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

Acknowledgement

- The slides accompany our recent NL2SQL survey[1] and handbook[2]. I would like to extend my gratitude to my students and collaborators for their valuable contributions.

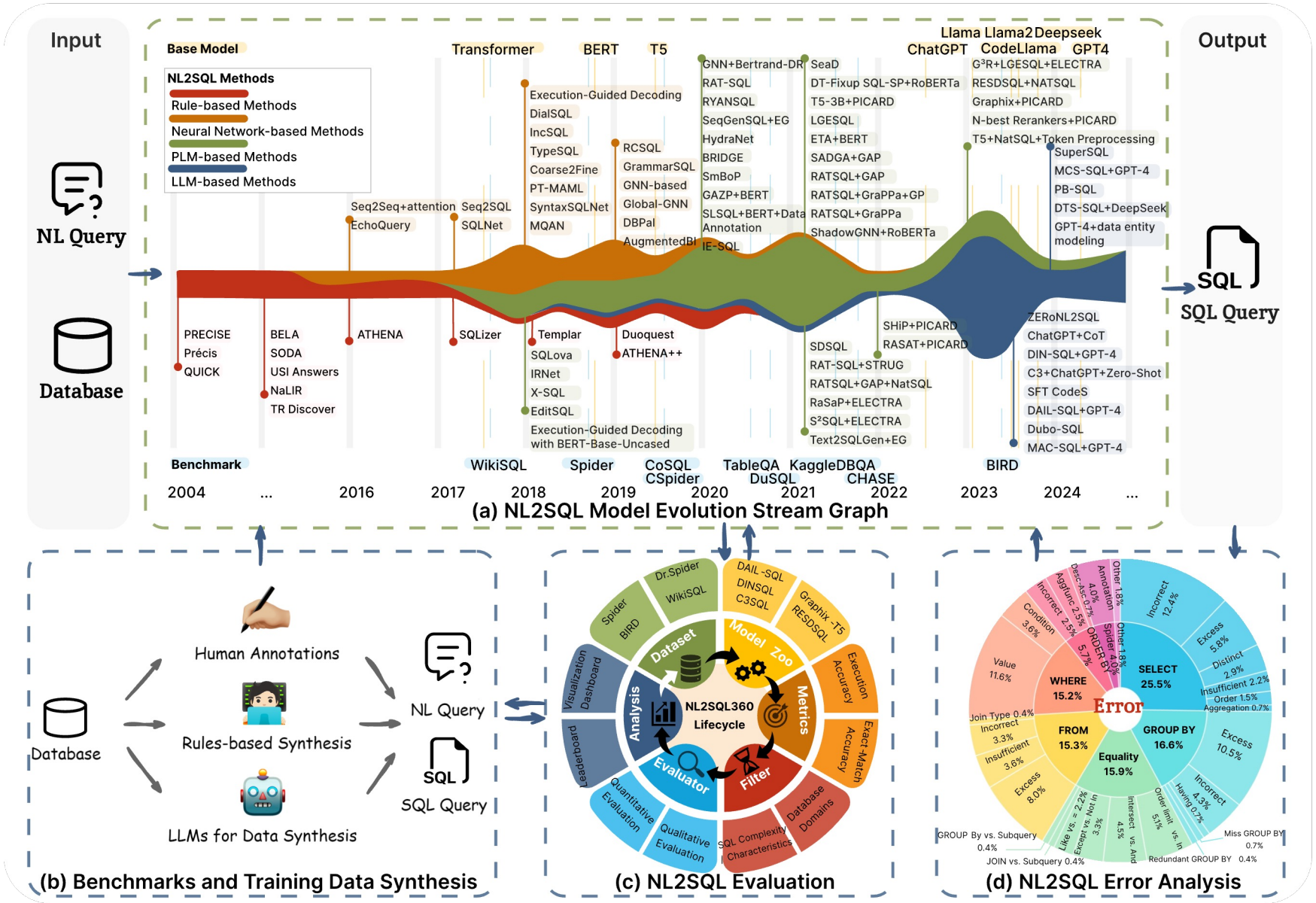
- **All materials in the slides, including the figures, are available for distribution and use for educational and research purposes**. If you find any resources in our survey, handbook, or slides helpful, *please feel free to cite it in your work*[3].
- As NL2SQL is an emerging area, we appreciate your informing us of any significant work we may have overlooked in our survey, so we can address it in future updates.

[1] Liu, Xinyu, et al. "A Survey of NL2SQL with Large Language Models: Where are we, and where are we going?." arXiv preprint arXiv:2408.05109 (2024).

[2] NL2SQL Handbook: https://github.com/HKUSTDial/NL2SQL_Handbook

[3]
```
@misc{liu2024surveynl2sqllargelanguage,
    title={A Survey of NL2SQL with Large Language Models: Where are we, and where are we going?},
    author={Xinyu Liu and Shuyu Shen and Boyan Li and Peixian Ma and Runzhi Jiang and Yuyu Luo and Yuxin Zhang and Ju Fan and Guoliang Li and Nan Tang},
    year={2024},
    eprint={2408.05109},
    archivePrefix={arXiv},
    primaryClass={cs.DB},
    url={https://arxiv.org/abs/2408.05109},
}
```

An Overview of the Survey: the entire lifecycle of NL2SQL



(a) NL2SQL Model Evolution Stream Graph
- Recent advances in NL2SQL solutions, driven by pre-trained language models (PLMs) and large language models (LLMs)

(b) Benchmarks and Training Data Synthesis
- How to collect high-quality training data
- How to synthesize high-quality training data

(c) NL2SQL Evaluation
- Multi-angle evaluation
- Scenario-based evaluation

(d) NL2SQL Error Analysis
- provide a taxonomy to summarize typical errors produced by NL2SQL methods

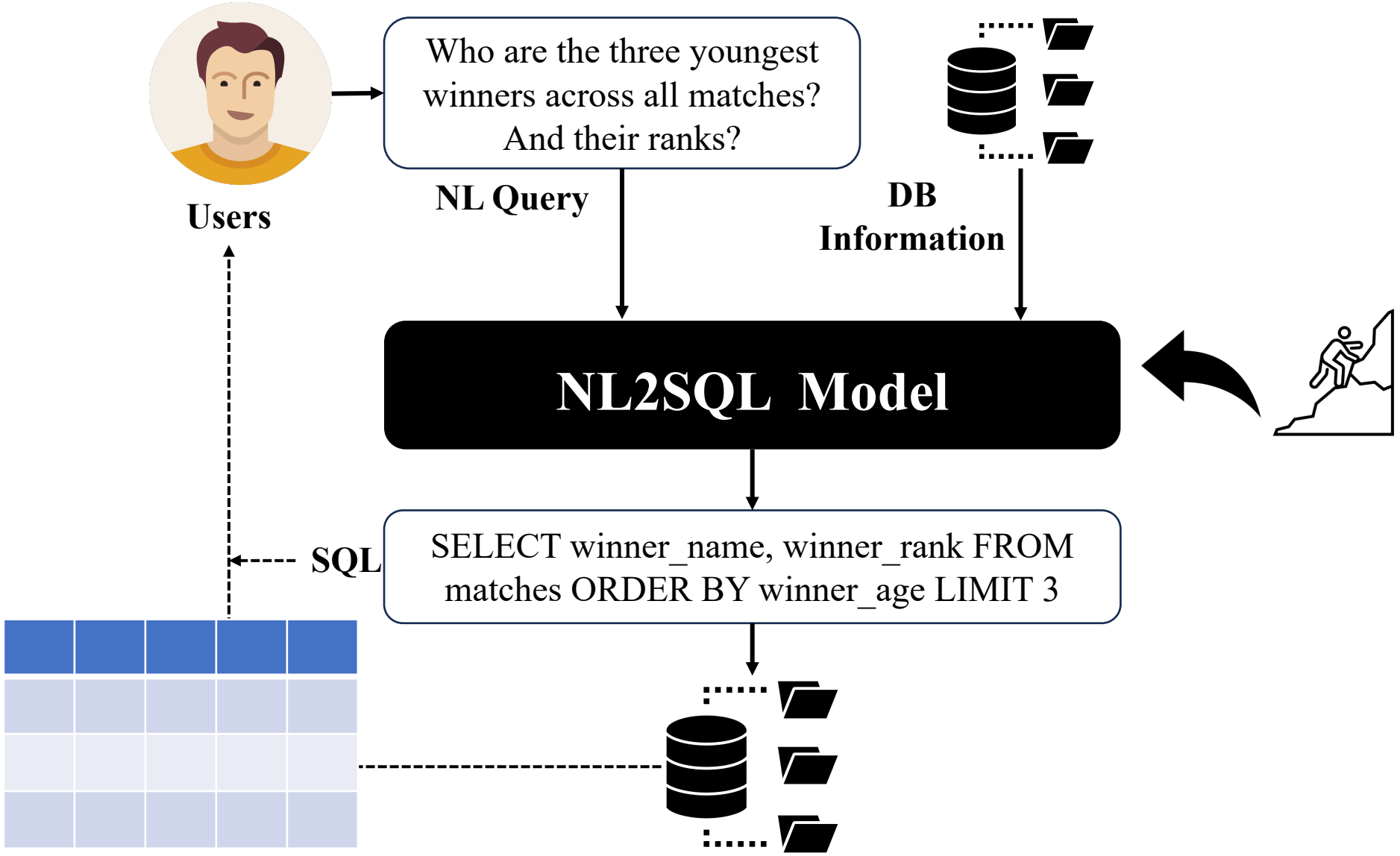
Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

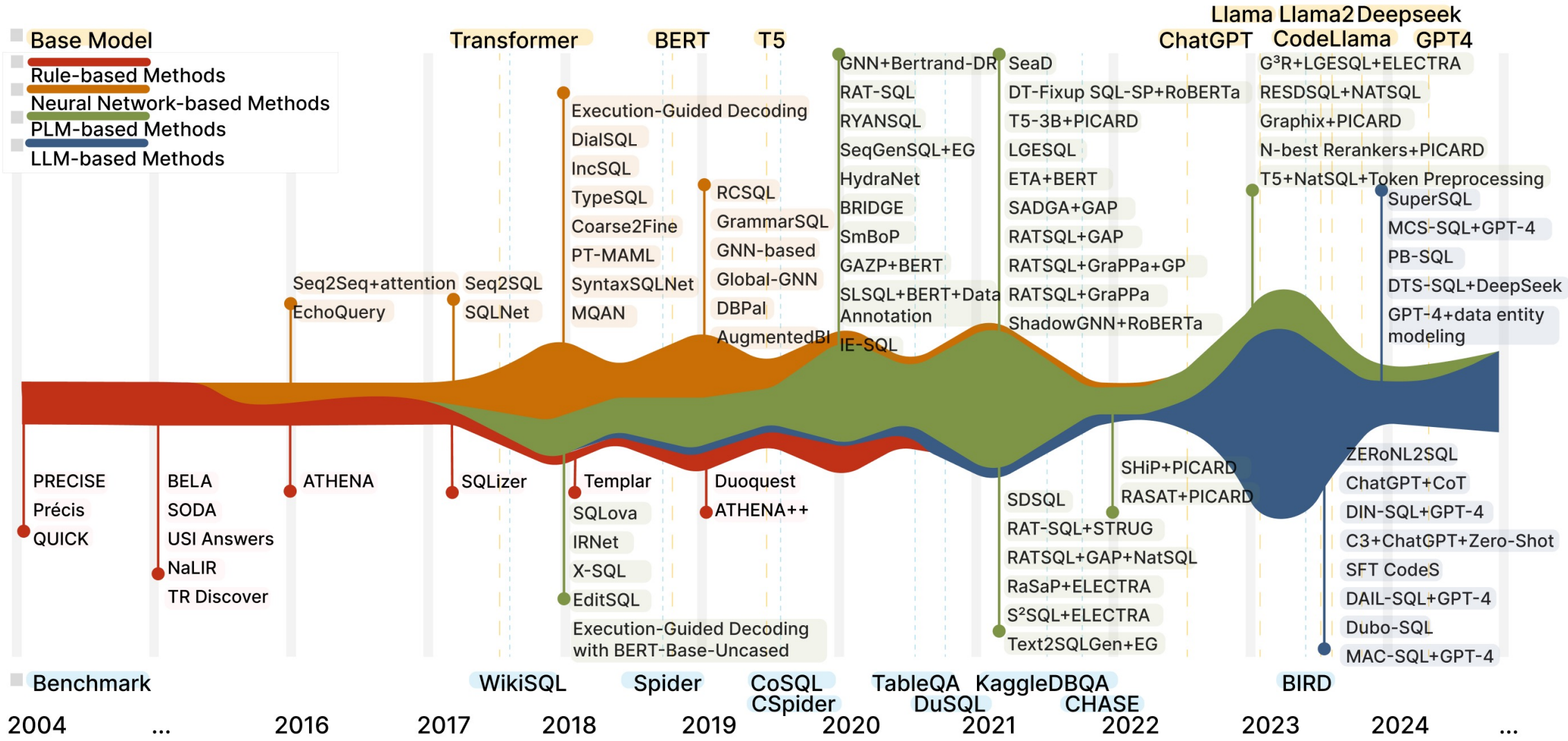
Outline

- NL2SQL Problem and Background
  - ⭐ Problem Formulation
    - NL2SQL Human Workflow
    - NL2SQL Task Challenges
    - Challenges Solving with Large Language Models
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

What is Natural Language to SQL (NL2SQL)?



NL2SQL Model Evolution Stream Graph



Outline

- NL2SQL Problem and Background
  - Problem Formulation
  - ⭐ NL2SQL Human Workflow
  - NL2SQL Task Challenges
  - Challenges Solving with Large Language Models
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

NL2SQL Human Workflow

Step-1 NL Understanding

Find the number of dog pets that are raised by female student

Step-2 Schema Linking and Database Content Retrieval

Pets

PetID	PetType	PetAge	...
	Dog		

Student

StuID	Sex	Age	...
	F		

Has_Pet

PetID	StuID	...

Table Linking
Columns Linking
Database Content
Foreign Key

Step-3 Translating the NL Intent into the SQL

Select count(*) *FROM* student *AS* T1 *JOIN* has_pet *AS* T2 *ON* T1.stuid=T2.stuid
JOIN pets *AS* T3 *ON* T2.petid=T3.petid *WHERE* T1.sex='F' *AND* T3.pettype='Dog'

Outline

- NL2SQL Problem and Background
  - Problem Formulation
  - NL2SQL Human Workflow
  - ⭐ NL2SQL Task Challenges
  - Challenges Solving with Large Language Models
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

NL2SQL Task Challenges

C1. Ambiguous NL Query

C2. Requiring Domain Knowledge

C3. Complex Schema

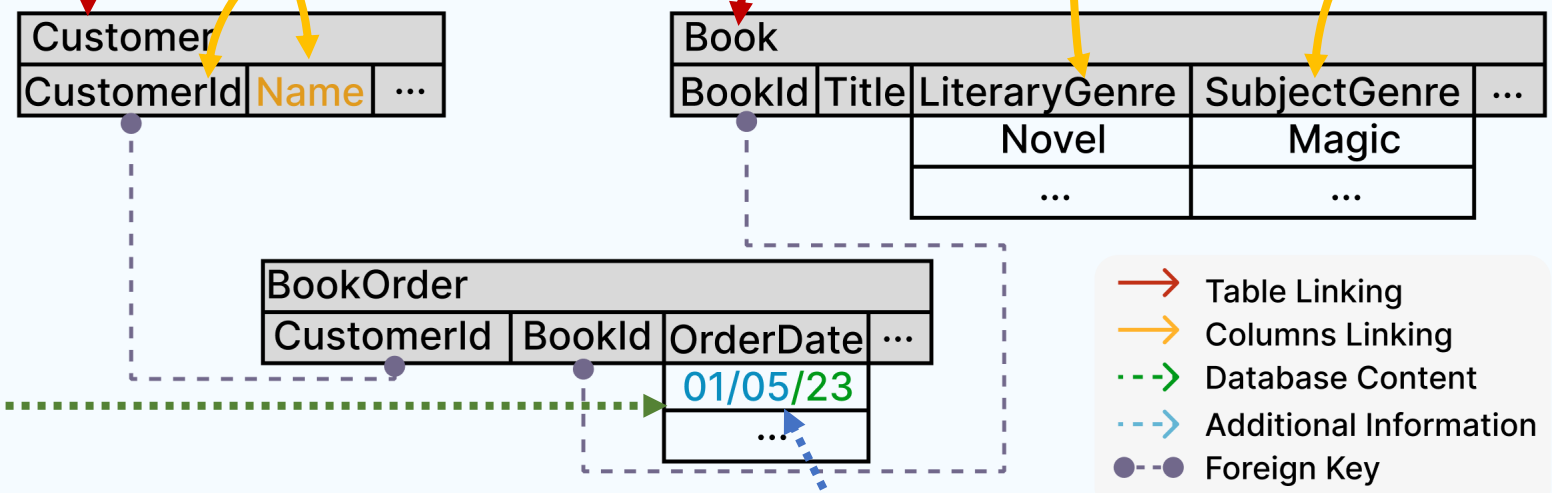
Natural Language Query:

Find the names of all customer who checked out books on exactly 3 different genres on Labor Day in 2023.

C3

Database:

C1



Additional Information: Note that Labor Day stand for May 1

C2

NL2SQL Task Challenges

C1. Ambiguous NL Query

C2. Requiring Domain Knowledge

C3. Complex Schema

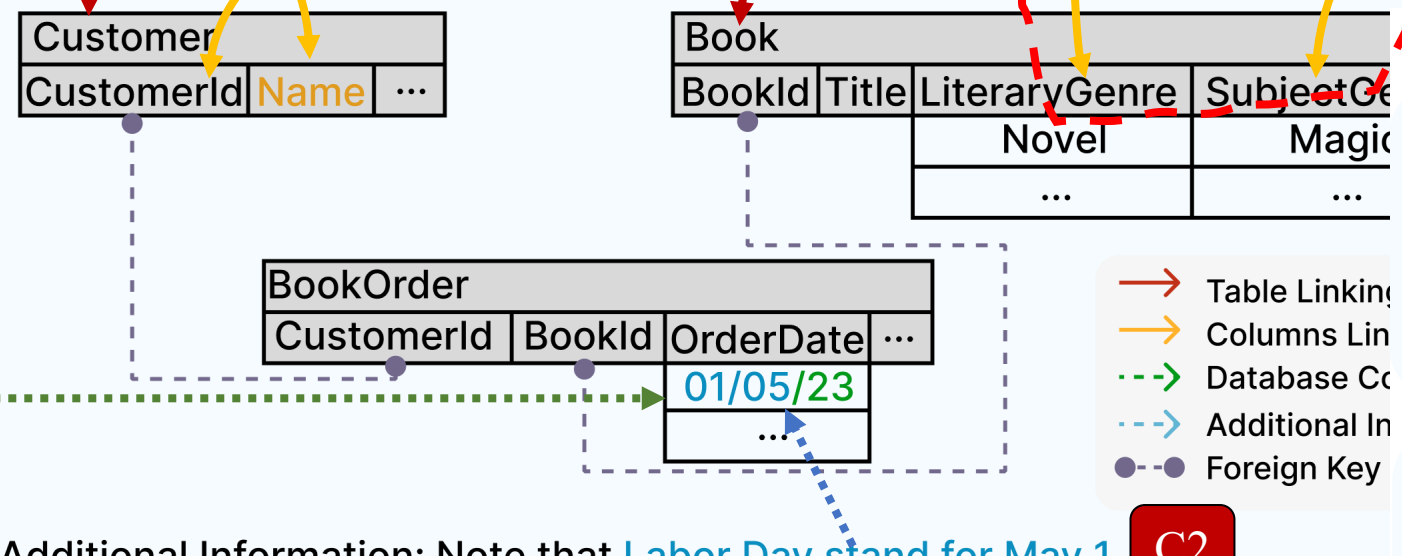
C4. Multiple Possible SQL Queries

Natural Language Query:

Find the names of all customer who checked out books on exactly 3 different genres on Labor Day in 2023.

C3

Database:



Additional Information: Note that Labor Day stand for May 1

C2

C1

SQL:

```
SELECT Name
FROM Customer
NATURAL JOIN BookOrder NATURAL JOIN Book
WHERE OrderDate = '01/05/23'
GROUP BY CustomerId, Name
HAVING COUNT(DISTINCT SubjectGenre) = 3;
```

LiteraryGenre

C1

```
SELECT Name
FROM Customer
WHERE CustomerId = (SELECT CustomerId
FROM BookOrder NATURAL JOIN Book
WHERE OrderDate = '01/05/23'
GROUP BY CustomerId
HAVING COUNT(DISTINCT SubjectGenre)=3);
```

LiteraryGenre

NL2SQL Task Challenges

C1. Ambiguous NL Query

C2. Requiring Domain Knowledge

C3. Complex Schema

C4. Multiple Possible SQL Queries

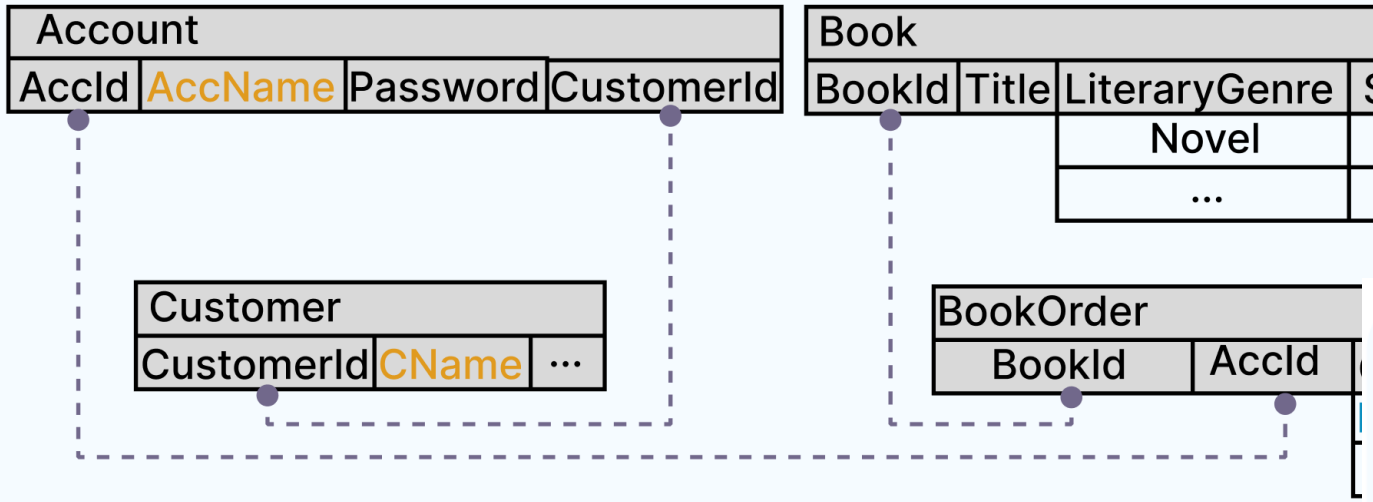
C5. Database Schema Dependency

C6. Database Domain Adaption

Natural Language Query:

Find the names of all customer who checked out books on exactly 3 different genres on Labor Day in 2023.

