A Survey of NL2SQL with Large Language Models: Where are we, and where are we going?

Speaker: Jiahao He

School of Information
Renmin University of China

March 12, 2025

Motivating Example

NL query: *Find the names of all customers who checked out books on exactly 3 different genres on Labor Day in 2023.*

Database:

Customer		
CustomerId	Name	...

Book				
BookId	Title	LiteralGenres	SubjectGenres	...

BookOrder			
CustomerID	BookID	OrderDate	...

Additional Information: Note that Labor Day stands for May 1st.

Motivating Example

1. Find key components in NL: entities, attributes, temporal information, specific conditions. . .
2. Find relevant tables, columns and, cell values in DB: Customer, Book, BookOrder, Name, LiteralGenres, SubjectGenres. . .
3. Writing SQL base on NL and DB understanding:

```
SELECT FROM Customer
NATURAL JOIN BookOrder
NATURAL JOIN Book
WHERE OrderDate='2023-05-01'
GROUP BY CustomerId
HAVING COUNT(DISTINCT
LiteralGenre)=3
```

```
SELECT FROM Customer
WHERE CustomerId = (SELECT
CustomerId
FROM BookOrder
NATURAL JOIN Book
WHERE OrderDate= '2023-05-01'
GROUP BY CustomerId
HAVING COUNT(DISTINCT
SubjectGenre)=3)
```

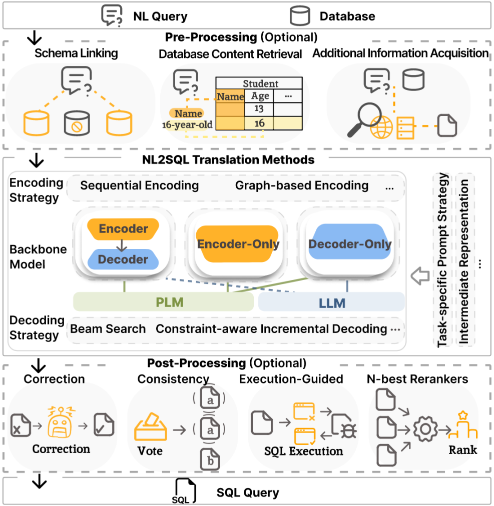
Chanllenges and Solutions

C1: **Uncertain natural language query**: lexical and syntactic ambiguity. . .

C2: **Complex database and dirty content**: complex relations, ambiguity attributes. . .

C3: **NL2SQL translation**: multiple possible queries. . .

C4: **technical challenges in developing NL2SQL solutions**: efficiency, reliability. . .



Preprocessing Strategies for NL2SQL

A. Schema Linking

- String Matching-based: Identifies matches when candidates are identical or when one is a substring of the other. -struggle with synonyms and vocabulary variations.
- Neural Network-based: Use neural networks to learn the relationship between the NL and the DB schema. -struggle to generalize across databases.
- In-Context Learning: Utilizes LLMs' ability to understand complex language patterns and relationships within data schemas. -computationally expensive.

Example

Given the database schema and question, perform the following actions:
1 - Rank all the table base on the possibiliy of being used in the SQL according to the question from the most relevant to the least relevant. Table or its columns that matches more with the question's words is highly relevant and must be placed ahead.

. . .

Preprocessing Strategies for NL2SQL

B. Database Content Retrieval: extracting cell values for specific SQL clauses

- String Matching-based: Simliar with schema linking.-synonyms and computationally expensive
- Neural Network-based: Capture complex data and semantic features through layers of nonlinear transformations, helping to resolve synonym issues -ambiguous NL.
- Index-based: Enabling faster access to relevant cell values. -frequent changes.

Example

NL: "Show me the names of all employees in the Sales department."

SQL: SELECT Employees.Name
FROM Employees
JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
WHERE Departments.DepartmentName = 'Sales';

Preprocessing Strategies for NL2SQL

C. Additional Information Acquisition: fetch additional information for the translation.

- Sample-based: Use few-shot examples from extensive domain knowledge
- Retrieval-based: Retrieval similar information.

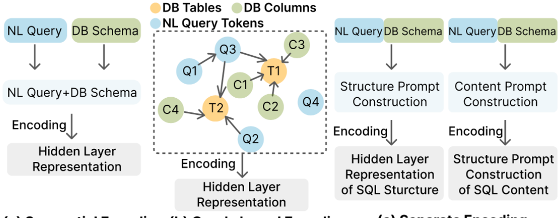
Example

NL: *Find the names of all customers who checked out books on exactly 3 different genres on Labor Day in 2023.*

Additional Information: Note that Labor Day stands for May 1st in China and September 4th in the US.

