.0**
RESDSQL$_{LARGE}$	Base	74.0	77.5	71.6	66.8	73.5	67.3	69.5	61.6	58.8	65.3	59.6	50.1	58.1	33.0	29.0	4.0
	+CycleSQL	**75.1**	**80.5**	**74.0**	**68.9**	**77.1**	**69.1**	**73.6**	**64.2**	**61.6**	**69.6**	**63.2**	**50.5**	**61.5**	**35.0**	**32.0**	**8.0**
RESDSQL$_{3B}$	Base	76.0	79.4	73.5	70.2	78.4	68.9	72.2	64.8	62.9	69.5	63.2	52.0	59.3	34.1	28.0	6.0
	+CycleSQL	**76.8**	**82.0**	**76.0**	**72.5**	**81.6**	**71.3**	**76.5**	**67.5**	**65.5**	**73.3**	**66.6**	**52.5**	**61.3**	**37.4**	**31.0**	**8.0**
LLM-based Models																	
GPT-3.5-TURBO (5-SHOTS)	Base	43.8	72.8	61.7	44.7	69.4	39.4	65.4	52.8	34.0	61.0	49.8	39.1	60.6	34.3	30.0	10.0
	+CycleSQL	**49.2**	**77.8**	**66.2**	**50.6**	**75.1**	**46.5**	**71.1**	**56.7**	**40.0**	**66.7**	**55.4**	**43.7**	**65.6**	**43.4**	**34.0**	**12.0**
GPT-4 (5-SHOTS)	Base	51.9	77.9	71.2	-	-	46.7	68.6	55.3	45.2	75.0	64.4	53.7	**68.5**	51.5	40.0	14.0
	+CycleSQL	**58.7**	**79.8**	**73.1**	-	-	**49.0**	**70.6**	**56.9**	**46.2**	**76.0**	**66.3**	**57.4**	**68.5**	**56.6**	**43.0**	**15.0**
CHESS	Base	23.4	41.1	37.5	-	-	21.2	39.7	36.6	17.4	37.0	32.5	19.5	35.6	64.6	46.0	**40.0**
	+CycleSQL	**25.6**	**42.3**	**38.2**	-	-	**23.1**	**41.1**	**37.9**	**19.3**	**38.1**	**33.0**	**20.2**	**36.1**	**65.7**	**52.0**	**40.0**
DAILSQL$_{3.5}$	Base	65.1	81.3	74.1													
	+CycleSQL	**67.2**	**81.8**	**74.3**													

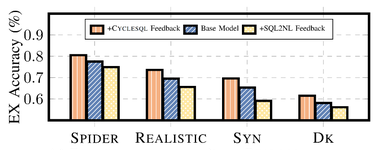
- ✓ CycleSQL **consistently** improves over all base models, particularly evident in the EX and TS metrics.
- ✓ Remarkable outcome: CycleSQL+RESDSQL-3B on EX metric **surpasses** the best-reported result among leading Seq2seq models on Spider

Analysis

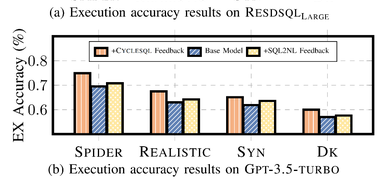
Model		Easy	Medium	Hard	Extra Hard
SmBoP	Base	90.7	82.7	70.7	52.4
	+CycleSQL	90.7	**84.1**$_{(\uparrow 1.4)}$	69.5$_{(\downarrow 1.2)}$	53.0$_{(\uparrow 0.6)}$
PICARD$_{3B}$	Base	95.6	85.4	67.8	50.6
	+CycleSQL	95.6	86.1$_{(\uparrow 0.7)}$	69.5$_{(\uparrow 1.7)}$	50.6
RESDSQL$_{LARGE}$	Base	92.3	83.4	66.1	51.2
	+CycleSQL	93.5$_{(\uparrow 1.2)}$	86.1$_{(\uparrow 0.7)}$	**73.0**$_{(\uparrow 6.9)}$	53.6$_{(\uparrow 2.4)}$
RESDSQL$_{3B}$	Base	94.0	85.7	65.5	55.4
	+CycleSQL	94.0	**89.0**$_{(\uparrow 3.3)}$	**74.7**$_{(\uparrow 9.2)}$	53.0$_{(\downarrow 0.4)}$
GPT-3.5-TURBO (5-SHOTS)	Base	84.3	78.5	65.5	48.2
	+CycleSQL	86.3$_{(\uparrow 2.0)}$	**83.0**$_{(\uparrow 4.5)}$	**73.0**$_{(\uparrow 7.5)}$	**56.0**$_{(\uparrow 7.8)}$
GPT-4 (5-SHOTS)	Base	90.3	84.3	63.8	56.6
	+CycleSQL	90.7$_{(\uparrow 0.4)}$	85.4$_{(\uparrow 1.1)}$	**66.7**$_{(\uparrow 2.9)}$	**59.6**$_{(\uparrow 3.0)}$
CHESS	Base	70.2	25.3	39.7	19.3
	+CycleSQL	**71.6**$_{(\uparrow 1.4)}$	25.6$_{(\uparrow 0.3)}$	**41.1**$_{(\uparrow 1.4)}$	**21.6**$_{(\uparrow 2.3)}$
DAILSQL$_{3.5}$	Base	91.1	86.5	77.0	57.2
	+CycleSQL	91.1	**86.8**$_{(\uparrow 0.3)}$	**77.6**$_{(\uparrow 0.6)}$	**59.0**$_{(\uparrow 1.8)}$

Average Iterations	SmBoP (+CycleSQL)	PICARD (+CycleSQL)	RESDSQL$_{LARGE}$ (+CycleSQL)	RESDSQL$_{3B}$ (+CycleSQL)	GPT-3.5-TURBO (+CycleSQL)
	2.44	3.78	1.90	1.88	1.87





(a) Execution accuracy results on RESDSQL$_{LARGE}$



(b) Execution accuracy results on GPT-3.5-TURBO

- ✓ 📈 Significant Improvement over **Extra-Hard-Queries** on LLM Models
- ✓ ⚖️ Good **Balance** between Translation and Inference Latency
- ✓ 👍 CycleSQL Feedback is **Better** than SQL2NL Feedback