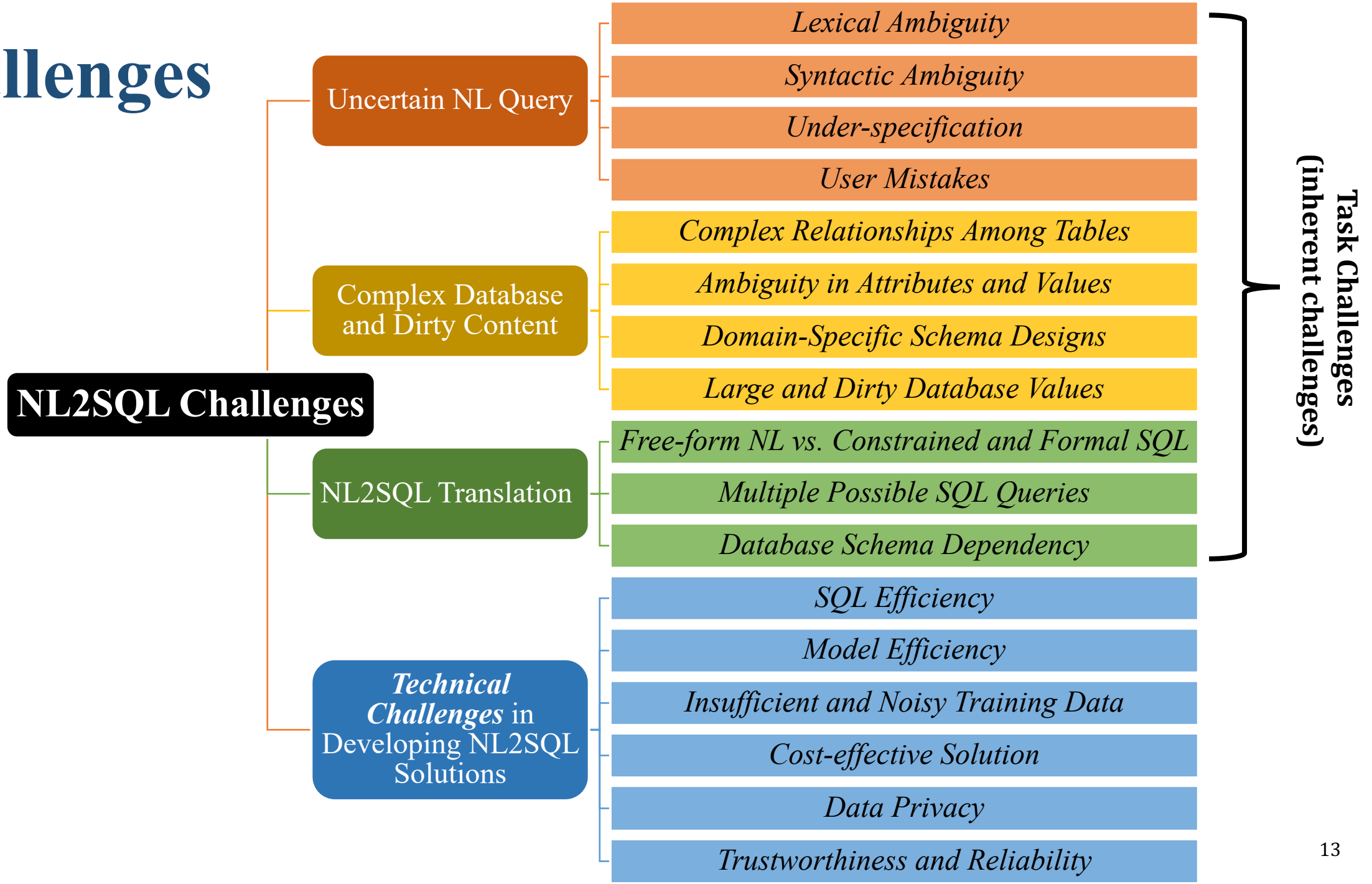
Database:



Additional Information: Note that Labor Day stand for May 1

```sql
SELECT CName
FROM Customer
NATURAL JOIN Account
NATURAL JOIN BookOrder
NATURAL JOIN Book
WHERE OrderDate = 'May 1st 2023'
GROUP BY CustomerId, CName
HAVING COUNT(DISTINCT SubjectGenre) = 3;
```

Challenges


NL2SQL Challenges
Uncertain NL Query
Lexical Ambiguity
Syntactic Ambiguity
Under-specification
User Mistakes
Complex Database and Dirty Content
Complex Relationships Among Tables
Ambiguity in Attributes and Values
Domain-Specific Schema Designs
Large and Dirty Database Values
NL2SQL Translation
Free-form NL vs. Constrained and Formal SQL
Multiple Possible SQL Queries
Database Schema Dependency
Technical Challenges in Developing NL2SQL Solutions
SQL Efficiency
Model Efficiency
Insufficient and Noisy Training Data
Cost-effective Solution
Data Privacy
Trustworthiness and Reliability
Task Challenges (inherent challenges)

Uncertain NL Query

Type	*Explanation*	*Example*
Lexical Ambiguity	A single word with multiple meanings	• The word "bat" can refer to: 1) *an animal*, 2) *a baseball bat*, *or* 3) *the action of swinging*
Syntactic Ambiguity	A sentence can be parsed in multiple ways	• Mary [saw [the man [with the telescope]]] • Mary [saw [the man]] [with the telescope]
Under-specification	Linguistic expressions lack sufficient detail to convey specific intentions or meanings clearly	• Labor Day in 2023 refers to September 4th in the US but May 1st in China
User Mistakes	Spelling mistakes and grammatical errors	• calendar → calender

Complex Database and Dirty Content

Type	*Explanation*	*Example*
Complex Relationships Among Tables	Databases often contain hundreds of tables with complex interrelationships.	• The *FIBEN* Benchmark[1] has 152 tables per database.
Ambiguity in Attributes and Values	Attributes and values within a database can be ambiguous, making it difficult for NL2SQL systems to determine the correct context.	• An attribute or value named "status" could refer to an order's shipping status in the shipping table, while in the payment table, it might represent the payment status.
Large and Dirty Database Values	In large databases, efficiently handling vast data volumes is critical, and data values may sometimes be inaccurate or "dirty".	• In BI scenario, the database may contain thousands of tables. • Missing values, duplicates, or inconsistencies.
Domain-Specific Schema Designs	Different domains often have unique database designs and schema patterns. The variations in schema design across domains make it difficult to develop a one-size-fits-all solution.	• Healthcare databases may include complex relational structures including patient records, medical history, and treatment plans, requiring specialized joins and query patterns.

[1] Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish R. Mittal, Diptikalyan Saha, Karthik Sankaranarayanan: ATHENA++: Natural Language Querying for Complex Nested SQL Queries. Proc. VLDB Endow. 13(11): 2747-2759 (2020)

NL2SQL Translation

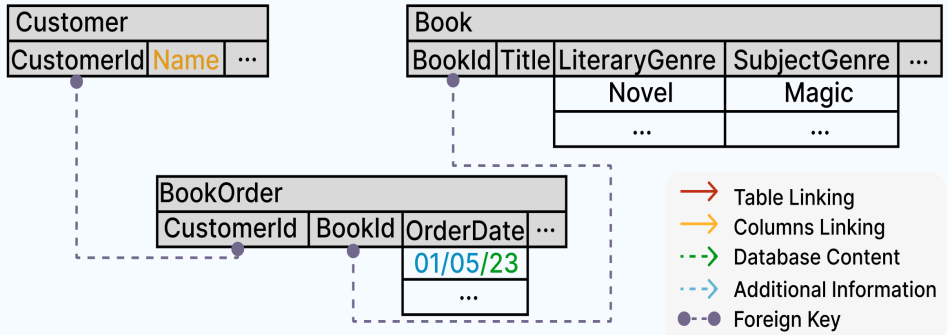
Challenge 1: Free-form NL vs. Constrained and Formal SQL
Natural language is flexible, while SQL queries must follow strict syntax rules.

Challenge 2: Multiple Possible SQL Queries
A single NL query can correspond to multiple SQL queries that fulfill the query intent, leading to ambiguity in determining appropriate SQL translation.

Natural Language Query:

Find the names of all customer who checked out books on exactly 3 different genres on Labor Day in 2023.

Database:



Additional Information: Note that Labor Day stand for May 1

Example

SQL:

```
SELECT Name
FROM Customer
NATURAL JOIN BookOrder NATURAL JOIN Book
WHERE OrderDate = '01/05/23'
GROUP BY CustomerId, Name
HAVING COUNT(DISTINCT SubjectGenre) = 3;
```

```
SELECT Name
FROM Customer
WHERE CustomerId = (SELECT CustomerId
FROM BookOrder NATURAL JOIN Book
WHERE OrderDate = '01/05/23'
GROUP BY CustomerId
HAVING COUNT(DISTINCT SubjectGenre)=3);
```

NL2SQL Translation

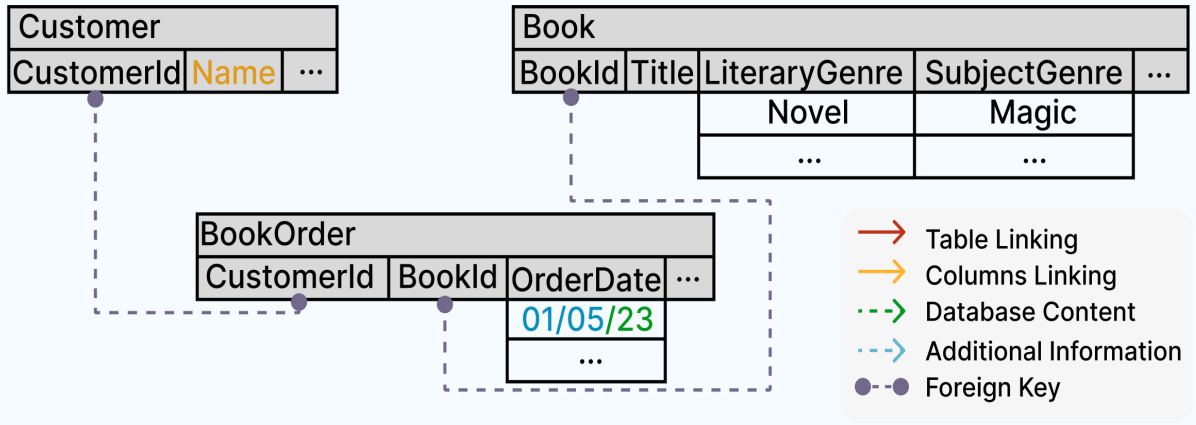
Challenge 3: Database Schema Dependency
The NL2SQL translation process is highly dependent on the database schema it interacts with.

- ***Example: The same user intent can result in different SQL queries for different database schemas***

Natural Language Query:

Find the names of all customer who checked out books on exactly 3 different genres on Labor Day in 2023.

Database:



Additional Information: Note that Labor Day stand for May 1

SQL:

```
SELECT Name
FROM Customer
NATURAL JOIN BookOrder NATURAL JOIN Book
WHERE OrderDate = '01/05/23'
GROUP BY CustomerId, Name
HAVING COUNT(DISTINCT SubjectGenre) = 3;
```

```
SELECT Name
FROM Customer
WHERE CustomerId = (SELECT CustomerId
FROM BookOrder NATURAL JOIN Book
WHERE OrderDate = '01/05/23'
GROUP BY CustomerId
HAVING COUNT(DISTINCT SubjectGenre)=3);
```

NL2SQL Translation

Challenge 3: Database Schema Dependency
The NL2SQL translation process is highly dependent on the database schema it interacts with.

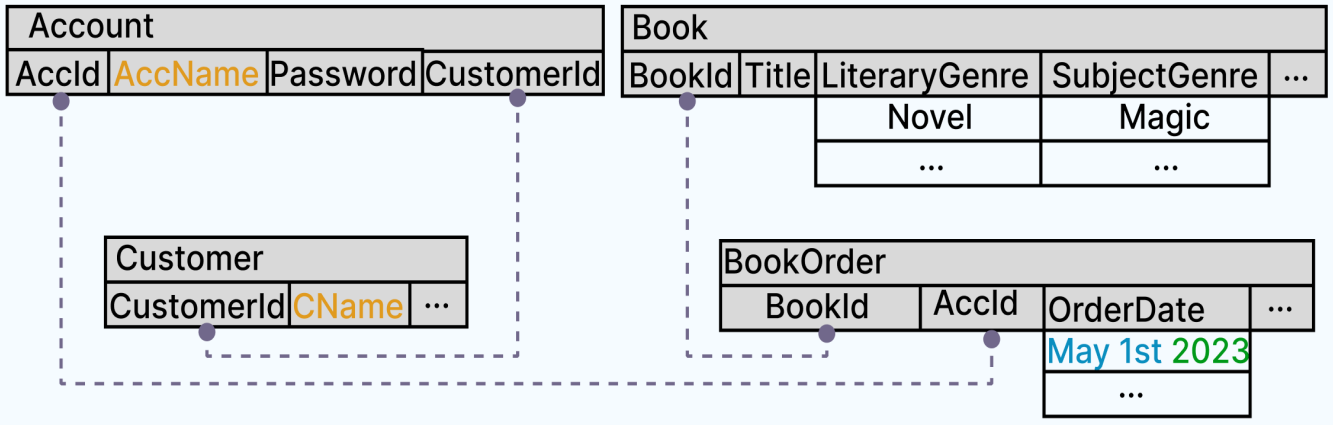
- ***Example: The same user intent can result in different SQL queries for different database schemas***

Natural Language Query:

Find the names of all customer who checked out books on exactly 3 different genres on Labor Day in 2023.

Database:

The update in the DB schema



Additional Information: Note that Labor Day stand for May 1

```sql
SELECT CName
FROM Customer
NATURAL JOIN Account
NATURAL JOIN BookOrder
NATURAL JOIN Book
WHERE OrderDate = 'May 1st 2023'
GROUP BY CustomerId, CName
HAVING COUNT(DISTINCT SubjectGenre) = 3;
```

Technical Challenges in Developing Solutions

Type	*Explanation*
SQL Efficiency	The SQL generated by NL2SQL models must be both correct and optimized for the SQL execution efficiency.
Model Efficiency	There is often a trade-off between model performance and efficiency, as larger models generally deliver better results due to scaling laws but suffer from high latency.
Insufficient and Noisy Training Data	Acquiring high-quality NL2SQL training data is challenging. The limited publicly available datasets are often insufficient for training robust models, and noisy annotations further compromise their quality.

Technical Challenges in Developing Solutions

Type	*Explanation*
Cost-effective Solution	Deploying NL2SQL models, especially those powered by large language models, requires substantial resources, including hardware and API costs.
Data Privacy	Data privacy is crucial for NL2SQL systems, especially when the databases contain sensitive data.
Trustworthiness and Reliability	For NL2SQL models to be widely used, they must be trustworthy and reliable, consistently delivering accurate results across various datasets and use cases. Trustworthiness also requires that the model's decisions are transparent.

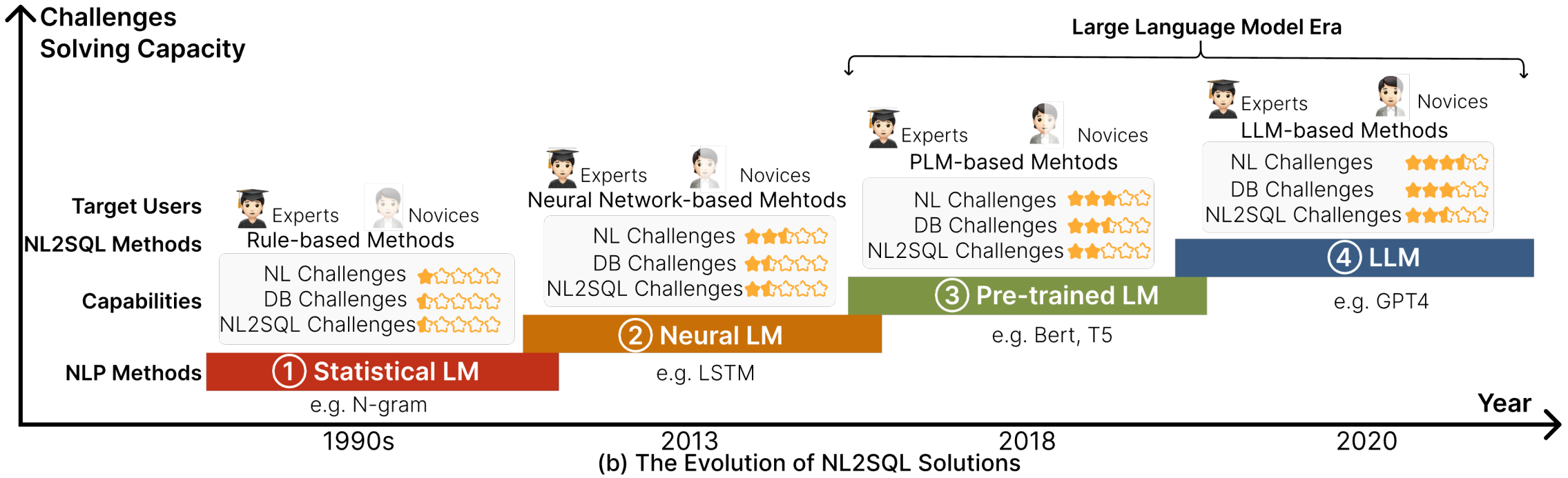
Outline

- NL2SQL Problem and Background
  - Problem Formulation
  - NL2SQL Human Workflow
  - NL2SQL Task Challenges
  - ⭐ Challenges Solving with Language Models
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Challenges Solving with Language Models

Type \ Level	★	★★	★★★	★★★★	★★★★★
NL Challenges	Token-level Recognition	Synonym Recognition	Semantic Understanding	Domain Knowledge Query Recognition	Multi-turn Dialogues
DB Challenges	Single-table Queries	Simple Multiple Tables	Multiple Tables with Complex Schema	Massive Tables and Values	Real-world Databases
NL2SQL Challenges	Single-table SQL	Multi-table SQL	Advanced SQL Feature Support	Adapting to Changed Schema	Efficient SQL Generation

(a) The Definition of Challenges Levels



(b) The Evolution of NL2SQL Solutions

Figure: The Evolution of NL2SQL Solutions from the Perspective of Language Models.

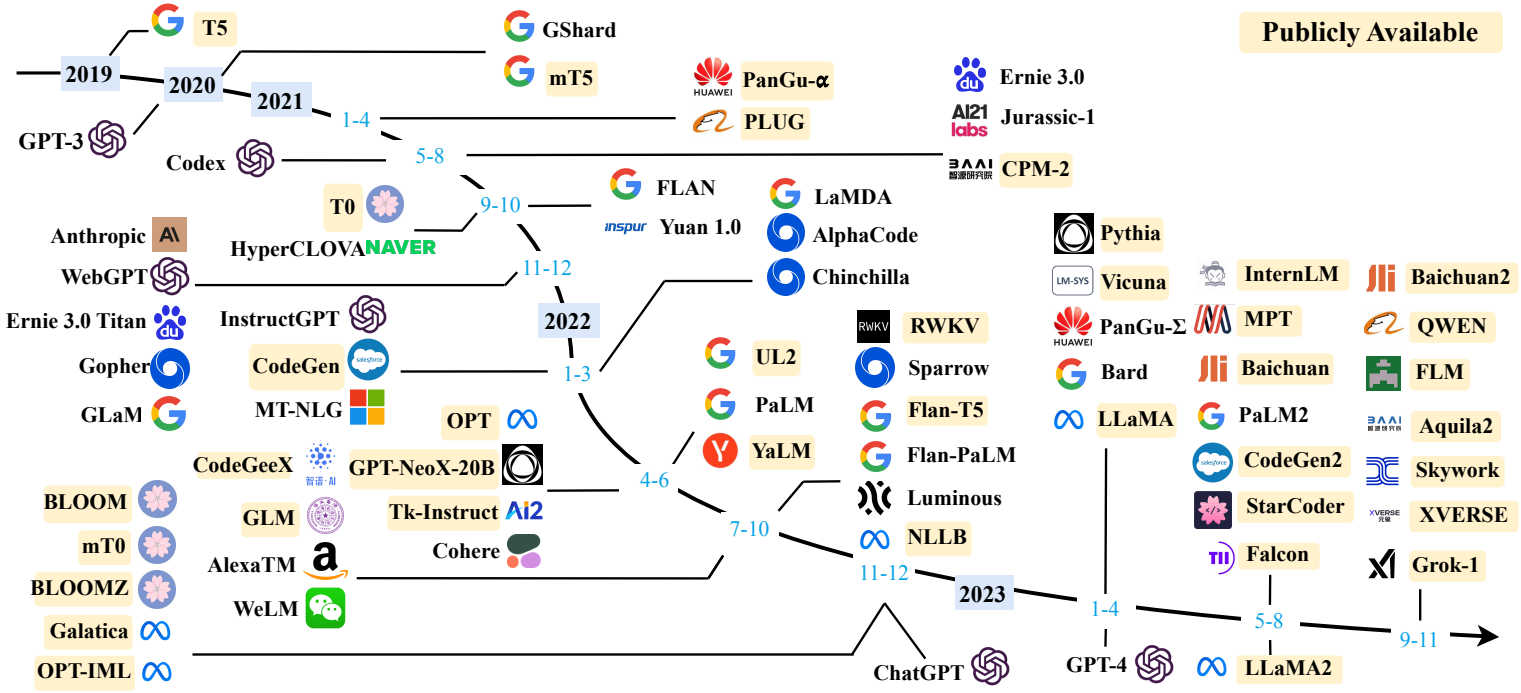
Recap: LLM vs. PLM

PLMs (Pre-trained Language Models):

- Trained on large text corpora
- General language understanding
- Examples: BERT, T5

LLMs (Large Language Models):

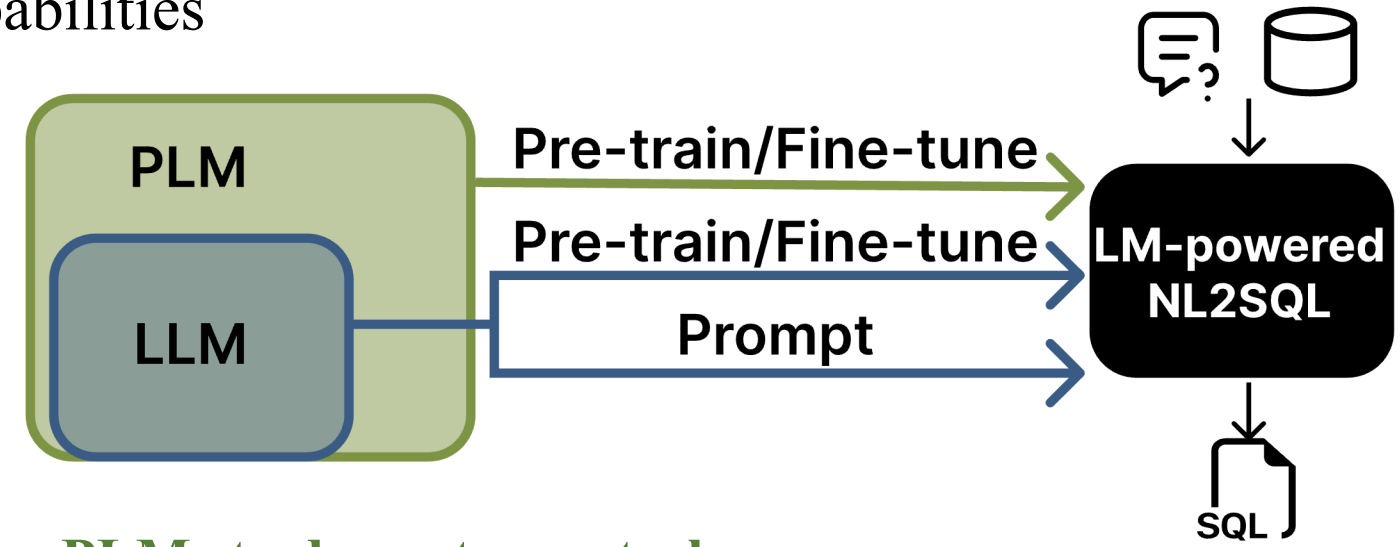
- A subset of PLMs with larger size and more training data
- Superior language understanding and emergent capabilities
- Examples: GPT-4, LLama2



Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Yi Qi, Xingyu Zhao, Kaiwen Cai, Yanghao Zhang, Sihao Wu, Peipei Xu, Dengyu Wu, André Freitas, Mustafa A. Mustafa: A survey of safety and trustworthiness of large language models through the lens of verification and validation. Artif. Intell. Rev. 57(7): 175 (2024)

How to adapt LLM/PLM to NL2SQL?