NL2SQL Translation Methods

A. Encoding Strategy:transforming NL and database schema into a structured format suitable for language model processing.

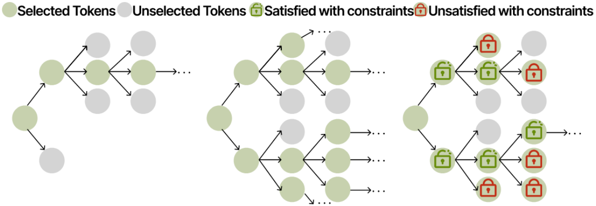

DB Tables
DB Columns
NL Query Tokens
NL Query
DB Schema
NL Query+DB Schema
Encoding
Hidden Layer Representation
Q1
Q2
Q3
Q4
T1
T2
C1
C2
C3
C4
Encoding
Hidden Layer Representation
NL Query
DB Schema
Structure Prompt Construction
Encoding
Hidden Layer Representation of SQL Sturcture
NL Query
DB Schema
Content Prompt Construction
Encoding
Structure Prompt Construction of SQL Content

(a) Sequential Encoding (b) Graph-based Encoding (c) Separate Encoding (SC-Prompt)

1) Sequential Encoding Strategy: Both the NL and the database schema are treated as token sequences.
2) Graph-based Encoding Strategy: Represents both NL and database schema as interconnected graphs.
3) Separate Encoding Strategy: Different parts of the NL are encoded separately and then combined to form the final SQL.

NL2SQL Translation Methods

B. Decoding Strategy: Transforming encoder-generated representations into SQL.



(a) Greedy Search-based Decoding (b) Beam Search-based Decoding (c) Constraint-aware Incremental Decoding

1) Greedy Search-based Decoding: Select the token with the highest probability at each decoding step.
2) Beam Search-based Decoding Strategy: Retains multiple candidate sequencesand expands the top-k tokens with the highest probabilities until a complete output sequence is generated.
3) Constraint-aware Incremental Decoding Strategy: As the model predicts the next token, it also checks the syntactic correctness of the partial query .

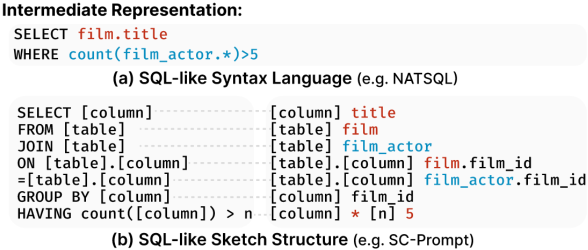
NL2SQL Translation Methods

C. Task-specific Prompt Strategy

1) Chain-of-Thought (CoT) Prompting: Use COT process to do entity and context retrieval, schema selection, SQL generation, and revision.
2) Decomposition Strategy: divides the NL2SQL task into sequential subtasks, allowing each sub-module to concentrate on a specific generation step.

NL2SQL Translation Methods

D. Intermediate Representation for NL2SQL Translation



Intermediate Representation:

```
SELECT film.title
WHERE count(film_actor.*)>5
```

(a) SQL-like Syntax Language (e.g. NATSQL)

```
SELECT [column]               [column] title
FROM [table]                  [table] film
JOIN [table]                  [table] film_actor
ON [table].[column]           [table].[column] film.film_id
=[table].[column]             [table].[column] film_actor.film_id
GROUP BY [column]             [column] film_id
HAVING count([column]) > n    [column] * [n] 5
```

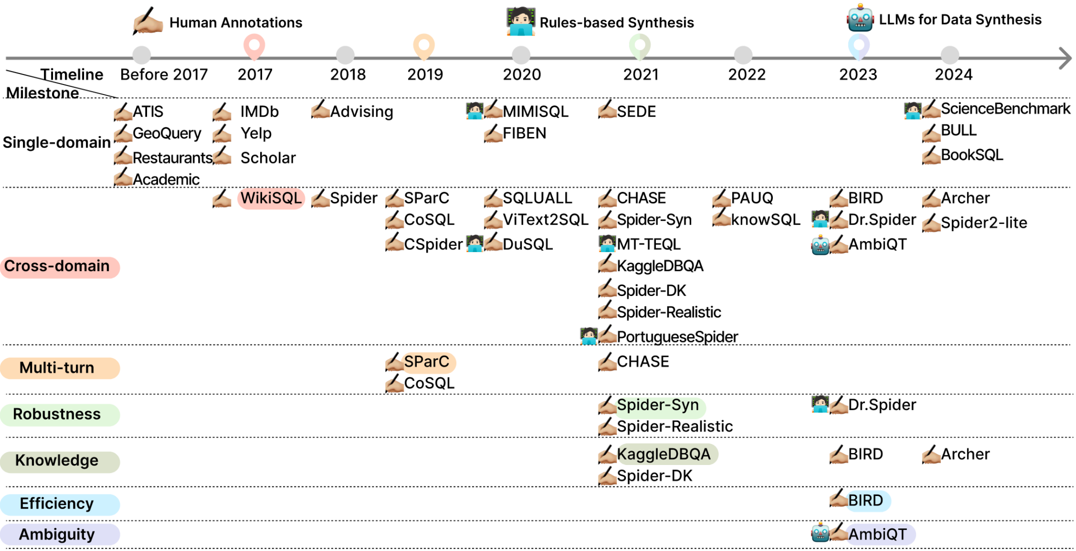
(b) SQL-like Sketch Structure (e.g. SC-Prompt)