- The LLMs are a type of PLMs characterized by superior language understanding and emergent capabilities



Steps for applying PLMs to downstream tasks:

1. Pre-training: Training a model on large-scale data to learn general knowledge.
2. Fine-tuning: Adapting a pre-trained model to a specific task using specialized data.

Steps for applying LLMs to downstream tasks:

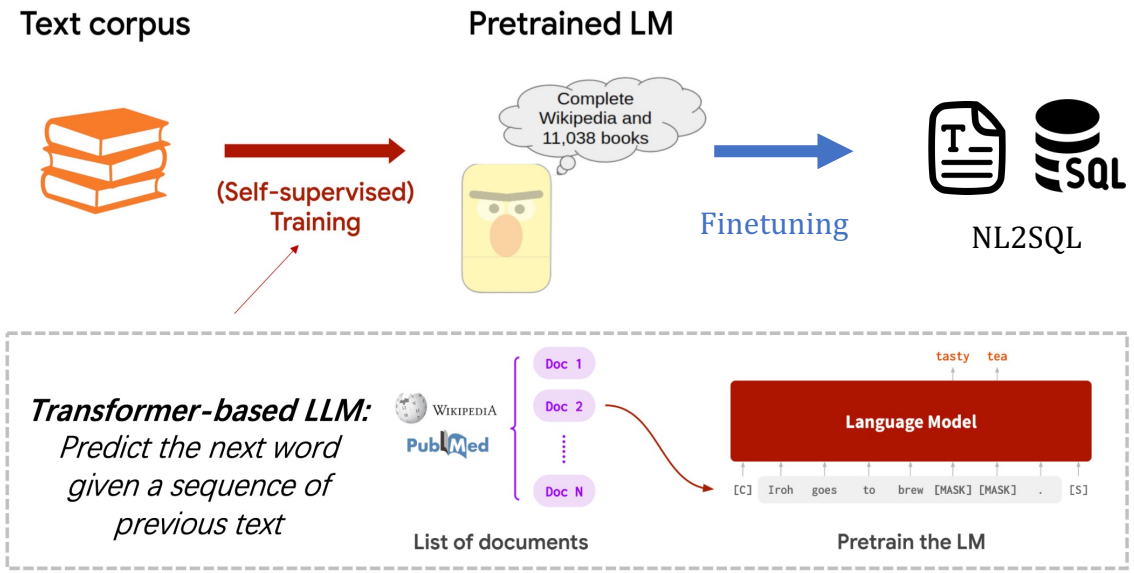
1. Pre-training (optional)
2. Fine-tuning (optional)
3. Prompt the LLMs with different strategies (e.g, In-context learning)

NL2SQL Solution in the Era of LLMs

Pre-train and Fine-tune LLMs for NL2SQL:

$$\text{LLM}^* = \mathcal{F}_{\text{fine-tune}}\left(\mathcal{F}_{\text{pre-train}}(\text{LLM}, \mathcal{D}_p), \mathcal{D}_f\right)$$

- Diverse datasets (D_p) include a broad range of linguistic patterns and domain-general knowledge, enabling the model to develop robust understanding capabilities.
- Specialized datasets (D_f) are closely aligned with the NL2SQL task.



You can get more idea from Section Practical Guidance for Developing NL2SQL Solutions (click it)

https://dbgroup.cs.tsinghua.edu.cn/ligl/papers/tutorial-llm-meets-db.pdf

NL2SQL Solution in the Era of LLMs

In-Context Learning for NL2SQL:

The goal is to optimize the prompt function P to guide the LLMs.

$$\mathcal{F}_{\text{LLM}}(P \mid \text{NL}, \text{DB}, \text{K}) \rightarrow \text{SQL},$$

- **K** denotes additional information or domain-specific knowledge related to NL or DB.
- **P** is a prompt function that transforms the input (NL, DB, K) into a suitable textual prompt for the LLMs.

```
### Instruction ###
/*Below is an instruction that describes a task , paired
with an input that provides further context . Write a
response that appropriately completes the request .*/

### Example ###
/* Some example questions and corresponding SQL queries
are provided based on similar problems : */
/* Answer the following : How many authors are there ? */
SELECT count (*) FROM authors
/* Answer the following : How many farms are there ?. */
SELECT count (*) FROM farm

### DB ###
/* Given the following database schema : */
CREATE TABLE continents (
ContId int primary key ,
Continent text ,
foreign key ( ContId ) references countries ( Continent )
);

### NL ###
/* Answer the following : How many continents are there ?*/
SELECT
```

Example: In-context learning for NL2SQL

Outline

- NL2SQL Problem and Background
- ⭐ Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

An Overview of Language Model-Powered NL2SQL

- **Pre-Processing**
  - Schema Linking
  - Database Content Retrieval
  - Additional Information Acquisition
- **NL2SQL Translation Methods**
  - Encoding Strategy
  - Decoding Strategy
  - Task-specific Prompt Strategies
  - Intermediate Representation
- **Post-Processing**
  - Correction
  - Consistency
  - Execution- Guided
  - N-best Rerankers

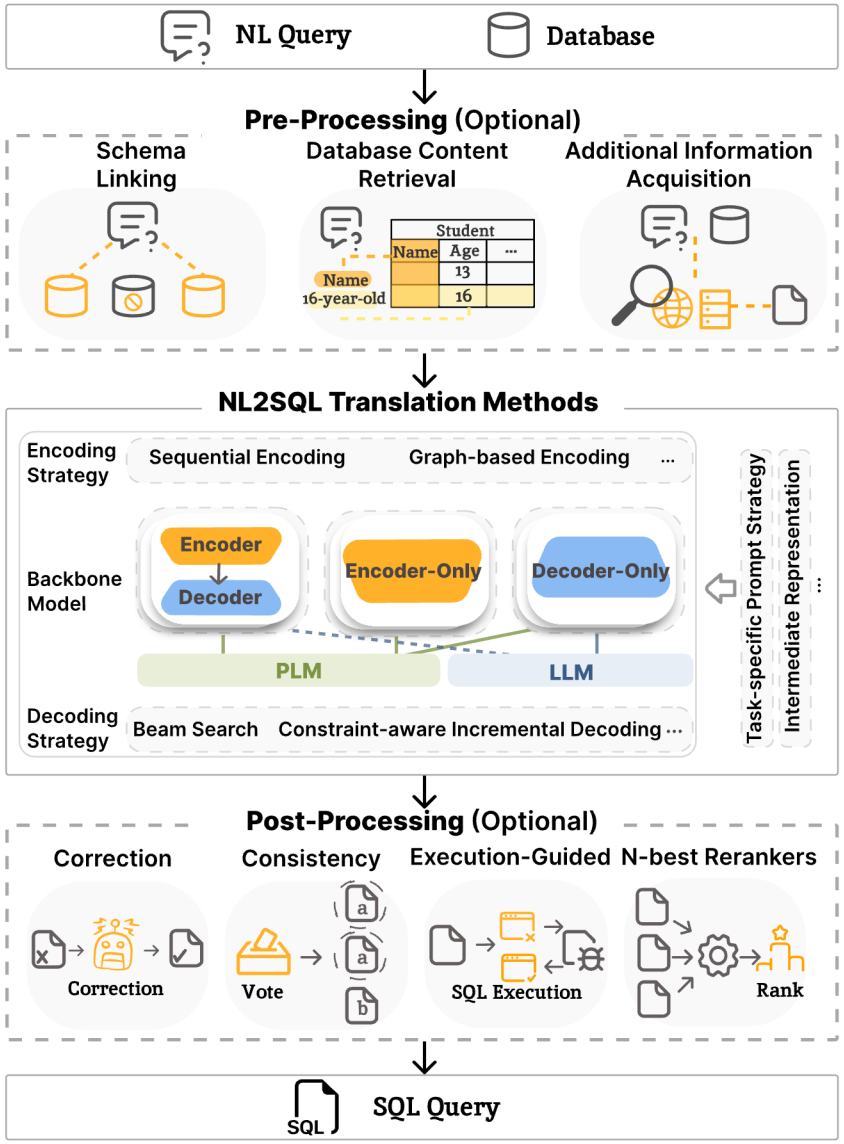


Figure: An Overview of NL2SQL Methods in the LM Era

TABLE I: Comparisons of Existing NL2SQL Solutions.

Methods	Years	Finetuning	Pre-Processing			NL2SQL Translation Methods					Post-Processing			
			Schema Linking	DB Content Retrieval	Additional Information Acquisition	Backbone Model	Encoding Strategy	Intermediate Representation	Task-specific Prompt Strategy	Decoding Strategy	Correction	Consistency	Execution Guided	N-best Rerankers
CHESS [51]	2024	-	✓	✓	✓	Decoder-Only	Sequential Encoding	-	COT	Greedy Search	✓	✓	✓	-
CodeS [46]	2024	-	✓	✓	-	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	-	✓	-
SFT CodeS [46]	2024	✓	✓	✓	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	-	✓	-
FinSQL [47]	2024	✓	✓	-	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	✓	✓	-	-
DTS-SQL [52]	2024	✓	✓	-	-	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	-	-	-
TA-SQL [53]	2024	-	✓	-	-	Decoder-Only	Sequential Encoding	Sketch Structure	COT	Greedy Search	-	-	-	-
SuperSQL [45]	2024	-	✓	✓	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	✓	-	-
ZeroNL2SQL [44]	2024	✓	-	-	-	Encoder-Decoder	Sequential Encoding	Sketch Structure	Decomposition	Beam Search	✓	-	✓	-
PET-SQL [54]	2024	✓	✓	-	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	✓	-	-
CoE-SQL [55]	2024	-	-	-	✓	Decoder-Only	Sequential Encoding	-	CoT	Greedy Search	✓	-	-	-
PURPLE [56]	2024	-	✓	-	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	✓	✓	✓	-
MetaSQL [57]	2024	-	✓	-	✓	Decoder-Only	Sequential Encoding	-	Decomposition	Greedy Search	-	-	-	✓
DEA-SQL [58]	2024	-	✓	-	✓	Decoder-Only	Sequential Encoding	-	Decomposition	Greedy Search	✓	-	-	-
DIN-SQL [5]	2023	-	✓	-	✓	Decoder-Only	Sequential Encoding	Syntax Language	Decomposition	Greedy Search	✓	-	-	-
DAIL-SQL [6]	2023	-	-	-	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	✓	-	-
C3-SQL [59]	2023	-	✓	-	-	Decoder-Only	Sequential Encoding	-	COT	Greedy Search	-	✓	-	-
RESDSQL [7]	2023	✓	✓	✓	-	Encoder-Decoder	Sequential Encoding	Syntax Language	Decomposition	Beam Search	-	-	-	-
T5-3B+NatSQL+Token Preprocessing [60]	2023	✓	✓	✓	-	Encoder-Decoder	Sequential Encoding	Syntax Language	-	Greedy Search	-	-	-	-
ACT-SQL [61]	2023	-	✓	-	✓	Decoder-Only	Sequential Encoding	-	CoT	Greedy Search	-	-	-	-
ODIS [62]	2023	-	-	-	✓	Decoder-Only	Sequential Encoding	-	-	Greedy Search	-	-	-	-
MAC-SQL [63]	2023	-	✓	-	-	Decoder-Only	Sequential Encoding	-	Decomposition	Greedy Search	✓	-	✓	-
SC-Prompt [64]	2023	✓	-	-	-	Encoder-Decoder	Separate Encoding	Sketch Structure	-	Beam Search	✓	-	-	-
CatSQL [65]	2023	✓	-	-	-	Encoder-Only	Sequential Encoding	Sketch Structure	-	Beam Search	✓	-	-	-
SQLFormer [66]	2023	✓	✓	-	-	Encoder-Decoder	Graph-based Encoding	-	-	Beam Search	-	-	-	-
G³R [67]	2023	✓	✓	✓	-	Encoder-Only	Graph-based Encoding	-	COT	Beam Search	-	-	-	✓
Graphix-T5 [43]	2022	✓	✓	✓	-	Encoder-Decoder	Graph-based Encoding	-	-	Constraint-aware Incremental	-	-	-	-
SHiP [68]	2022	✓	-	✓	-	Encoder-Decoder	Graph-based Encoding	-	-	Constraint-aware Incremental	-	-	-	-
N-best List Rerankers [69]	2022	✓	✓	✓	-	Encoder-Decoder	Sequential Encoding	-	-	Constraint-aware Incremental	-	-	-	✓
RASAT [70]	2022	✓	-	✓	-	Encoder-Decoder	Graph-based Encoding	-	-	Constraint-aware Incremental	-	-	-	-
PICARD [71]	2022	✓	-	✓	-	Encoder-Decoder	Sequential Encoding	-	-	Constraint-aware Incremental	-	-	-	-
TKK [72]	2022	✓	-	✓	-	Encoder-Decoder	Separate Encoding	Sketch Structure	Decomposition	Constraint-aware Incremental	-	-	-	-
S²SQL [73]	2022	✓	✓	✓	-	Encoder-Only	Graph-based Encoding	-	-	Greedy Search	-	-	-	-
RAT-SQL [74]	2021	✓	✓	✓	-	Encoder-Only	Graph-based Encoding	Syntax Language	-	Beam Search	-	-	-	-
SmBoP [75]	2021	✓	-	✓	-	Encoder-Only	Graph-based Encoding	-	-	Beam Search	-	-	-	-
RaSaP [76]	2021	✓	✓	✓	-	Encoder-Only	Graph-based Encoding	-	-	Beam Search	-	-	-	-
BRIDGE [77]	2020	✓	-	✓	-	Encoder-Only	Sequential Encoding	-	-	Others	-	-	-	-

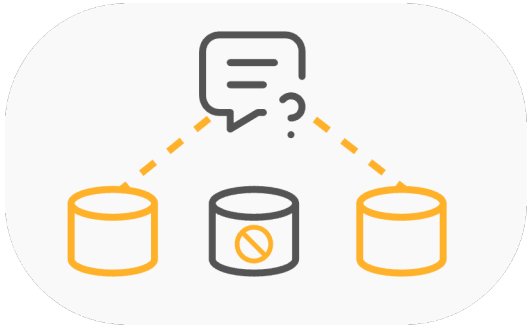
Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - ⭐ Pre-Processing
    - Schema Linking
    - Database Content Retrieval
    - Additional Information Acquisition
  - NL2SQL Translation Methods
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

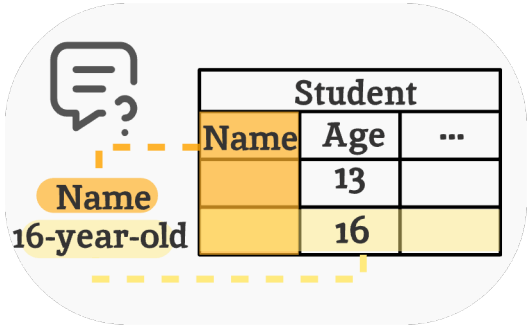
Pre-Processing

Motivation:

- The pre-processing ensures the accurate mapping and processing of key information within the limited input, by identifying the tables and columns related to the given NL query.
- The pre-processing serves as an enhancement to the model's inputs in the NL2SQL translation.



Schema Linking



Database Content Retrieval

Additional Information Acquisition

Schema Linking

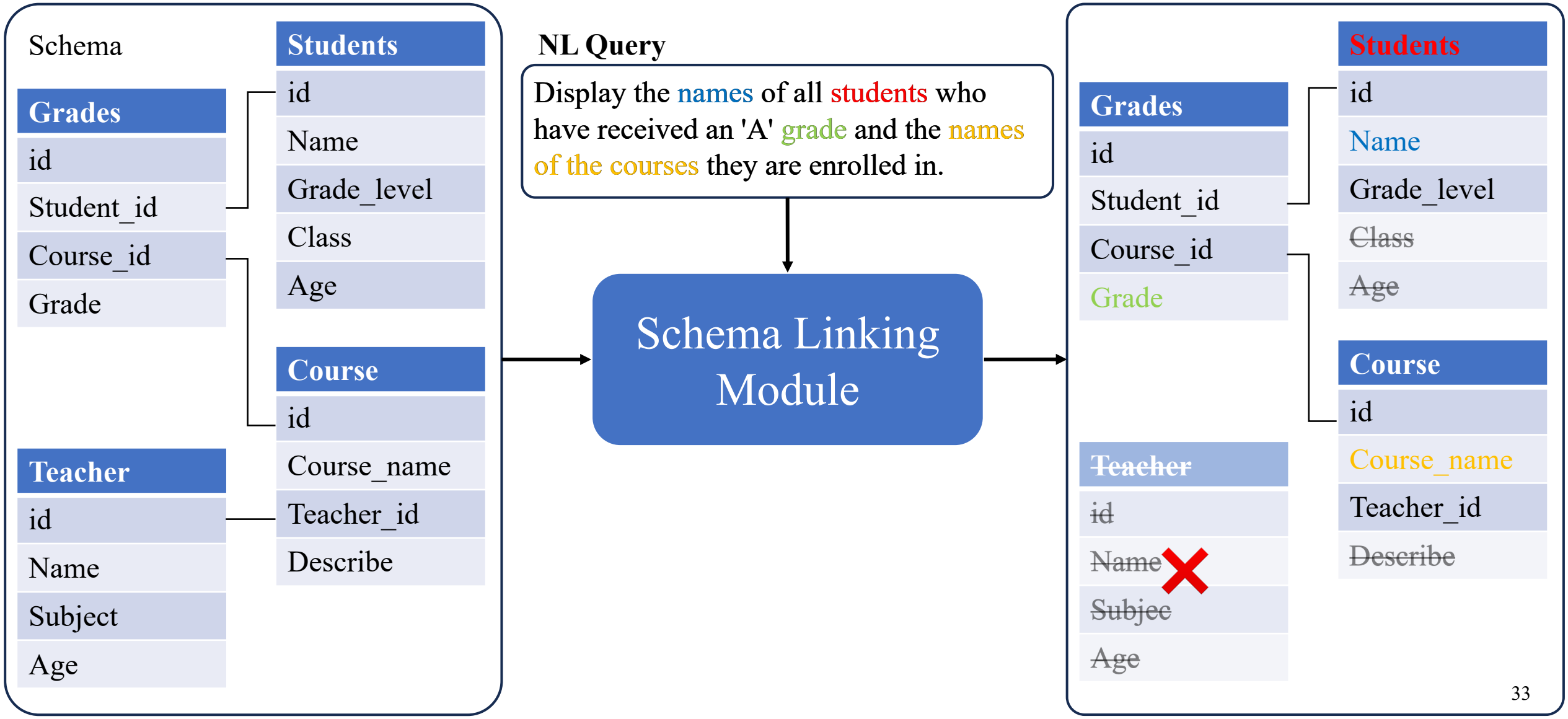
Motivation:

- Schema complexity and presence of irrelevant tables with NL in database.
- The model has input limitations, e.g., the input limit of ChatGPT is 4096 tokens.

Goal:

- Identify the tables and columns related to the given NL.

Workflow of Schema Linking



Method Classification of Schema Linking

Method classification：

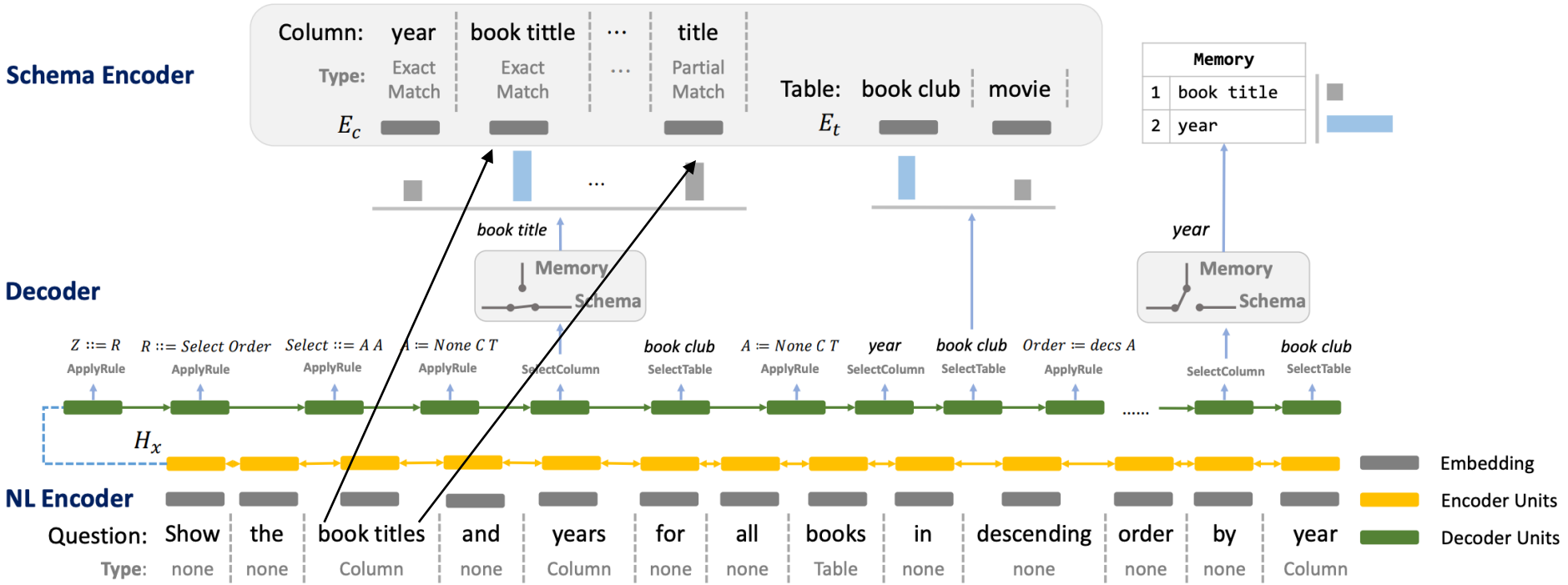
- String Matching-based Schema Linking
- Neural Network-based Schema Linking
- In-Context Learning for Schema Linking

Schema Linking

String Matching-based Schema Linking

IRNet designs string-based schema linking and an intermediate representation called SemQL.

- It uses n-grams (1-6 length) extracted from user queries as query candidates
- It evaluates the string-level similarity between the query candidates and the database schema, then classifies them as Exact or Partial matches.



Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, Dongmei Zhang:
Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. ACL (1) 2019: 4524-4535

String Matching-based Schema Linking

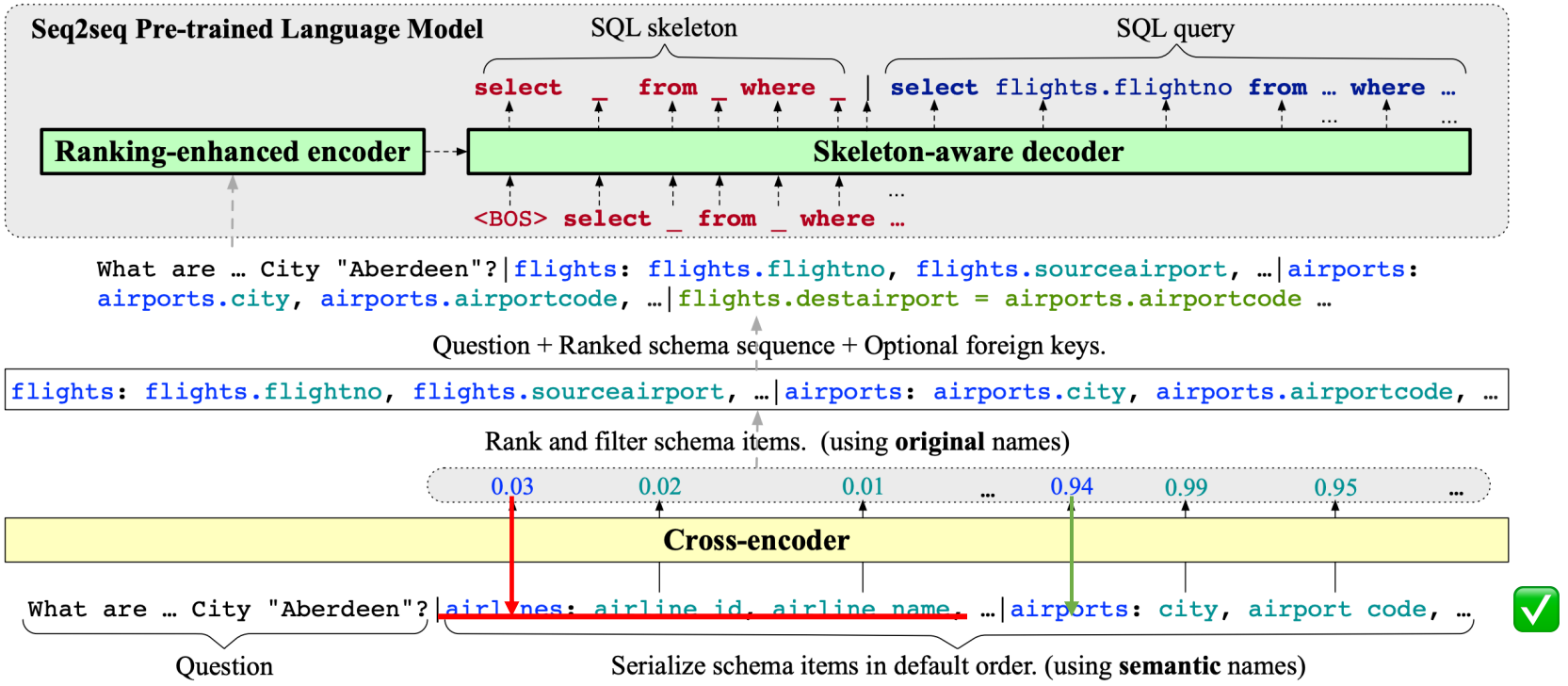
Limitations:

- False positives when candidates share common words.
- Lack the ability to handle synonyms.

Neural Network-based Schema Linking

RESDSQL proposes a ranking-enhanced encoding and skeleton-aware decoding framework.

- A **cross-encoder** is trained to classify tables and columns based on the input query.

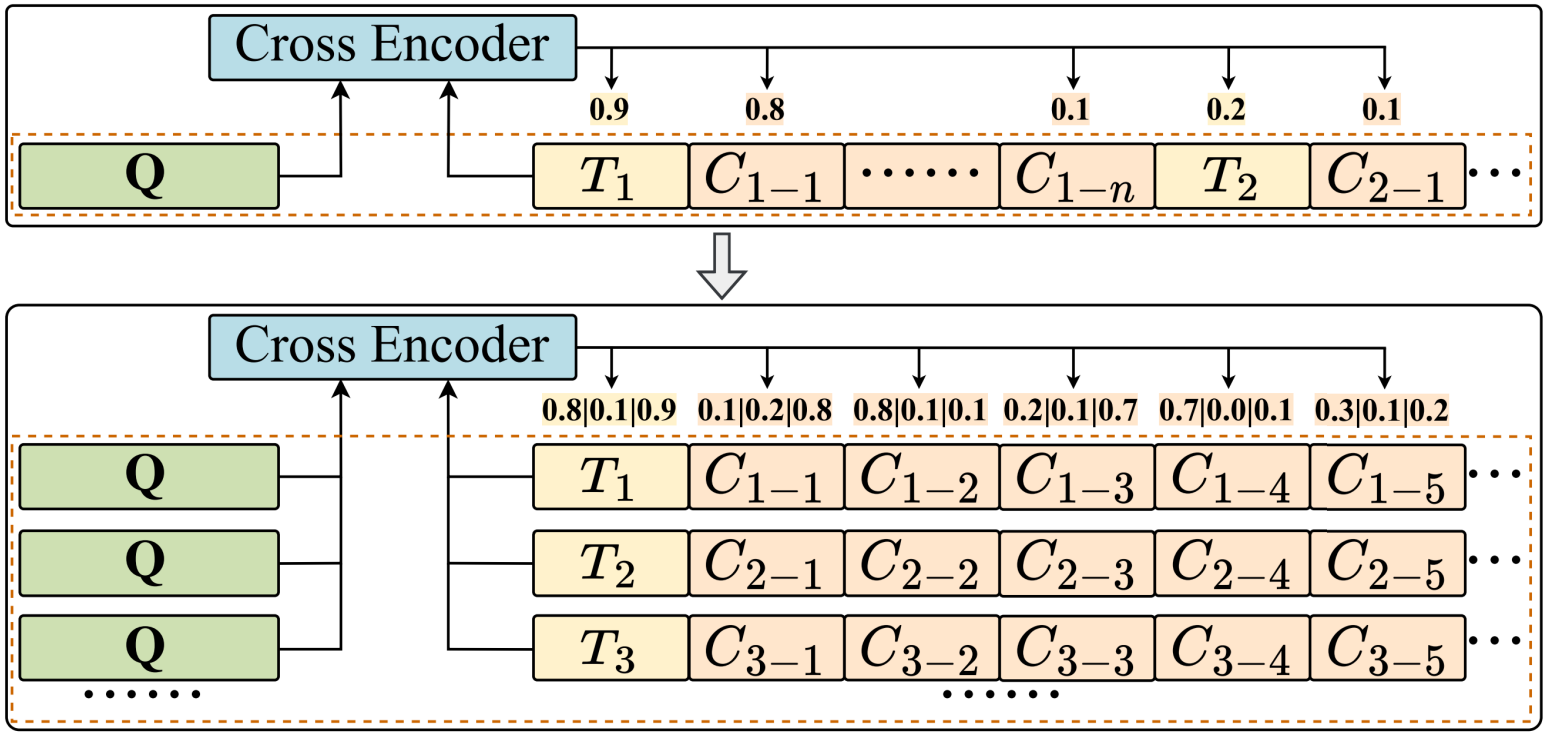


Haoyang Li, Jing Zhang, Cuiping Li, Hong Chen:
RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. AAAI 2023: 13067-13075

Neural Network-based Schema Linking

FinSQL, a model-agnostic LLMs-based NL2SQL framework for financial analysis.

- Organizing the tables into a **batch**
- Devising a **Parallel Cross-Encoder** to retrieve relevant tables and columns from hundreds of schema items



Original Cross-Encoder (RESDSQL)

Parallel Cross-Encoder (FinSQL)

Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, Jinshu Lin: FinSQL: Model-Agnostic LLMs-based Text-to-SQL Framework for Financial Analysis. SIGMOD Conference Companion 2024: 93-105

Neural Network-based Schema Linking

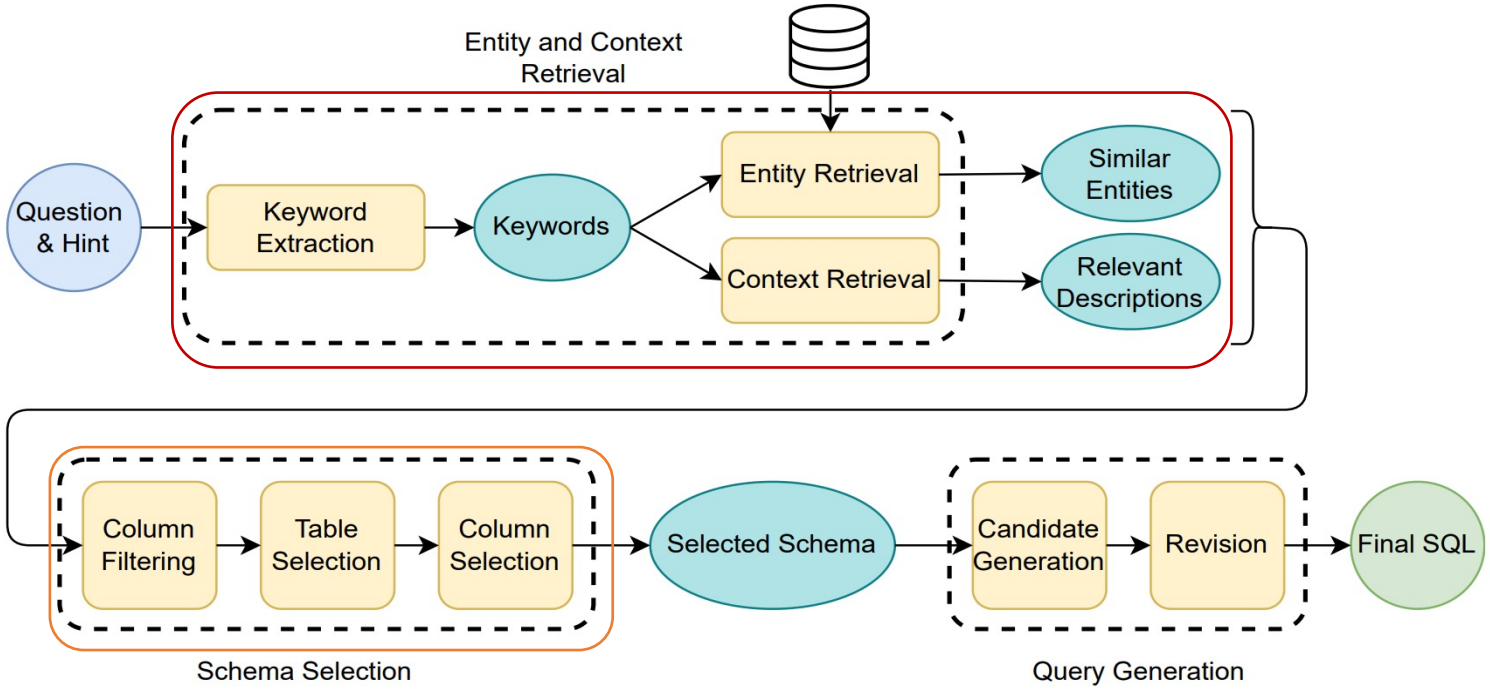
Limitations:

- Highly rely on the quantity and quality of training data
- Struggle to generalize effectively across databases with new schemas or domains

In-Context Learning for Schema Linking

CHESS breaks down the NL2SQL task into a 3-staged pipeline, including entity and context retrieval, schema selection, and query generation.

- Utilizing GPT-4 to extract keywords from both NL and Hint.
- It implements an efficient three-stage schema selection with different prompts.

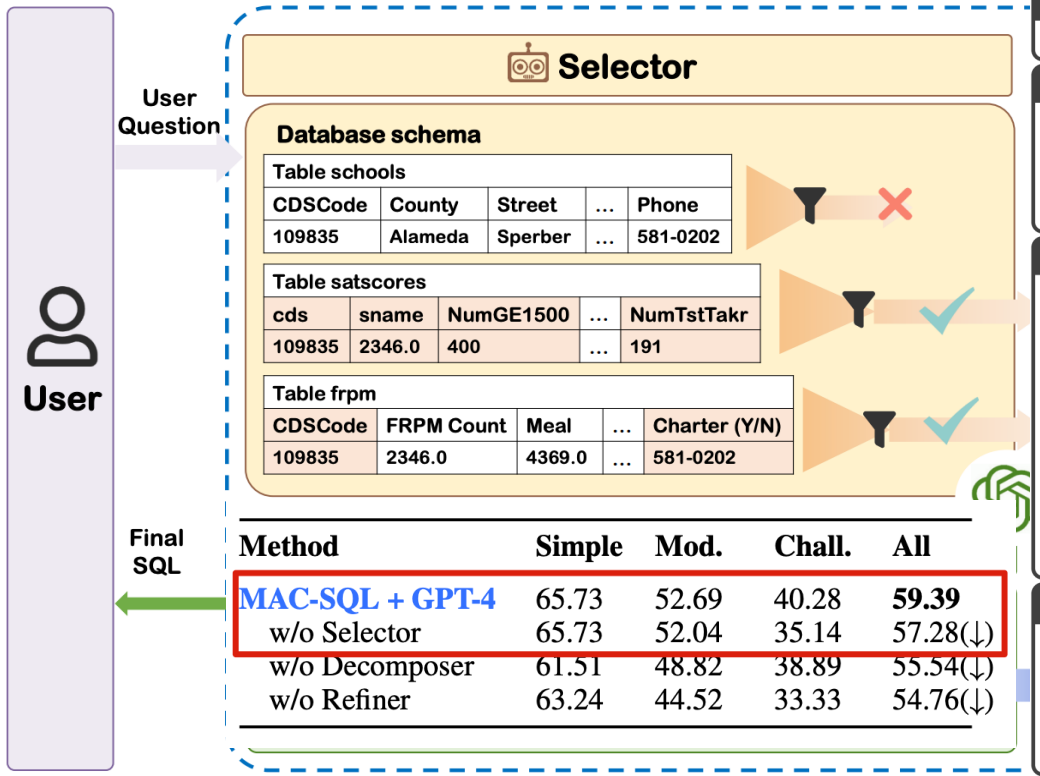


Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, Amin Saberi:
CHESS: Contextual Harnessing for Efficient SQL Synthesis. CoRR abs/2405.16755 (2024) Shayan Talaei, Mohammadreza

In-Context Learning for Schema Linking

MAC-SQL is a multi-agent collaborative framework for NL2SQL.

- The **Selector agent** performs the schema linking task.
- The Selector is activated only when the length of the database schema prompt exceeds a specified threshold.



Method	Simple	Mod.	Chall.	All
MAC-SQL + GPT-4	65.73	52.69	40.28	**59.39**
w/o Selector	65.73	52.04	35.14	57.28(↓)
w/o Decomposer	61.51	48.82	38.89	55.54(↓)
w/o Refiner	63.24	44.52	33.33	54.76(↓)

Task Description

```
As an experienced and professional database administrator, your task is to ...
```

Instruction

```
1. Discard any table schema that is not related to the user question and evidence.
2. Sort the columns in each relevant table in descending order of relevance and keep the top 6
 columns.
3. Ensure that at least 3 tables are included in the final output JSON.
4. The output should be in JSON format.
```

Demonstration

```
[DB_ID] banking_system
[Schema] Table schemas of account, client, loan, district ...
[Foreign keys] ...
[Question]: What is the gender of the youngest client who opened account in the lowest average
 salary branch?
[Evidence]: Later birthdate refers to younger age; A11 refers to average salary
[Answer]
```json
{ "account": "keep_all",
  "client": "keep_all",
  "loan": "drop_all",
  "district": ["district_id", "A11", "A2", ...] }
```
```

Test Question

```
[DB_ID] {db_id}
[Schema] {desc_str}
[Foreign keys] {fk_str}
[Question] {query}
[Evidence] {evidence}
[Answer]
```

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, Zhoujun Li: MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. CoRR abs/2312.11242 (2023)

In-Context Learning for Schema Linking

Limitations:

- Large schema may exceed context length limitation
- Highly rely on the quality of the prompt
- High API cost

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
    - Schema Linking
    - ⭐ Database Content Retrieval
    - Additional Information Acquisition
  - NL2SQL Translation Methods
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Database Content Retrieval

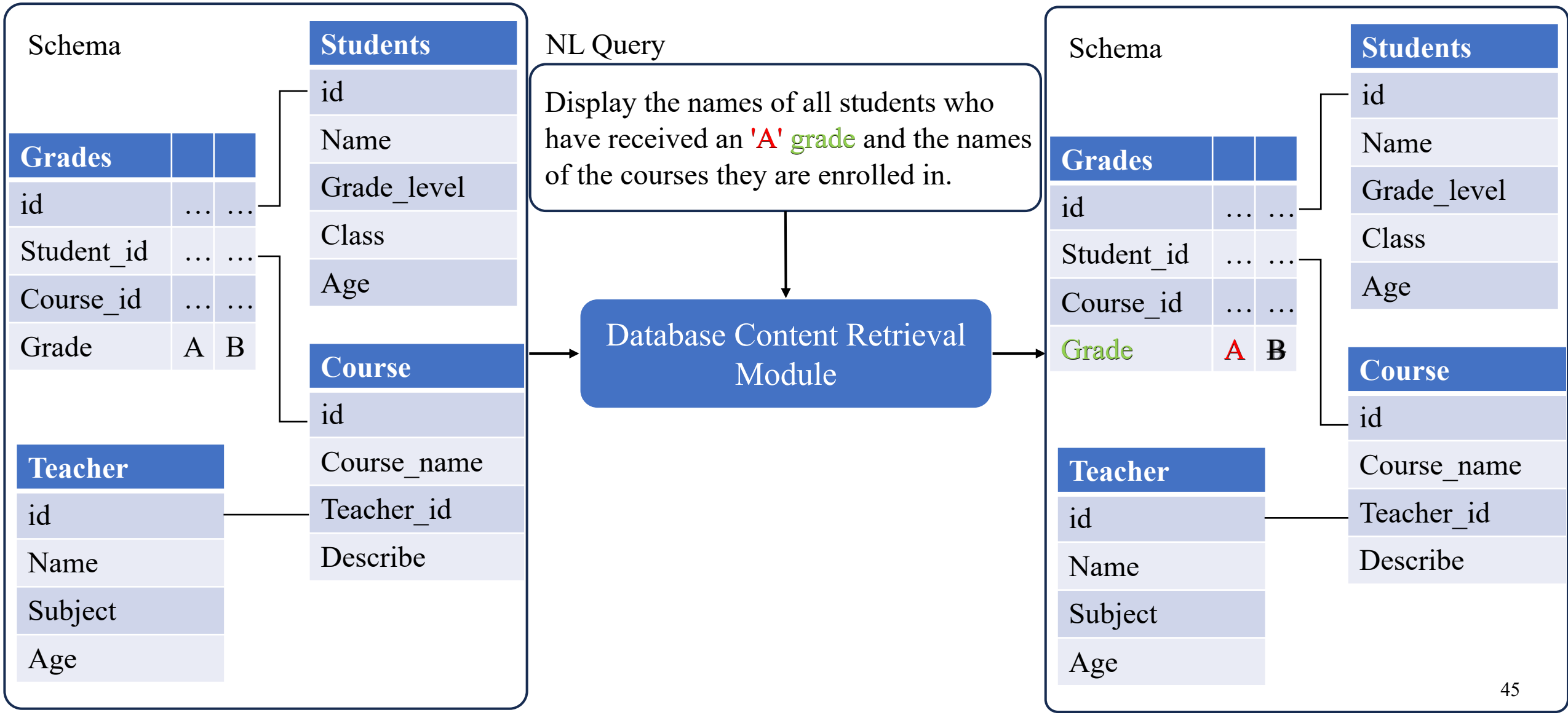
Motivation:

- Due to the large databases and input limitations of the model, retrieving cell values is resource-intensive.
- Effectively handle some SQL clauses such as WHERE and JOIN .

Goal:

- **Efficiently retrieve cell values** related to the given NL.

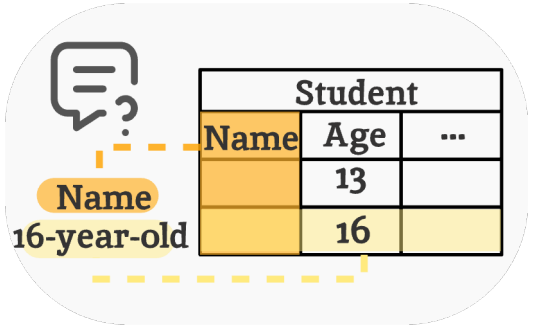
Workflow of Database Content Retrieval



Method Classification of DB Content Retrieval

Method classification：

- String Matching-based Database Content Retrieval
- Neural Network-based Database Content Retrieval
- Index Strategy for Database Content Retrieval

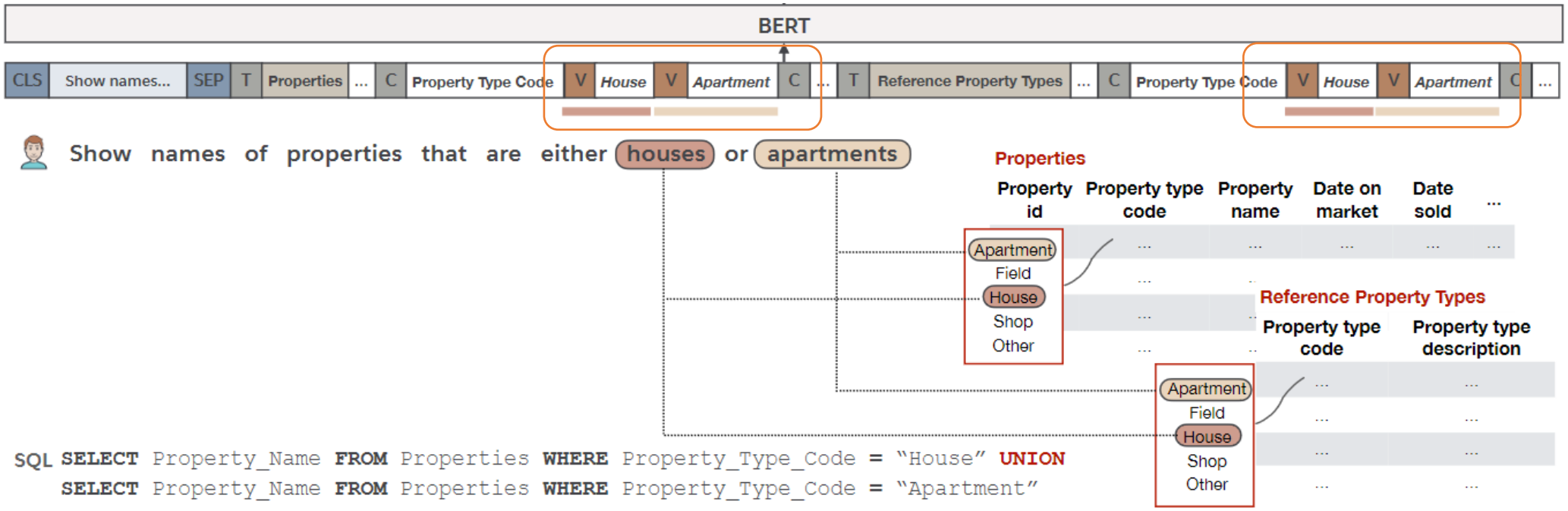


Database Content Retrieval

String Matching-based Database Content Retrieval

BRIDGE designs an anchor text matching to automatically extract cell values mentioned in the NL.