1) SQL-like syntax language: Use simplified SQL-like structure.
2) SQL-like sketch structure: Use SQL-like sketches that mirror SQL structure for parsing, enabling diverse NL queries to be mapped into a specific sketch space.

Postprocessing Strategies for NL2SQL

A. SQL Correction Strategies: Mainly focus on correcting syntax errors in the generated SQL queries.
B. Output Consistency: Ensure that the generated SQL query is consistent with the NL query.
C. Execution Guided Strategies: Refine SQL based on execution results, ensuring the query retrieves data correctly.
D. N-best Reranker Strategies: Reorder the top-N model outputs, often leveraging a larger model (BERT-based reranker) or additional knowledge sources.

NL2SQL Benchmarks



✍ Human Annotations

🧑‍💻 Rules-based Synthesis

🤖 LLMs for Data Synthesis

Timeline / Milestone	Before 2017	2017	2018	2019	2020	2021	2022	2023	2024
Single-domain	✍ATIS ✍GeoQuery ✍Restaurants ✍Academic	✍IMDb ✍Yelp ✍Scholar	✍Advising		🧑‍💻✍MIMISQL ✍FIBEN	✍SEDE			🧑‍💻✍ScienceBenchmark ✍BULL ✍BookSQL
Cross-domain		✍WikiSQL	✍Spider	✍SParC ✍CoSQL ✍CSpider	✍SQLUALL ✍ViText2SQL 🧑‍💻✍DuSQL	✍CHASE ✍Spider-Syn 🧑‍💻MT-TEQL ✍KaggleDBQA ✍Spider-DK ✍Spider-Realistic 🧑‍💻✍PortugueseSpider	✍PAUQ ✍knowSQL	✍BIRD 🧑‍💻✍Dr.Spider 🤖✍AmbiQT	✍Archer ✍Spider2-lite
Multi-turn				✍SParC ✍CoSQL		✍CHASE			
Robustness						✍Spider-Syn ✍Spider-Realistic		🧑‍💻✍Dr.Spider	
Knowledge						✍KaggleDBQA ✍Spider-DK		✍BIRD	✍Archer
Efficiency								✍BIRD	
Ambiguity								🤖✍AmbiQT	

Fig. 10: Timeline for NL2SQL Benchmarks

Evaluation and Error Analysis

Metrics:

- *Execution Accuracy(EX):* $EX = \sum_{i=1}^{N} \mathbb{1}(V_i = \widehat{V}_i)/N$, $\mathbb{1}(\cdot) = 1$ when $\cdot$ is true, and 0 otherwise.
- *String-Match Acurracy(SM):* $SM = \sum_{i=1}^{N} \mathbb{1}(Y_i = \widehat{Y}_i)/N$.
- *Component-Match Acurracy(CM):* $CM^C = \sum_{i=1}^{N} \mathbb{1}(Y_i^C = \widehat{Y}_i^C)/N$.
- *Exact-Match Accuracy(EM):*$EM = \sum_{i=1}^{N} \mathbb{1}(\wedge_{C_k \in \mathbb{C}} Y_i^{C_k} = \widehat{Y}_i^{C_k})/N$
- *Valid Efficiency Score(VES):*

$$VES = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(V_i = \widehat{V}_i) \cdot \mathbf{R}(Y_i, \widehat{Y}_i), \quad \mathbf{R}(Y_i, \widehat{Y}_i) = \sqrt{\mathbf{E}(Y_i)/\mathbf{E}(\widehat{Y}_i)}$$

- *Query Variance Testing(QVT):*

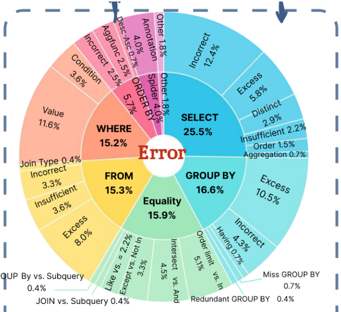
$$QVT = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\sum_{j=1}^{m_i} \mathbb{1}\left(\mathcal{F}(Q_{ij}) = Y_i\right)}{m_i} \right)$$

Evaluation and Error Analysis

Evaluation toolkits: MT-TEQL, NL2SQL360. . .