- It uses a heuristic method to calculate the maximum sequence match between the problem and the cell values to determine the matching boundary.



Xi Victoria Lin, Richard Socher, Caiming Xiong:
Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. EMNLP (Findings) 2020: 4870-4888

String Matching-based Database Content Retrieval

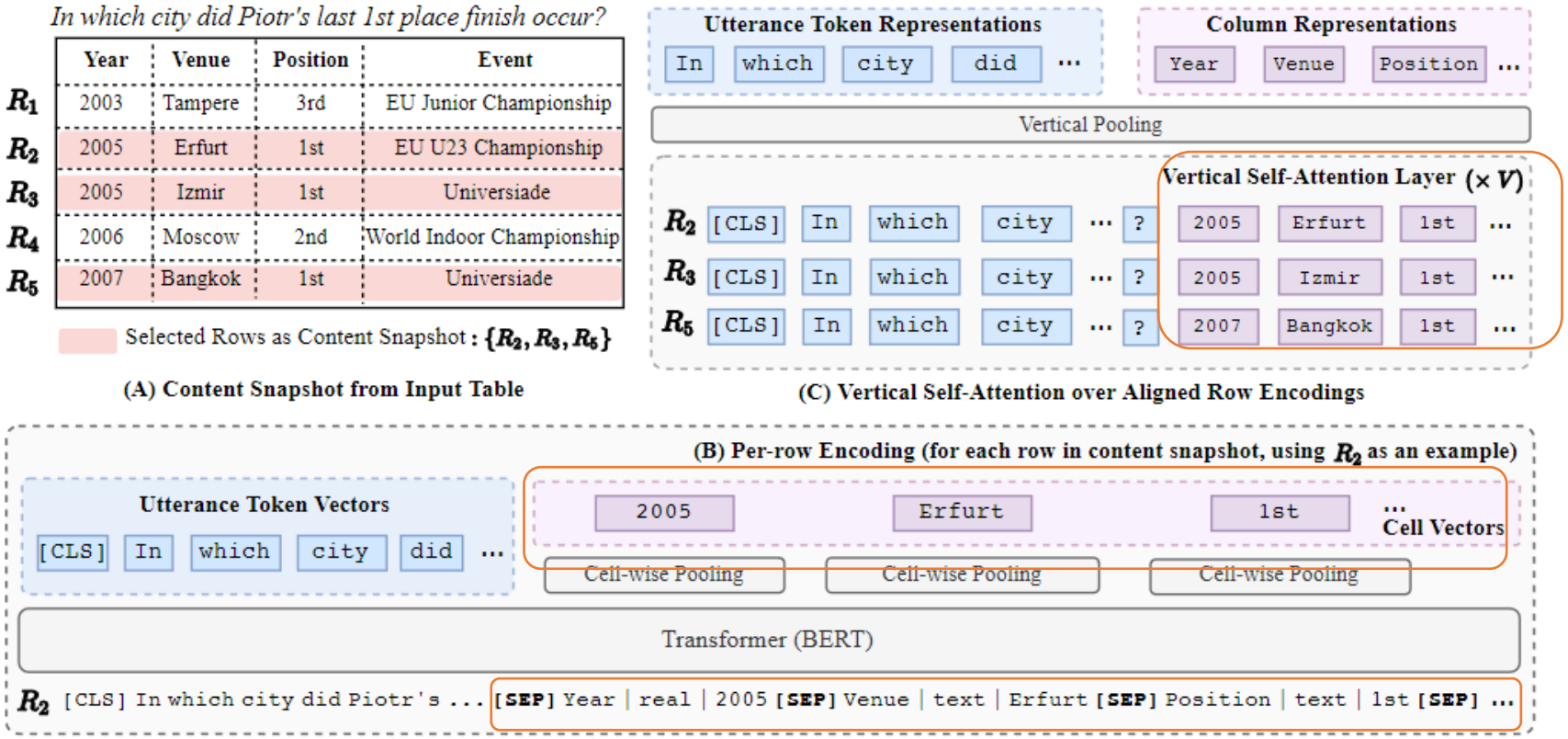
Limitations:

- Struggles with synonyms
- High computational cost for large databases

Neural Network-based Database Content Retrieval

TABERT uses a method called **database content snapshots.**

- It uses an attention mechanism to manage information between cell values across different rows.

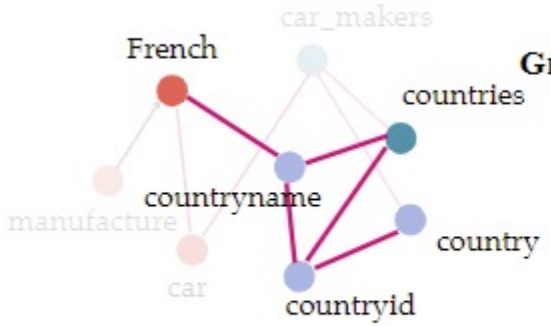


Pengcheng Yin, Graham Neubig, Wen-tau Yih, Sebastian Riedel:
TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. ACL 2020: 8413-8426

Neural Network-based Database Content Retrieval

Graphix-T5 improves structural reasoning capabilities **by modeling the relationship** between cell values and the NL query.

- it uses a value-match relation that defines the relationship between cell values and questions.



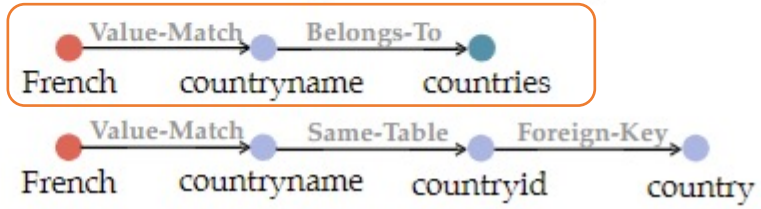
Question: *How many French car manufacturers are there?*

T5-3B: SELECT COUNT(*) FROM **car_makers** WHERE **country** = "France"

Graphix-T5-3B: SELECT COUNT(*) FROM **car_makers** AS T1 JOIN **countries** AS T2 ON T1.**country** = T2.**countryid** WHERE T2.**countryname** = "France"

Gold: SELECT COUNT(*) FROM **car_makers** AS T1 JOIN **countries** AS T2 ON T1.**country** = T2.**countryid** WHERE T2.**countryname** = 'France';

Muti-hop Path



Question Column VALUE-MATCH x is part of the candidate cell values of column y.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, Yongbin Li: Graphix-T5: Mixing Pre-Trained Transformers with Graph-Aware Layers for Text-to-SQL Parsing. CoRR abs/2301.07507 (2023)

Neural Network-based Database Content Retrieval

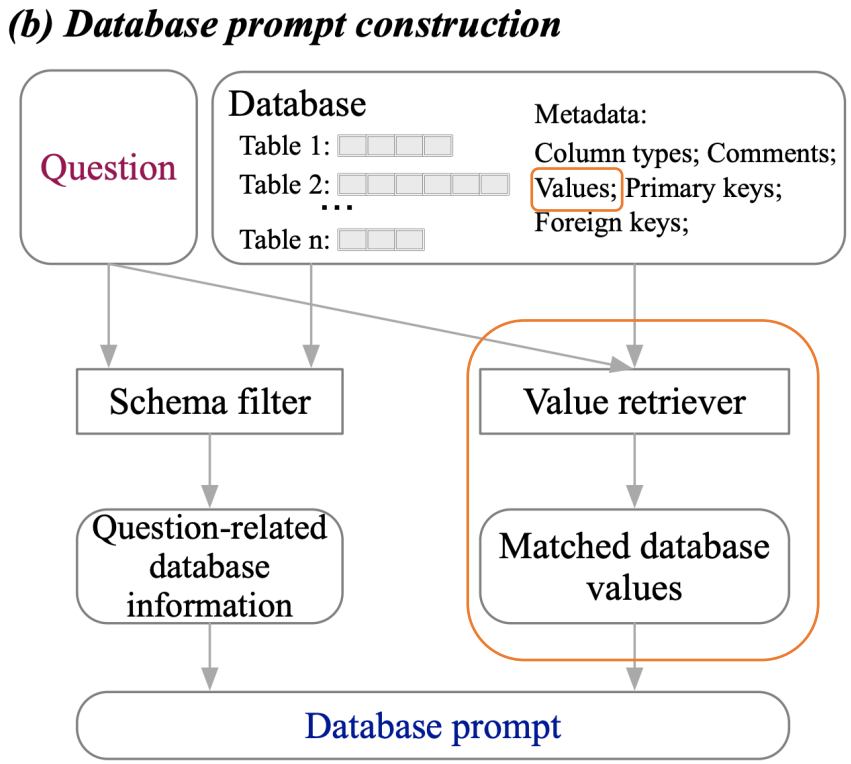
Limitations:

- Lead to the misinterpretation of ambiguous or context-sensitive queries.
- Require significant computing resources.

Index Strategy for Database Content Retrieval

CodeS introduces a **coarse-to-fine cell value matching approach**.

- It builds the index for all values using BM25.
- The index identifies candidate values that are relevant to NL.

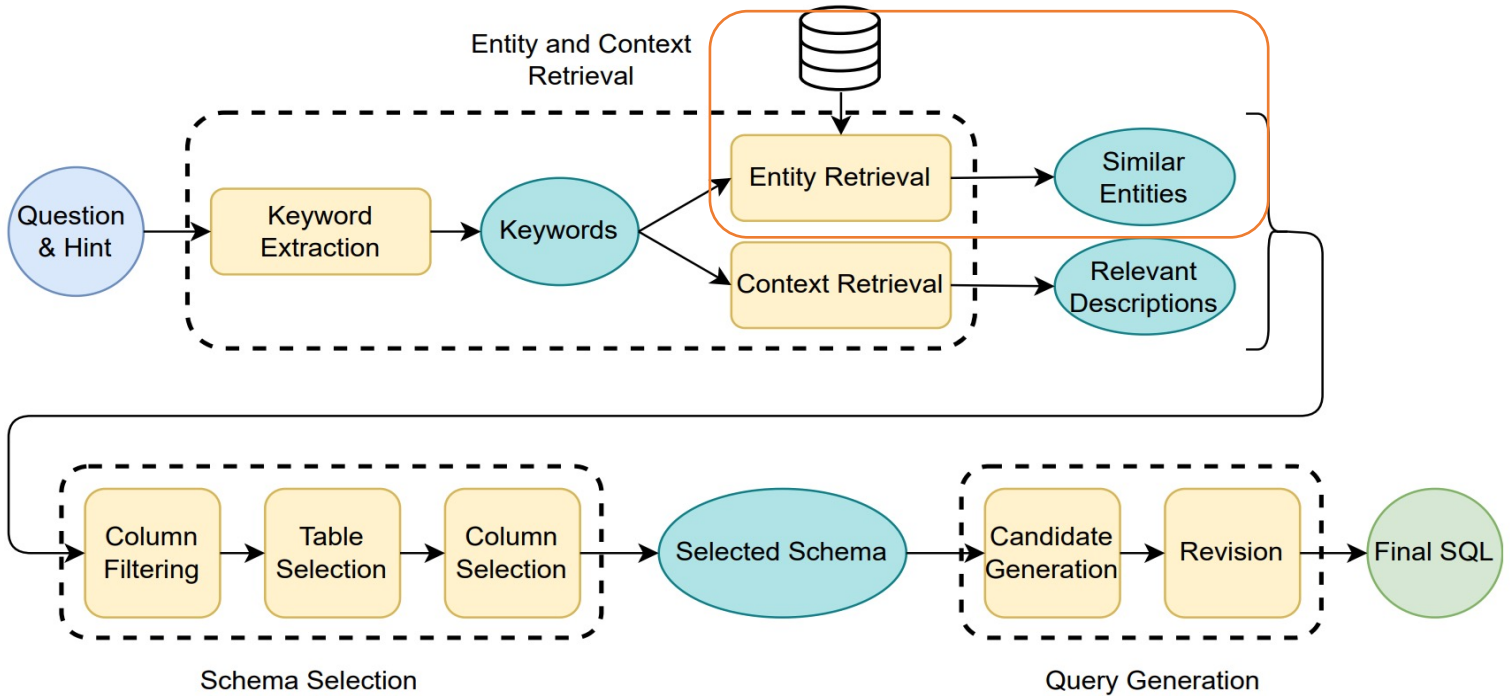


Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, Hong Chen: CodeS: Towards Building Open-source Language Models for Text-to-SQL. CoRR abs/2402.16347 (2024)

Index Strategy for Database Content Retrieval

CHESS uses **a Locality-sensitive Hashing algorithm** for approximate nearest neighbor searches.

- It indexes unique cell values to quickly identify the top similar values related to the NL query.



Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, Amin Saberi: CHESS: Contextual Harnessing for Efficient SQL Synthesis. CoRR abs/2405.16755 (2024)

Index Strategy for Database Content Retrieval

Limitations:

- Time-consuming to build.
- Requires frequent updates for dynamic databases, which increases inference overhead.

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
    - Schema Linking
    - Database Content Retrieval
    - ⭐ Additional Information Acquisition
  - NL2SQL Translation Methods
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Additional Information Acquisition

Motivation:

- Incorporate additional information (e.g., domain knowledge) to improve the comprehension capabilities of NL2SQL models for understanding the NL query, performing the schema linking, and benefiting the NL2SQL translation.

Additional Information Acquisition

Goal:

- To retrieve useful information such as Demonstration examples, Specific domain knowledge, Formulaic Evidence, and Format Information for the NL2SQL backbone model or specific modules.

Method Classification?

Method classification:

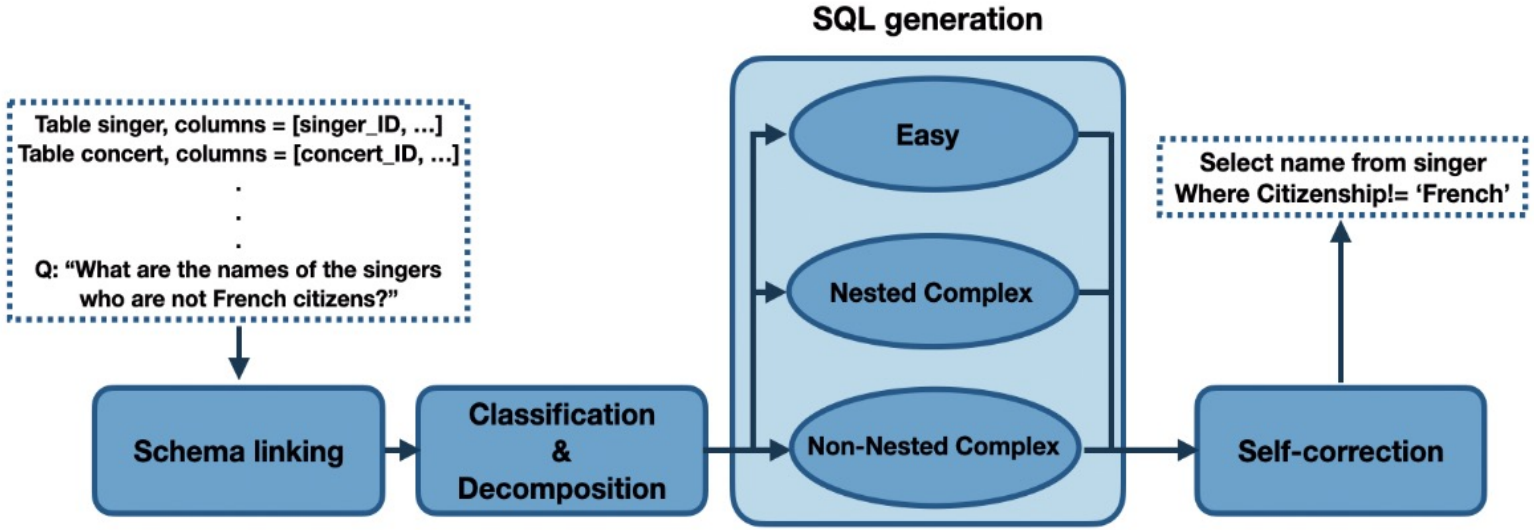
- Sample-based Methods
- Retrieval-based Methods

Additional Information Acquisition

Sample-based Methods

DIN-SQL inserts additional information through few-shot learning across multiple stages of the workflow.

- These stages comprise schema linking, query classification, task decomposition, and self-correction.
- It allows DIN-SQL to effectively tackle various challenges, including the complexity of schema links, identification of multiple table joins, and handling of nested queries.

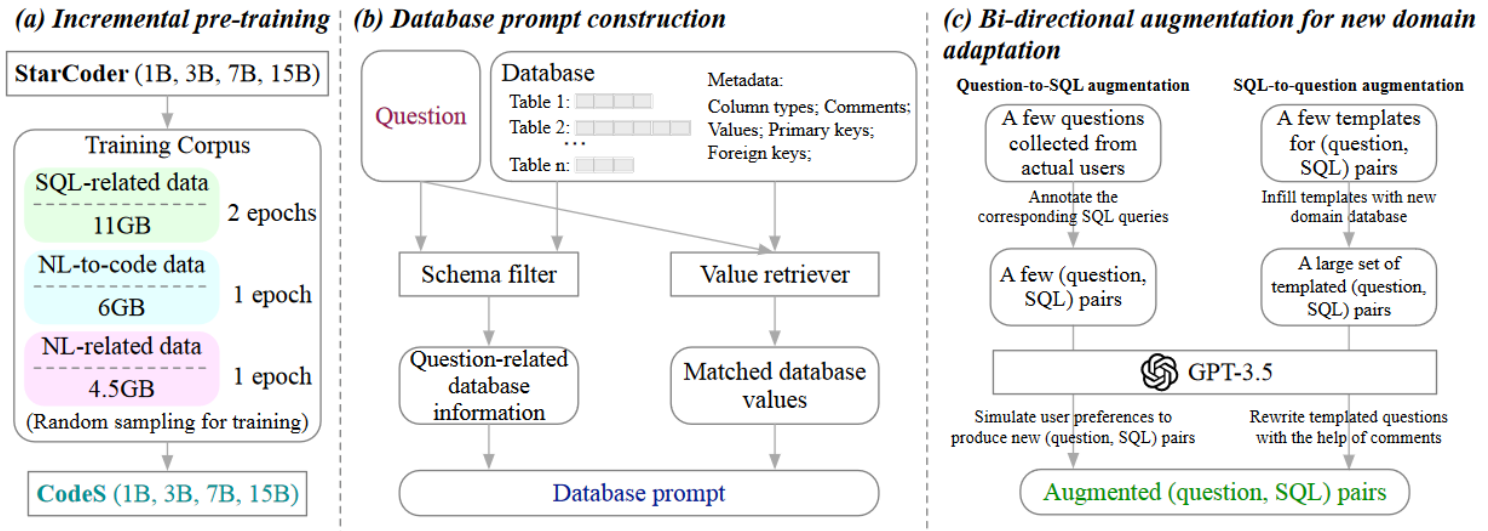


Mohammadreza Pourreza, Davood Rafiei: DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. NeurIPS 2023

Sample-based Methods

CodeS utilizes metadata examples of cross-domain databases as the main additional information, including data types and annotation text, which help the model resolve potential ambiguity issues and understand entity relationships

- This extracted information is transformed into coherent text and concatenated with the question query to form the final input context.



Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, Hong Chen: CodeS: Towards Building Open-source Language Models for Text-to-SQL. Proc. ACM Manag. Data 2(3): 127 (2024)

Sample-based Methods

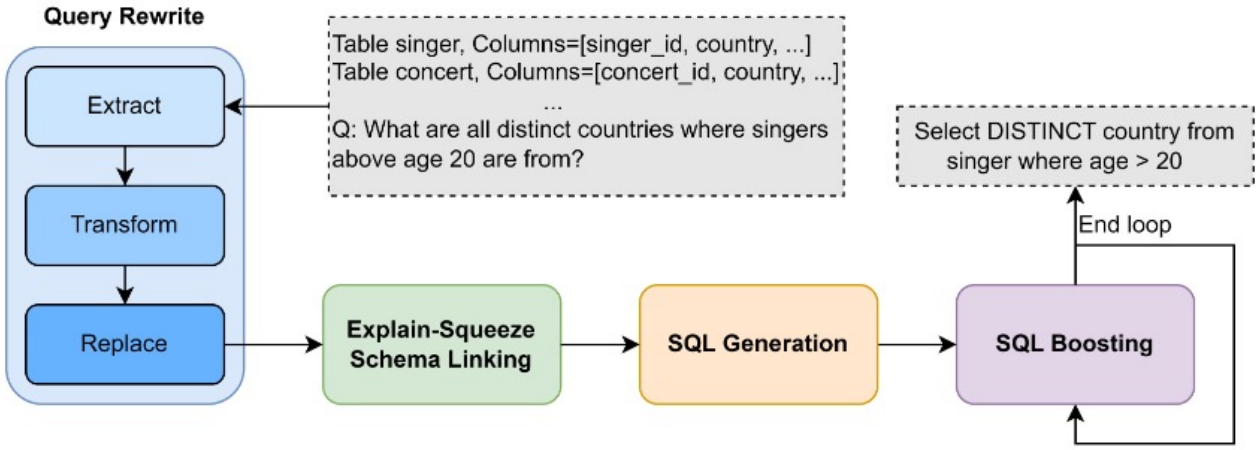
Limitations:

- Increasing computational cost
- Potential hallucination caused by low-quality samples or wrong knowledge

Retrieval-based Methods

ReBoost engages with the LLMs model using the Explain-Squeeze Schema Linking mechanism.

- This mechanism is a two-phase strategy. Initially, it presents a generalized schema to the LLMs to establish a foundational understanding.
- Subsequently, it employs targeted prompting to elicit detailed associations between query phrases and specific database entities, thereby enhancing accuracy in mapping queries to database structures without incurring excessive token cost.

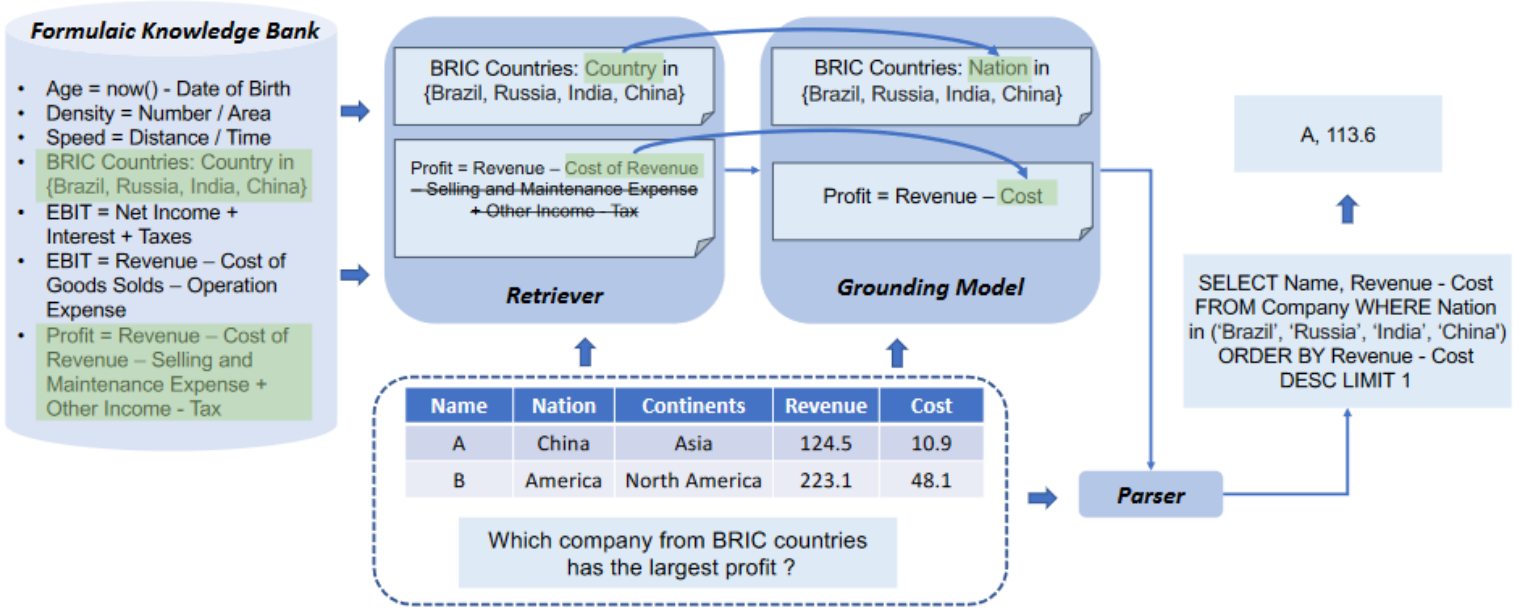


Guanghu Sui, Zhishuai Li, Ziyue Li, Sun Yang, Jingqing Ruan, Hangyu Mao, Rui Zhao: Reboost Large Language Model-based Text-to-SQL, Text-to-Python, and Text-to-Function - with Real Applications in Traffic Domain. CoRR abs/2310.18752 (2023)

Retrieval-based Methods

REGROUP constructs a formulaic knowledge base encompassing various domains, such as finance, real estate, and transportation.

- It leverages a Dense Passage Retriever (DPR) to compute similarity scores for the retrieval results from the formulaic knowledge base.
- Subsequently, an Erasing-Then-Awakening (ETA) model is used to integrate the entities in these formulaic knowledge items with the entities in NL and schema.



Longxu Dou, Yan Gao, Xuqi Liu, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, Min-Yen Kan, Jian-Guang Lou: Towards Knowledge-Intensive Text-to-SQL Semantic Parsing with Formulaic Knowledge. CoRR abs/2301.01067 (2023)

Retrieval-based Methods

Limitations:

- Insufficient exploration of retrieving formulated or structured domain knowledge bases
- Computational cost of retrieving and embedding additional information

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
    - ⭐ Encoding Strategy
    - Decoding Strategy
    - Task-specific Prompt Strategies
    - Intermediate Representation
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Encoding Strategy

Motivation:

- Converts unstructured and semi-structured data into a form that can be processed for generating SQL queries.
- The encoding process captures the semantic meaning of the NL input and the structural information of the database schema, enabling the model to understand and map the user's intent to the corresponding SQL query.

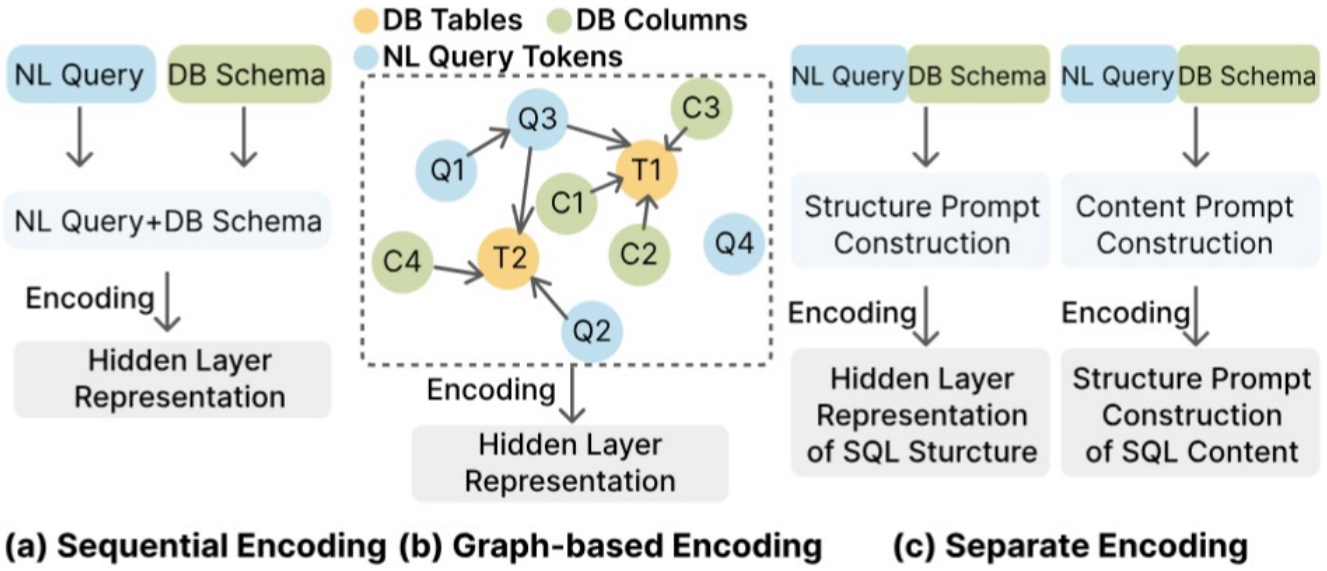
Goal:

- Transform NL and database schema into a structured format that can be effectively utilized by a language model.

Method classification of Encoding Strategy

Method classification:

- Sequential Encoding Strategy
- Graph-based Encoding Strategy
- Separate Encoding Strategy

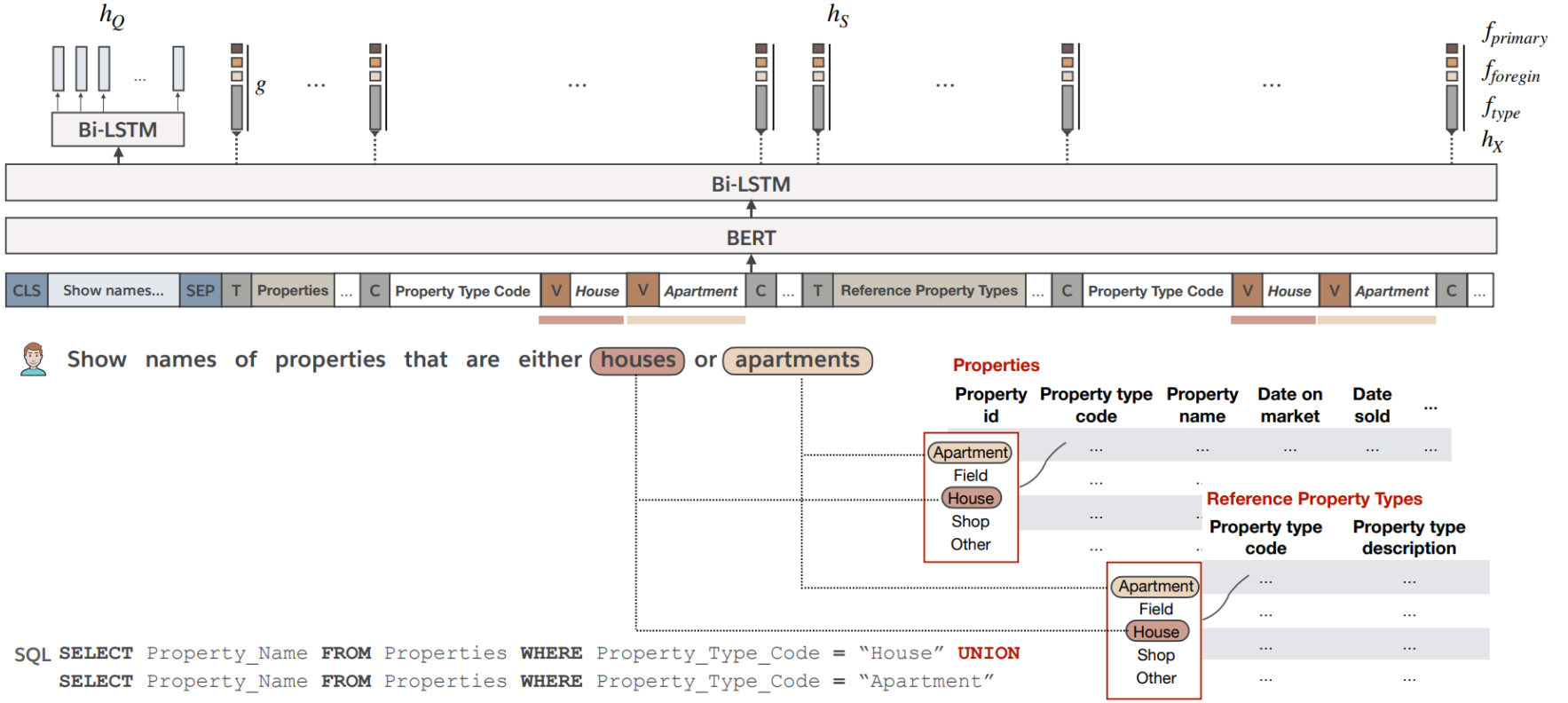


An Overview of the Encoding Strategies

Sequential Encoding Strategy

BRIDGE enhances the alignment between text and the database schema.

- It represents the NL question and database schema as a tagged sequence.
- It inserts matched database cell values (anchor texts) after the corresponding fields.

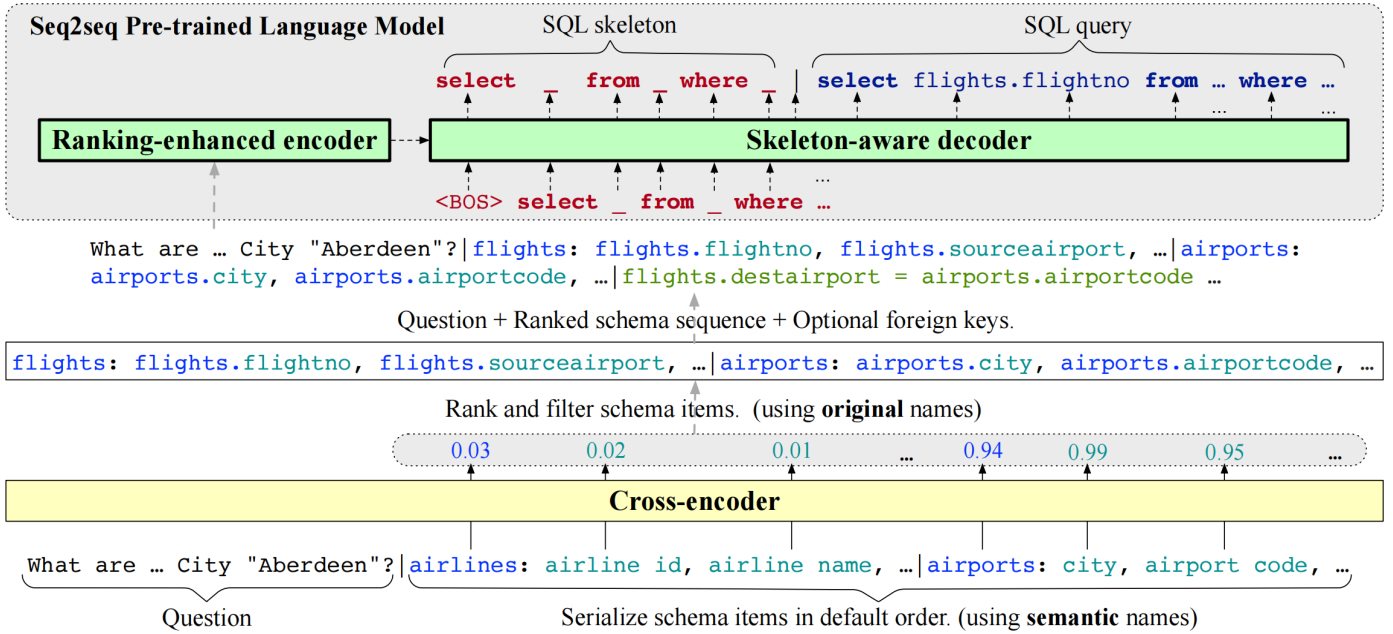


Xi Victoria Lin, Richard Socher, Caiming Xiong:
Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. EMNLP (Findings) 2020: 4870-4888

Sequential Encoding Strategy

RESDSQL uses a ranking-enhanced encoder to sort and filter schema items, thereby reducing the complexity of schema linking during encoding.

- This method ensures that the most relevant schema items are prioritized, improving the overall efficiency of the encoding process.



Haoyang Li, Jing Zhang, Cuiping Li, Hong Chen:
RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. AAAI 2023: 13067-13075

Sequential Encoding Strategy

Limitations:

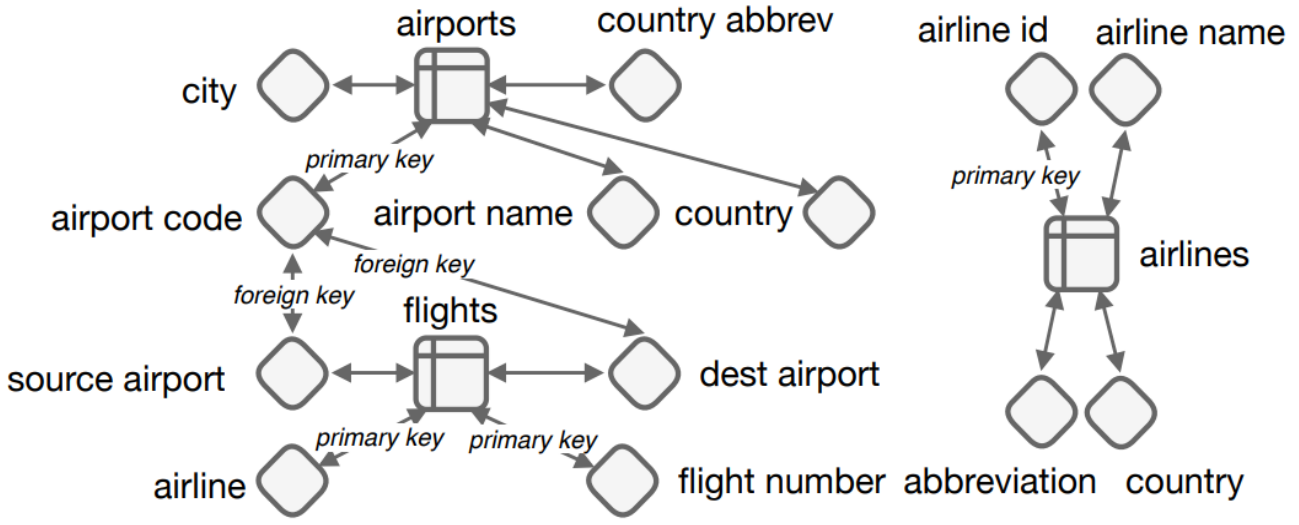
- Its linear approach, which treats all inputs as sequences of tokens, may fail to fully capture the complex relationships between the database schema and NL query.
- This can affect the model's ability to understand and generate complex queries.

Graph-based Encoding Strategy

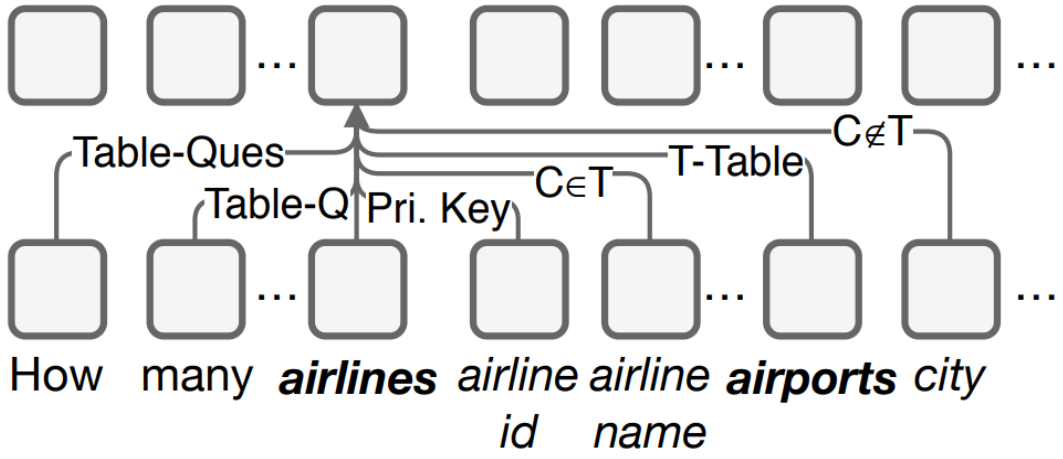
RAT-SQL introduces a relation-aware self-attention mechanism.

- It allows the model to explicitly consider and utilize predefined relational information when jointly encoding the question and the database schema.
- These relationships are represented as a graph structure, and it can more effectively capture the structural information in the schema and its alignment with the NL query.

An illustration of an example schema as a graph G



One RAT layer in the schema encoder.

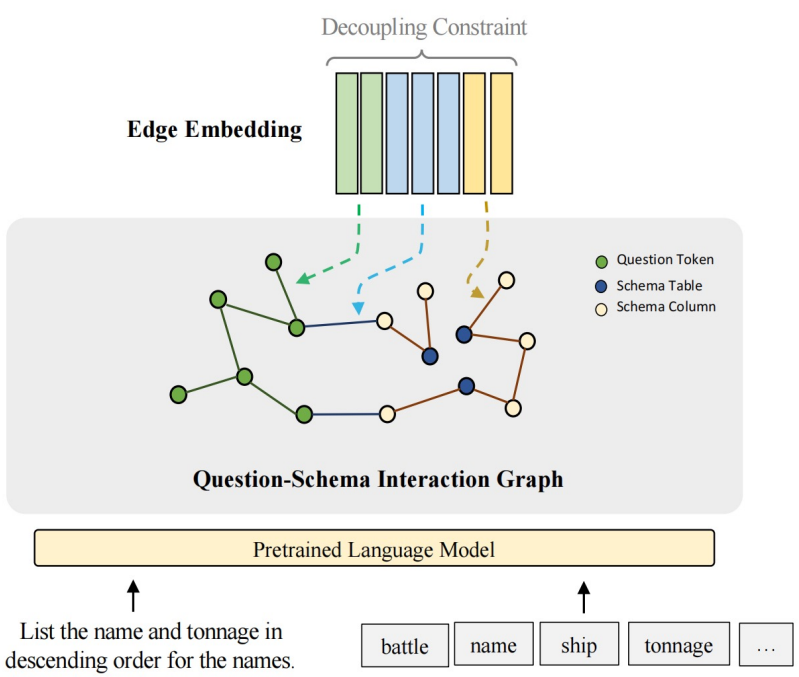


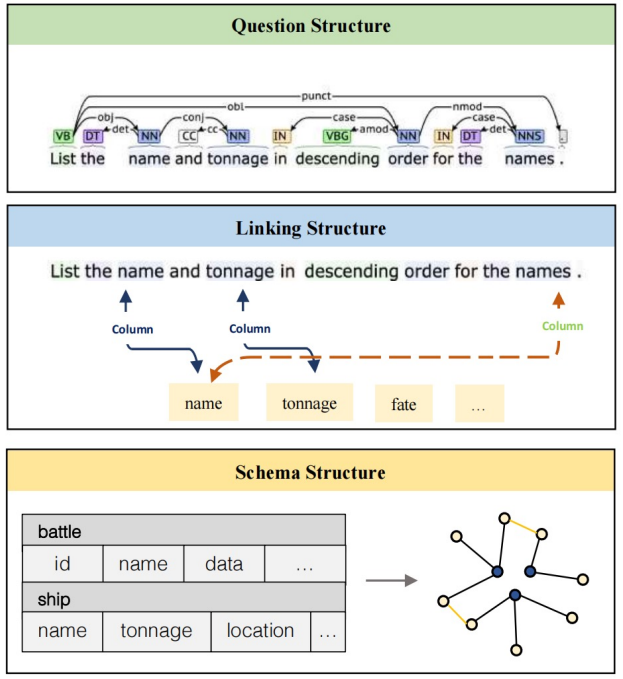
Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, Matthew Richardson:
RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. ACL 2020: 7567-7578

Graph-based Encoding Strategy

S²SQL uses ELECTRA as its pre-trained language model.

- It enhances encoding by injecting syntactic structure information at the encoding stage, improving the semantic understanding and generation of models.





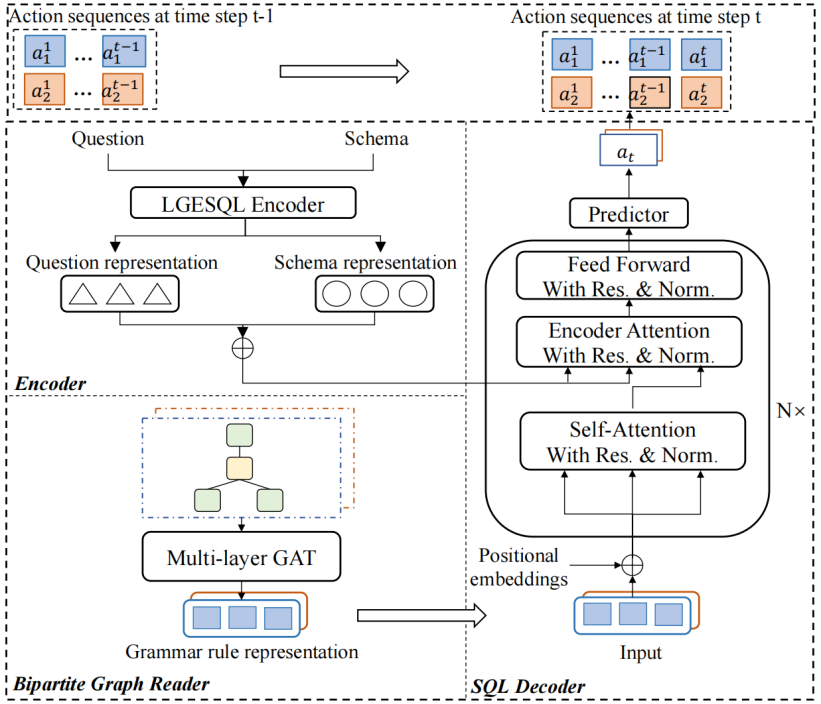
An overview of S²SQL framework.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, Yongbin Li:
S²SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers. ACL (Findings) 2022: 1254-1262

Graph-based Encoding Strategy

G³R introduces the LGESQL encoder.

- It captures and integrates multisource heterogeneous information by constructing and utilizing a heterogeneous graph and a Graph Attention Network (GAT) , thereby enhancing the representation capability for NL and database schema.