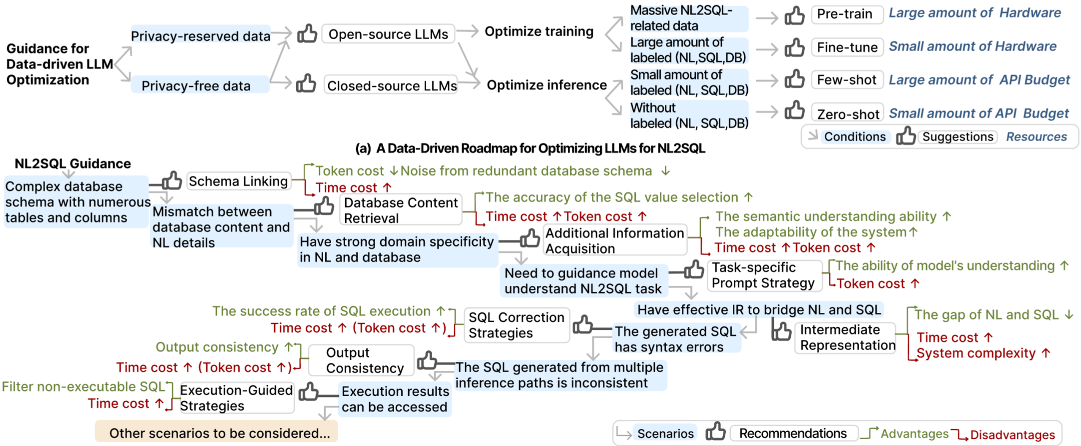
Error analysis taxonomy:

1. *Error Location*: identify specific SQL components where errors occur, such as the `SELECT` or `WHERE` clause.
2. *Cause of Error*: focus on the underlying reasons for the error.



(d) NL2SQL Error Analysis

Practical Guidance for NL2SQL Solutions



Guidance for Data-driven LLM Optimization

Privacy-reserved data → Open-source LLMs

Privacy-free data → Closed-source LLMs

Optimize training

Optimize inference

Massive NL2SQL-related data → Pre-train *Large amount of Hardware*

Large amount of labeled (NL,SQL,DB) → Fine-tune *Small amount of Hardware*

Small amount of labeled (NL, SQL,DB) → Few-shot *Large amount of API Budget*

Without labeled (NL, SQL,DB) → Zero-shot *Small amount of API Budget*

Conditions | Suggestions | *Resources*

(a) A Data-Driven Roadmap for Optimizing LLMs for NL2SQL

NL2SQL Guidance

Complex database schema with numerous tables and columns → Schema Linking: Token cost ↓ Noise from redundant database schema ↓; Time cost ↑

Mismatch between database content and NL details → Database Content Retrieval: The accuracy of the SQL value selection ↑; Time cost ↑ Token cost ↑

Have strong domain specificity in NL and database → Additional Information Acquisition: The semantic understanding ability ↑; The adaptability of the system ↑; Time cost ↑ Token cost ↑

Need to guidance model understand NL2SQL task → Task-specific Prompt Strategy: The ability of model's understanding ↑; Token cost ↑

Have effective IR to bridge NL and SQL → Intermediate Representation: The gap of NL and SQL ↓; Time cost ↑; System complexity ↑

The generated SQL has syntax errors → SQL Correction Strategies: The success rate of SQL execution ↑; Time cost ↑ (Token cost ↑)

The SQL generated from multiple inference paths is inconsistent → Output Consistency: Output consistency ↑; Time cost ↑ (Token cost ↑)

Execution results can be accessed → Execution-Guided Strategies: Filter non-executable SQL; Time cost ↑

Other scenarios to be considered...

Scenarios | Recommendations | Advantages | Disadvantages

(b) Decision Flow for Recommending NL2SQL Modules

Fig. 11: A Data-Driven Roadmap and a Decision Flow for Recommending NL2SQL modules.

Open Problems

The open NL2SQL problems:

- Database retrieval: find relevant databases to access.
- Handling heterogeneous schemas:
- Answer aggregation:
- Domain Adaption:
- Scalability and efficiency:

Developing cost-effective NL2SQL methods:
Make NL2SQL solutions trustworthy:

- Interpreting NL2SQL solutions:
- NL2SQL debugging tools:
- Interactive NL2SQL Tools:

The End