An overview of the architecture of G³R framework.

Yanzheng Xiang, Qian-Wen Zhang, Xu Zhang, Zejie Liu, Yunbo Cao, Deyu Zhou:
G³R: A Graph-Guided Generate-and-Rerank Framework for Complex and Cross-domain Text-to-SQL Generation. ACL (Findings) 2023: 338-352

Graph-based Encoding Strategy

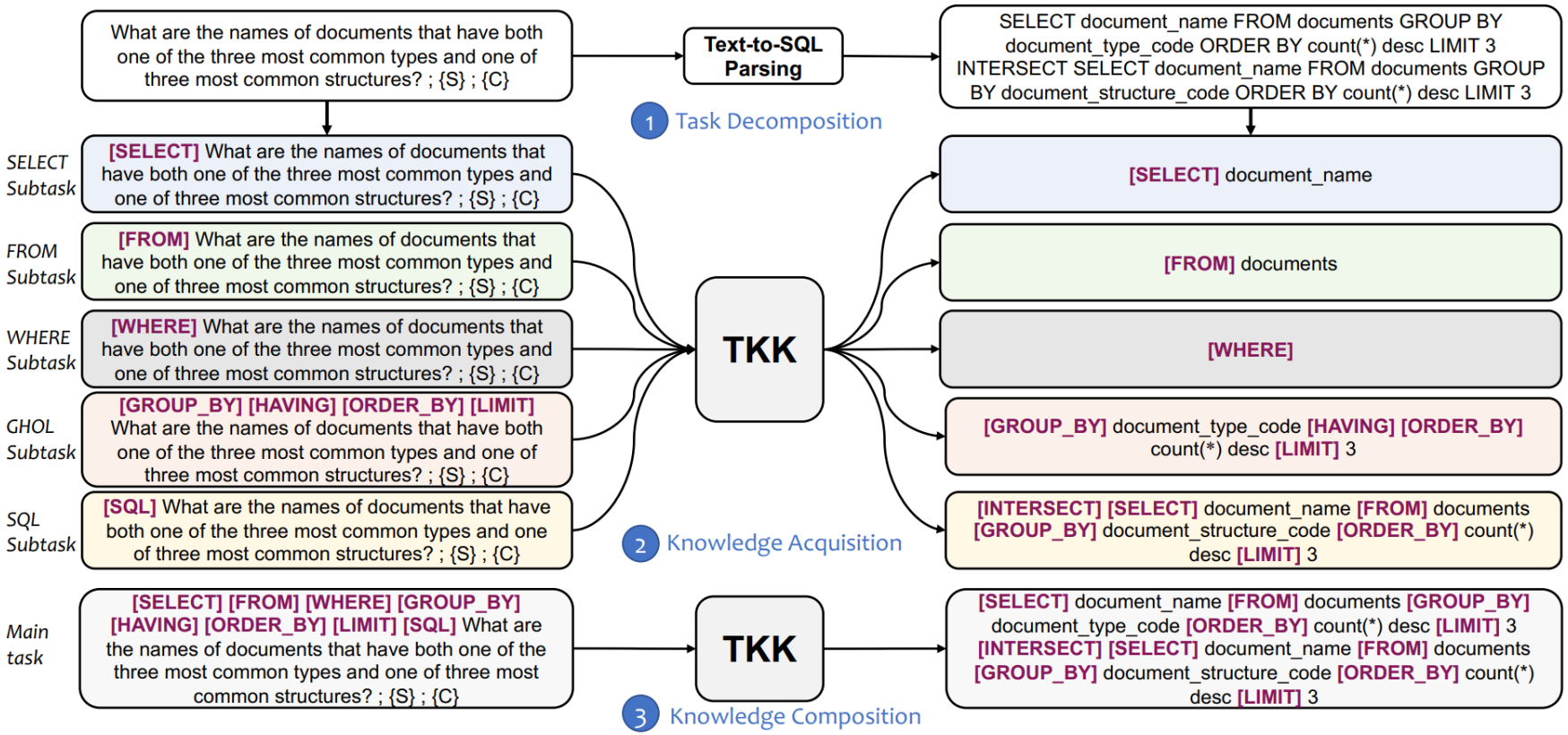
Limitation:

- Require more sophisticated algorithms for constructing and processing graph structures.
- Need a large amount of training data to fully leverage its structural advantages.

Separate Encoding Strategy

TKK employs task decomposition and multi-task learning strategies in encoding.

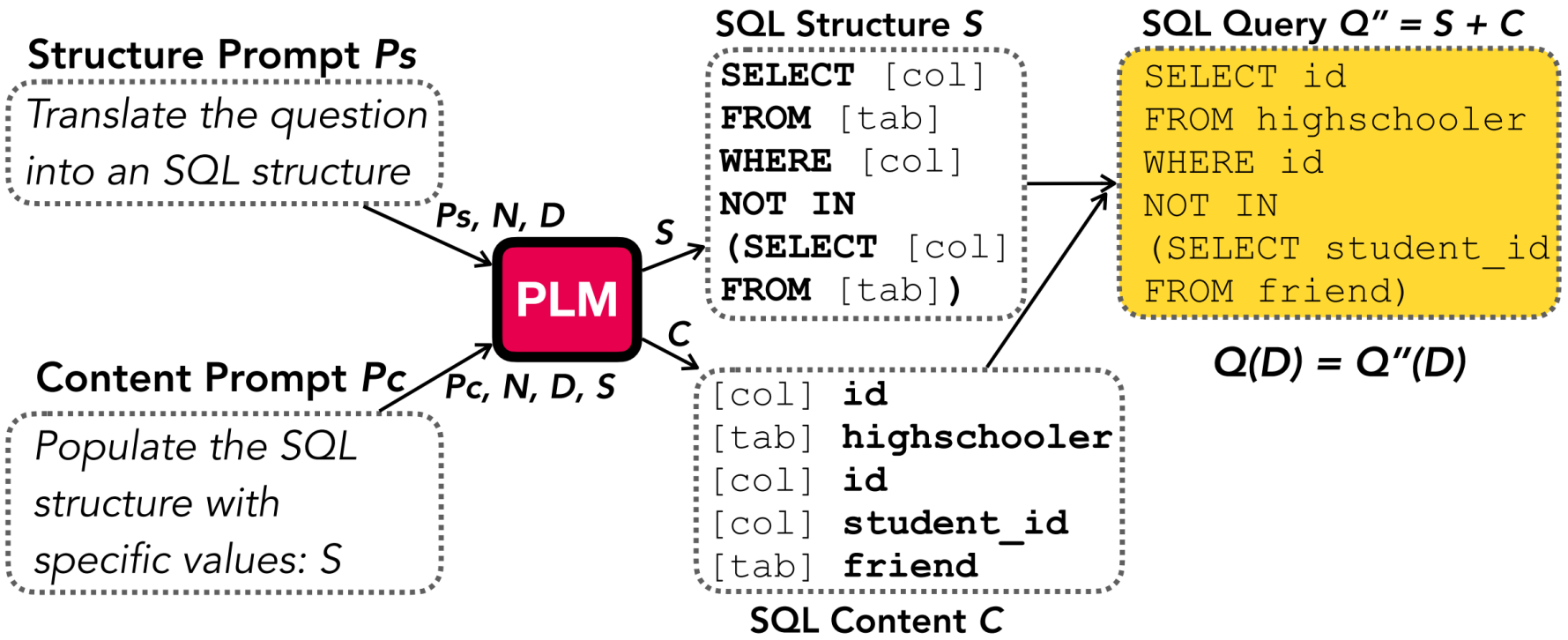
- It breaks down the complex NL2SQL task into multiple subtasks.
- In this way, it can progressively acquire and combine knowledge.



Chang Gao, Bowen Li, Wenxuan Zhang, Wai Lam, Binhua Li, Fei Huang, Luo Si, Yongbin Li: Towards Generalizable and Robust Text-to-SQL Parsing. EMNLP (Findings) 2022: 2113-2125

Separate Encoding Strategy

SC-Prompt divides text encoding into two stages: the structure stage and the content stage, with each stage being encoded separately.



SC-prompt framework with two-stage prompt learning

Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, Xiaoyong Du:
Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning. Proc. ACM Manag. Data 1(2): 147:1-147:28 (2023)

Separate Encoding Strategy

Limitations:

- Require multiple processing of input data
- May extend the training and inference time of the model

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
    - Encoding Strategy
    - ⭐ Decoding Strategy
    - Task-specific Prompt Strategies
    - Intermediate Representation
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Decoding Strategy

Motivation:

- The choice of decoding strategy directly affects the quality and performance of the generated SQL queries.
- An excellent decoding strategy not only produces syntactically correct SQL queries but also ensures that the semantics of the SQL queries align with the NL and can even optimize the execution efficiency of the queries.

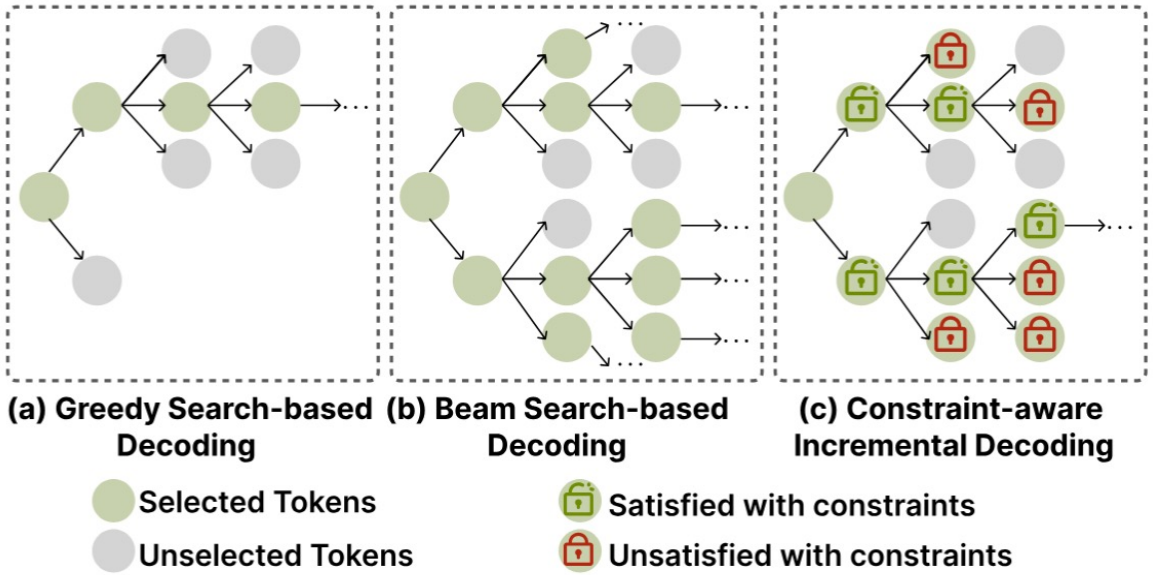
Goal:

- Convert the representations generated by the encoder into the target SQL queries.

Method classification of Decoding Strategy

Method classification:

- Greedy search-based decoding strategy
- Beam search-based decoding strategy
- Constraint-aware incremental decoding strategy

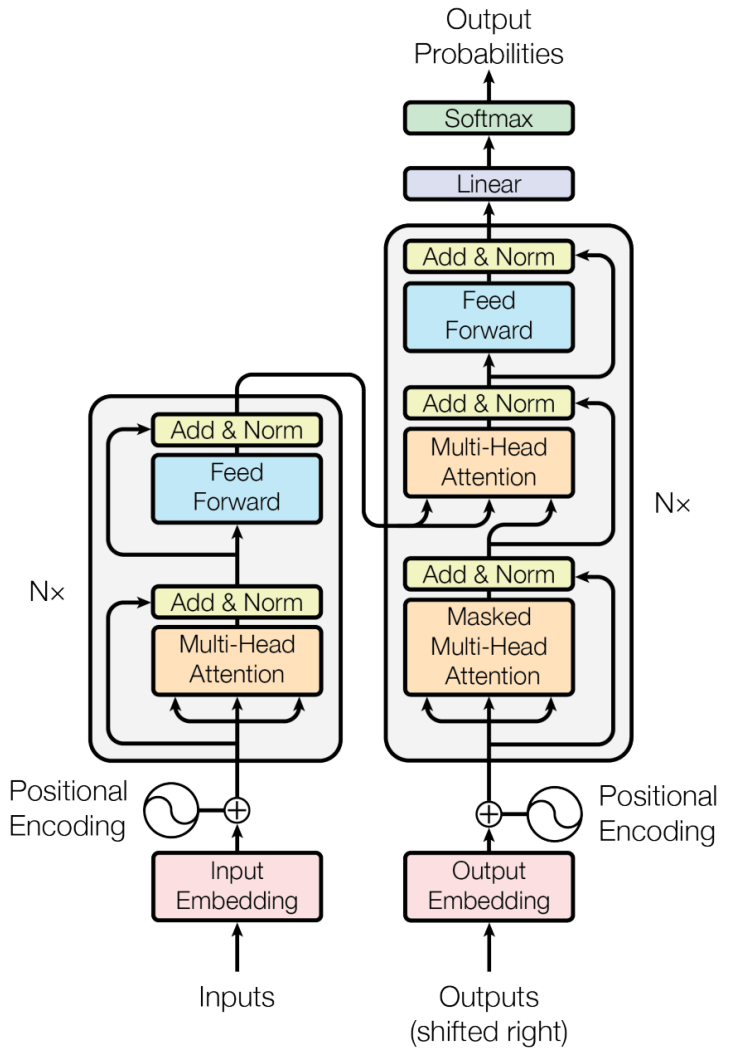


An Overview of the Decoding Strategies

Greedy Search-based Decoding Strategy

The greedy search-based decoding strategy is a simple and fast approach for decoding.

- At each decoding step, greedy search selects the token with the highest current probability as the output.
- This strategy builds the final output sequence by continuously choosing the locally optimal solution.
- Since the default decoding strategy of GPT series models (e.g., GPT-4) is greedy search-based decoding, NL2SQL solutions based on GPT fall into this category.



The Transformer-Model Architecture

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: Attention is All you Need. NIPS 2017: 5998-6008

Greedy Search-based Decoding Strategy

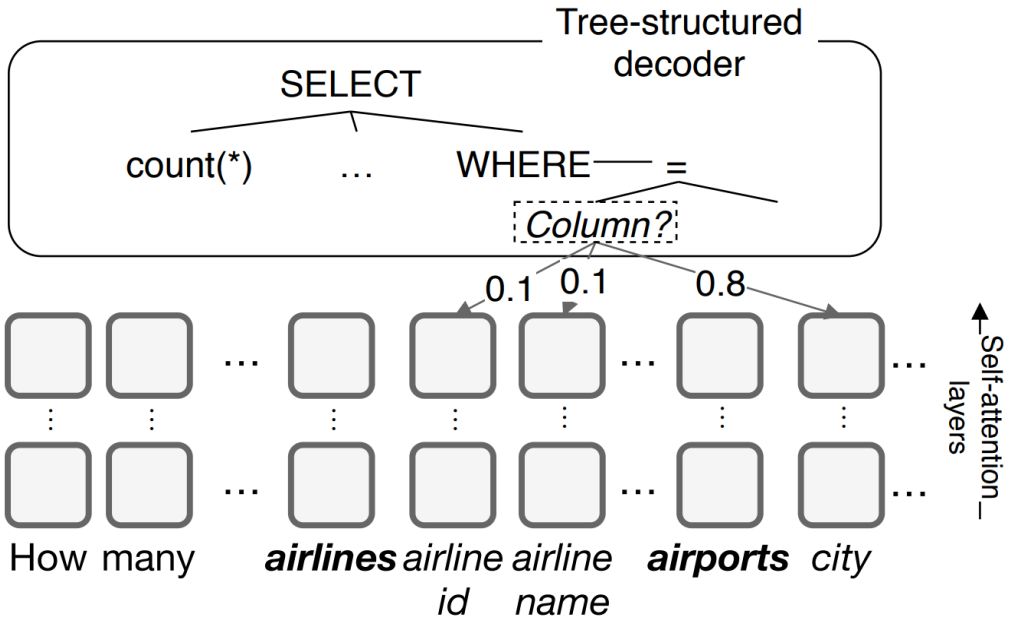
Limitations:

- Greedy search only considers the optimal solution at the current step, ignoring long-term dependencies and overall optimal solutions.
- May result in the generation of SQL queries that are not globally optimal.
- Errors at each step could lead to biases in decision-making in subsequent steps, potentially accumulating errors throughout the decoding process, especially when handling complex queries.

Beam Search-based Decoding Strategy

RAT-SQL combines relation-aware graph structure encoding and generation techniques.

- During the decoding process, RAT-SQL uses beam search to generate multiple candidate SQL queries
- These queries are then reranked, and the optimal query is selected based on graph structure information.

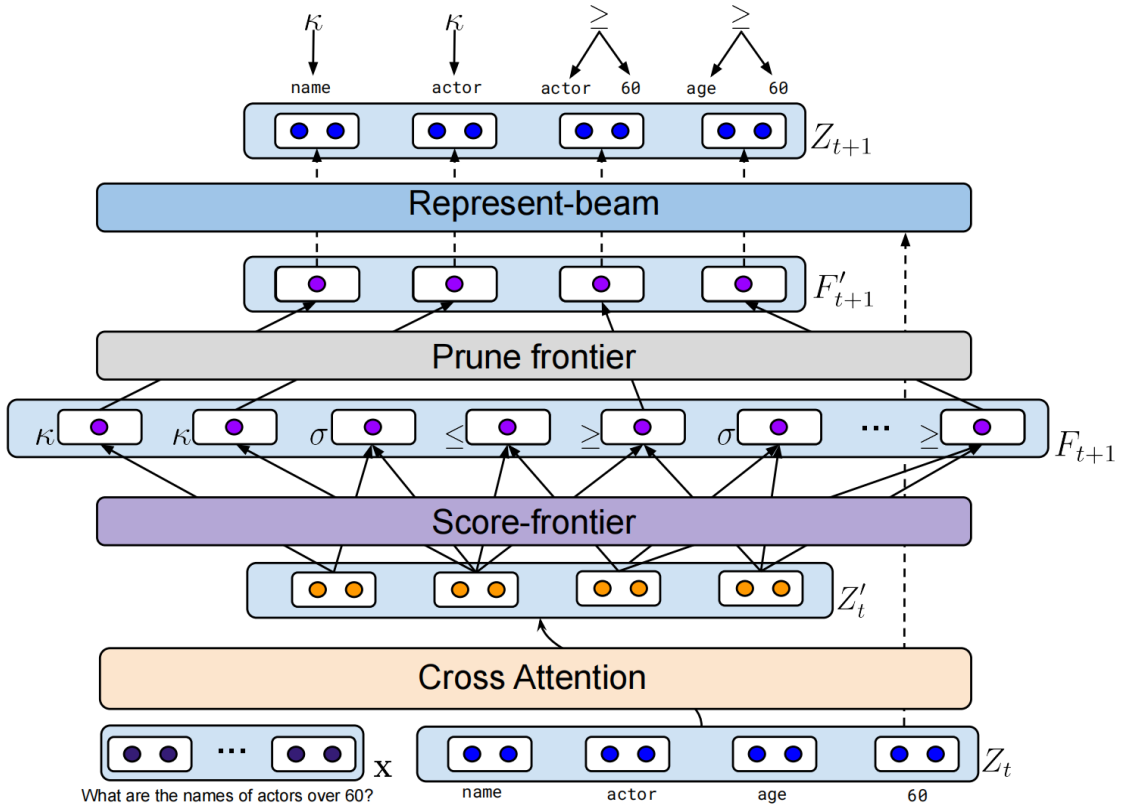


Choosing a column in a tree decoder.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, Matthew Richardson:
RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. ACL 2020: 7567-7578

Beam Search-based Decoding Strategy

SmBoP uses a semi-autoregressive bottom-up decoding approach, improving decoding efficiency by parallelizing the construction and scoring of sub-trees, achieving logarithmic time complexity.



An overview of the decoding procedure of SMBOP.

Ohad Rubin, Jonathan Berant:
SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. SPNLP@ACL-IJCNLP 2021: 12-21

Beam Search-based Decoding Strategy

Limitations:

- More data needs to be processed at each decoding step.
- This significantly increases the demand for memory and computational resources and results in slower decoding speeds than greedy search.

Constraint-aware Incremental Decoding Strategy

PICARD (Parsing Incrementally for Constrained Auto-Regressive Decoding) introduces the constraint-aware incremental decoding strategy.

- The constraint-aware incremental decoding strategy is specifically designed for NL2SQL tasks.
- This strategy aims to ensure the generation of syntactically correct SQL queries by incorporating constraints during the decoding process.

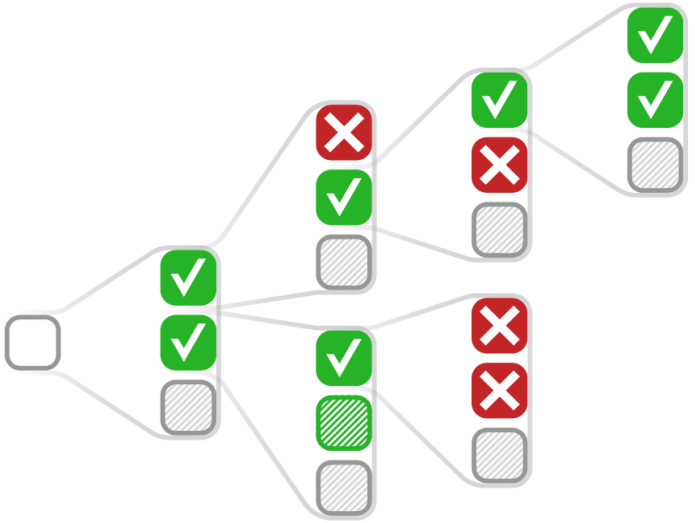


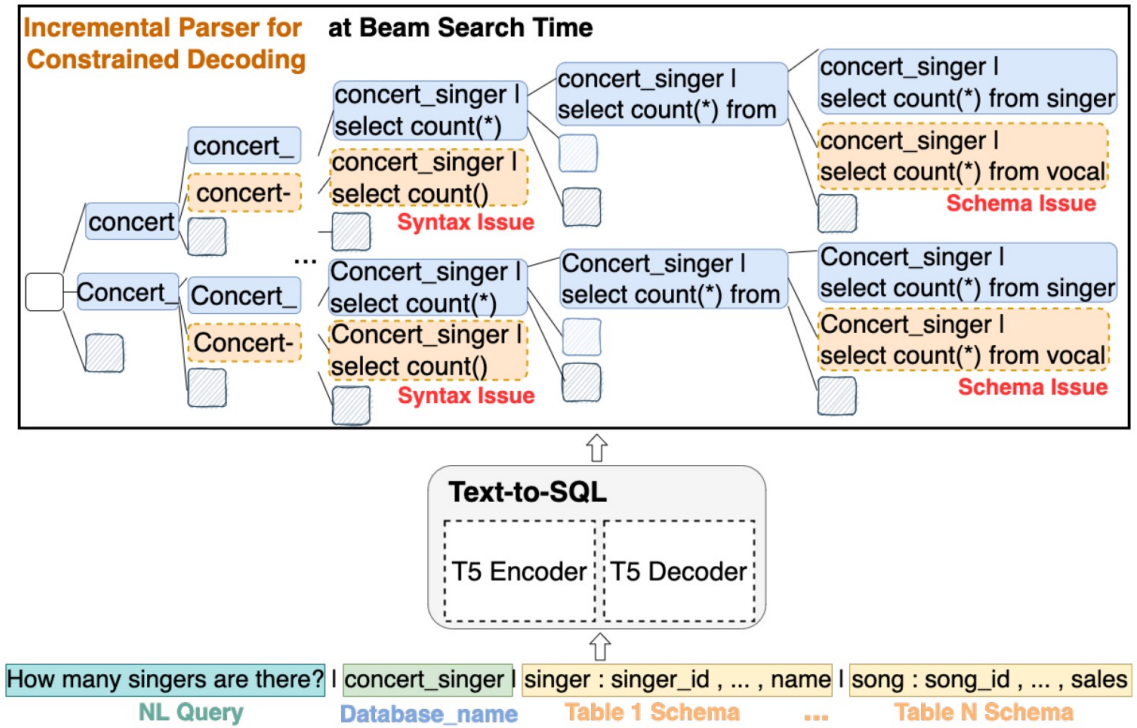
Illustration of constrained beam search with beam size 2 and PICARD.

Torsten Scholak, Nathan Schucher, Dzmitry Bahdanau:
PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. EMNLP (1) 2021: 9895-9901

Constraint-aware Incremental Decoding Strategy

Due to its ability to improve the accuracy of SQL generation, PICARD has been adopted as a decoding strategy by several models, such as **N-best List Rerankers**.

- Compared to versions without PICARD, these models have shown improved accuracy.



PICARD explained by an example. The prediction pattern is "⟨Database name⟩ | ⟨pred SQL⟩".

Lu Zeng, Sree Hari Krishnan Parthasarathi, Dilek Hakkani-Tur:
N-Best Hypotheses Reranking for Text-to-SQL Systems. SLT 2022: 663-670

Constraint-aware Incremental Decoding Strategy

Limitations:

- Incremental decoding and progressively adding constraints may require more computational resources and processing time.

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
    - Encoding Strategy
    - Decoding Strategy
    - ⭐ Task-specific Prompt Strategies
    - Intermediate Representation
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Task-Specific Prompt Strategy

Motivation:

- Instruct the LLMs to optimize the SQL query generation process according to task-specific rules.

Goal:

- Improve the accuracy of translating complex semantic NL query into the corresponding SQL query.

Task-Specific Prompt Strategy

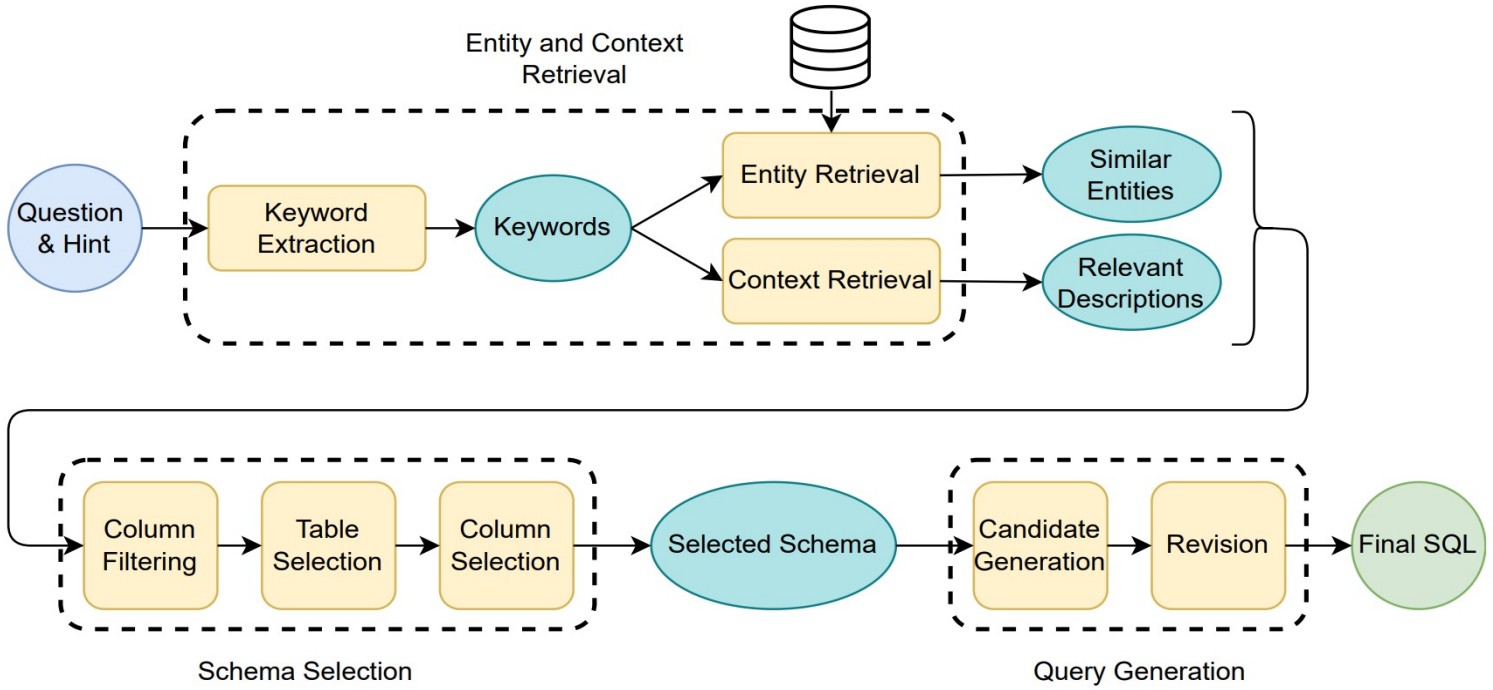
Method classification：

- Chain-of-Thought
- Task Decomposition

Chain-of-Thought

CHESS transforms NL into SQL statements using a streamlined pipeline that relies on LLMs and CoT.

- This CoT process comprises entity and context retrieval, schema selection, SQL generation, and revision.



Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, Dongmei Zhang: Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. ACL (1) 2019: 4524-4535

Chain-of-Thought

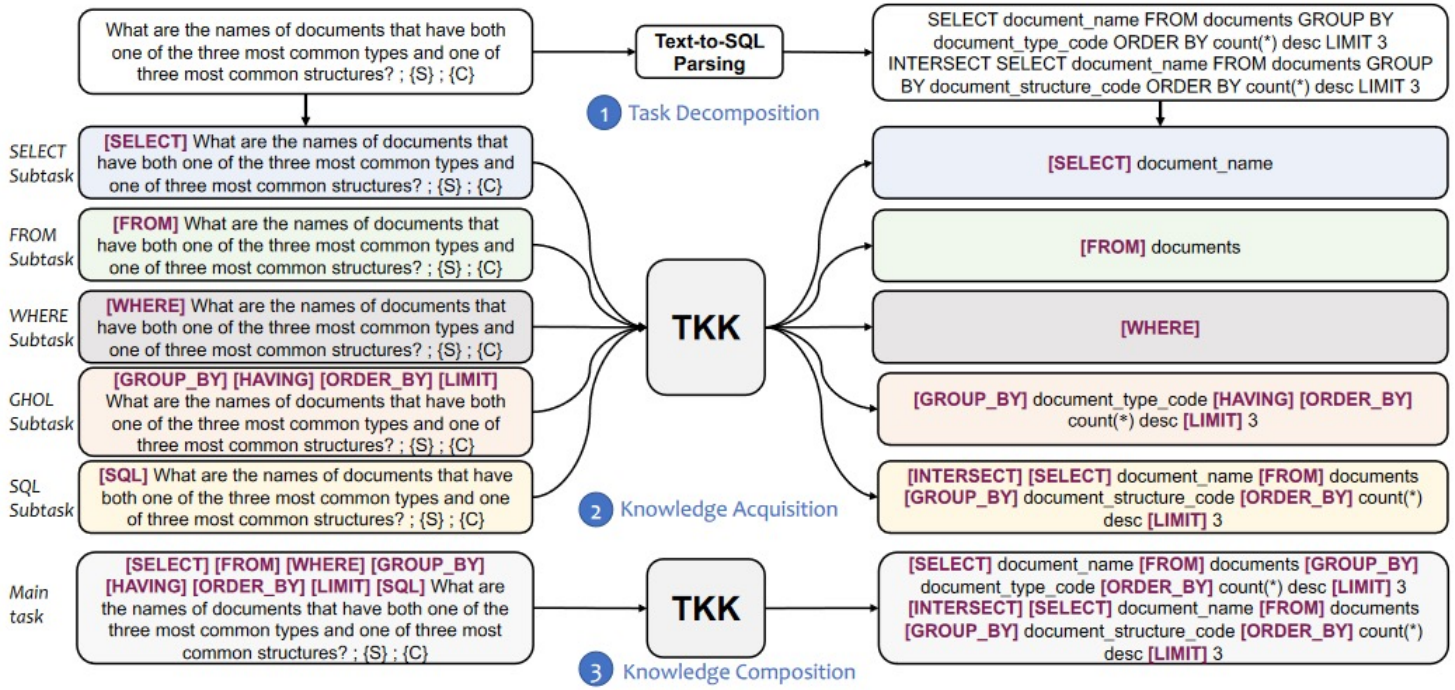
Limitations:

- Hallucination caused by unknown domain knowledge
- Hard to control the output of key components

Decomposition

TKK divides the initial NL2SQL parsing tasks into various small individual subtasks, with each corresponding to the mapping of the NL query to one or more clauses of the SQL query.

- This decomposition approach allows the model to concentrate on learning the generation of each clause, thereby compelling the model to understand the problem, the database schema, and the alignment between each SQL clause.

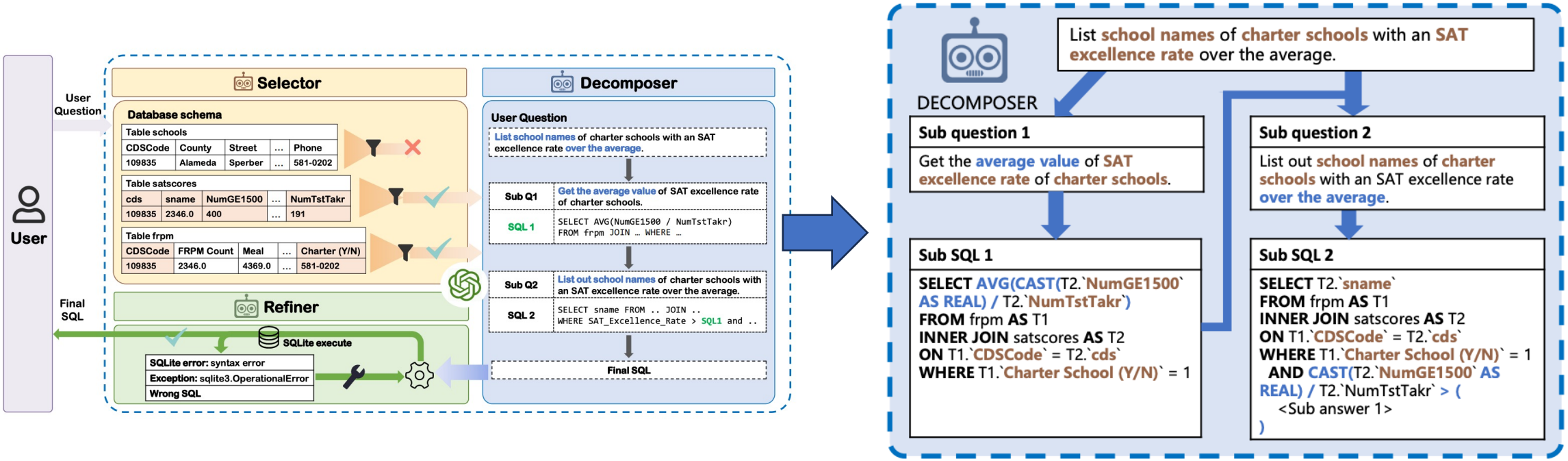


Chang Gao, Bowen Li, Wenxuan Zhang, Wai Lam, Binhua Li, Fei Huang, Luo Si, Yongbin Li: Towards Generalizable and Robust Text-to-SQL Parsing. EMNLP (Findings) 2022: 2113-2125

Decomposition

MAC-SQL incorporates a ***Decomposer*** agent designed to break down the user's original problem into several subproblems.

- This decomposition process aims to lessen the complexity of the origin question, enabling the generation of simpler SQL queries to solve each individual subproblem.

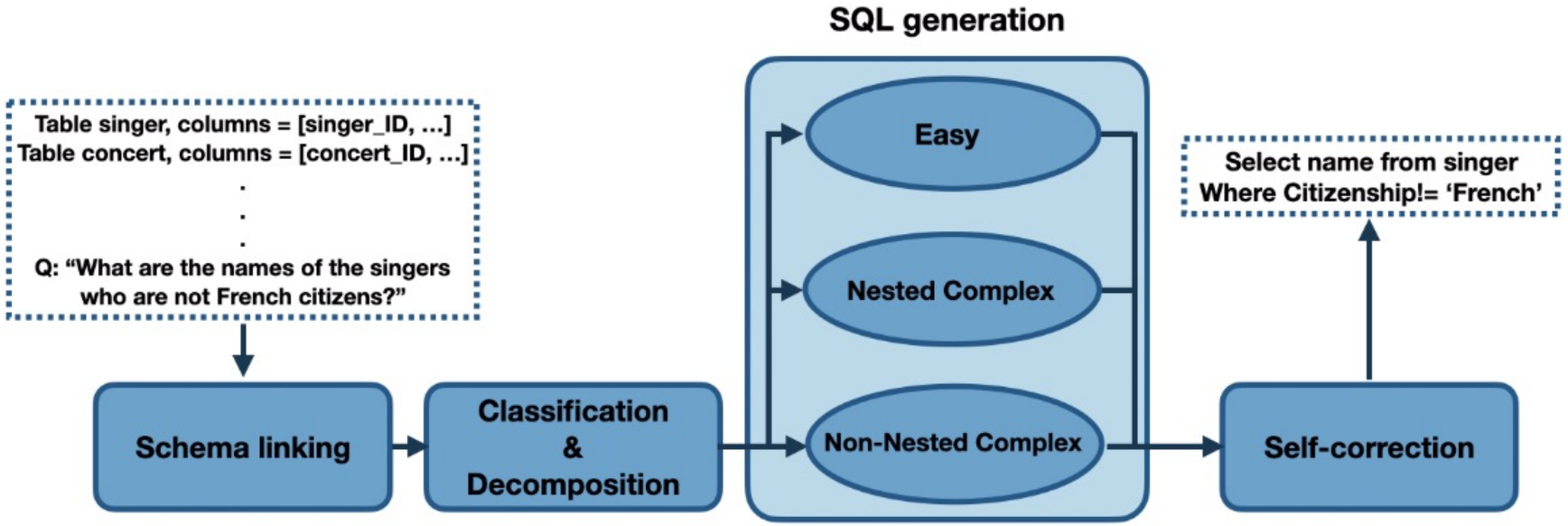


Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, Zhoujun Li: MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. CoRR abs/2312.11242 (2023)

Decomposition

DIN-SQL employs a sophisticated categorization module for decomposition.

- It classifies queries into distinct complexity groups: Easy, Non-Nested Complex, and Nested Complex, with the reference of NL and database schema.
- By strategically identifying and separating schema linking, join conditions, and nested structures, the module facilitates a structured generation of SQL queries and amplifies the accuracy of translating complex the NL query into executable SQL.



Mohammadreza Pourreza, Davood Rafiei: DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. NeurIPS 2023

Decomposition

Limitations:

- Increasing computational cost
- Lowing efficiency because the model need to finish more sub-tasks

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
    - Encoding Strategy
    - Decoding Strategy
    - Task-specific Prompt Strategies
    - ⭐ Intermediate Representation
  - Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Intermediate Representation

Motivation:

- Design a grammar-free intermediate representation compared to SQL as the bridge between the "free-form" NL query and the "constrained and formal" SQL query.

Goal:

- Capture the essential components and relationships of an NL query without the strict syntax rules of SQL.

NL Query:

Which film has more than 5 actors and less than 3 in the inventory?

Intermediate Representation:

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

```
SELECT film.title
WHERE count(film_actor.*)>5 And count(inventory.*)<3
```

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

```
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
INTERSECT
(SELECT [column]                [column] title
FROM [table]                    [table] film
JOIN [table]                    [table] inventory
ON [table].[column]             [table].[column] film.film_id
=[table].[column]               [table].[column] inventory.film_id
GROUP BY [column]               [column] film_id
HAVING count([column]) < n)     [column] * [n] 3
```

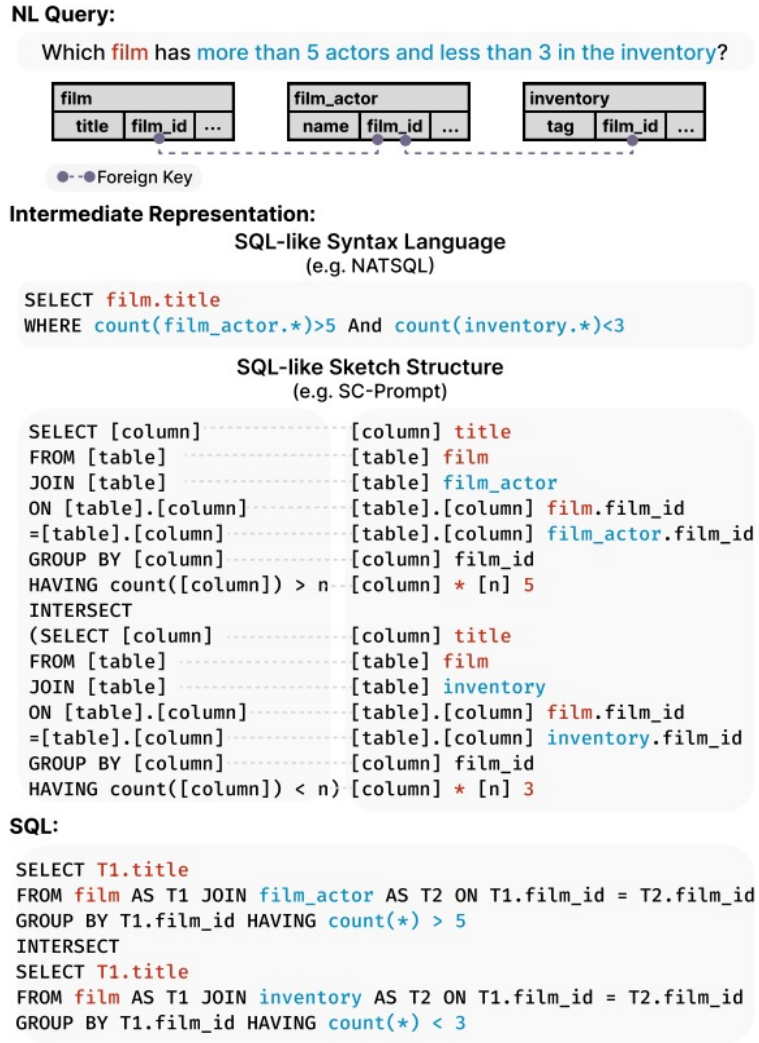
SQL:

```
SELECT T1.title
FROM film AS T1 JOIN film_actor AS T2 ON T1.film_id = T2.film_id
GROUP BY T1.film_id HAVING count(*) > 5
INTERSECT
SELECT T1.title
FROM film AS T1 JOIN inventory AS T2 ON T1.film_id = T2.film_id
GROUP BY T1.film_id HAVING count(*) < 3
```

Method Classification of Intermediate Representation

Method classification：

- SQL-Like Syntax Language
- SQL-Like Sketch Structure



Intermediate Representation

NatSQL is a widely recognized SQL-like syntax language that eliminates SQL statement operators, keywords, set operators, and other elements seldom found in user problem descriptions.

- It enhances schema linking by minimizing the necessary number of schema items.

Question :
Which film has more than 5 actors and less than 3 in the inventory?

SQL :
```
SELECT T1.title FROM film AS T1 JOIN film_actor AS T2 ON T1.film_id = T2.film_id GROUP
BY T1.film_id HAVING count(*) > 5 INTERSECT SELECT T1.title FROM film AS T1 JOIN
inventory AS T2 ON T1.film_id = T2.film_id GROUP BY T1.film_id HAVING count(*) < 3
```

The IR of RAT-SQL : (Remove the JOIN ON Clause)
```
SELECT title FROM film, film_actor GROUP BY film_id HAVING count(*) > 5
INTERSECT SELECT title FROM film, inventory GROUP BY film_id HAVING count(*) < 3
```

The IR of SyntaxSQL : (Remove the JOIN ON and FROM Clause)
```
SELECT film.title GROUP BY film.film_id HAVING count(*) > 5 INTERSECT
SELECT film.title GROUP BY film.film_id HAVING count(*) < 3
```

SemQL : (Remove the JOIN ON, FROM and GROUP BY Clause. Merge the HAVING and WHERE clause)
```
SELECT film.title WHERE count(film_actor.*) > 5  INTERSECT
SELECT film.title WHERE count(inventory.*) < 3
```

NatSQL : (Further remove the set operators based on SemQL)
```
SELECT film.title WHERE count(film_actor.*) > 5 and count(inventory.*) < 3
```

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John H. Drake, Qiaofu Zhang:
Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. EMNLP (Findings) 2021: 2030-2042

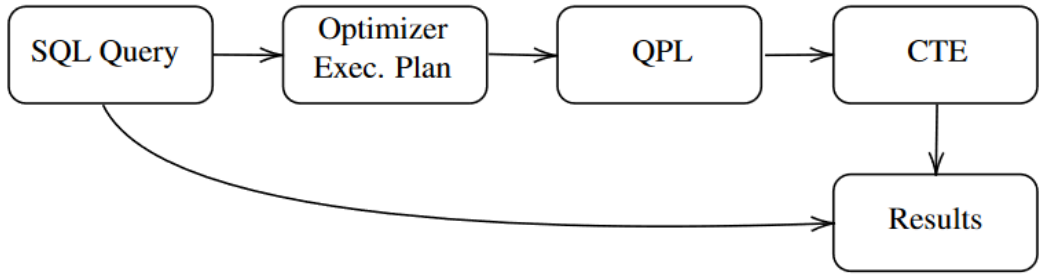
SQL-Like Syntax Language

QPL leverages the problem decomposition strategy to improve the parsing of intricate SQL queries.

- By breaking down a SQL query into modularized sub-queries, the complexity of the original query is reduced.
- This approach mitigates parsing difficulties associated with complex problems and cross-domain complex queries.

Operator	Description
Scan	Scan all rows in a table with optional filtering predicate
Aggregate	Aggregate a stream of tuples using a grouping criterion into a stream of groups
Filter	Remove tuples from a stream that do not match a predicate
Sort	Sort a stream according to a sorting expression
TopSort	Select the top-K tuples from a stream according to a sorting expression
Join	Perform a logical join operation between two streams based on a join condition
Except	Compute the set difference between two streams of tuples
Intersect	Compute the set intersection between two streams of tuples
Union	Compute the set union between two streams of tuples

Table 2: Description of QPL Operators



Ben Eyal, Moran Mahabi, Ophir Haroche, Amir Bachar, Michael Elhadad:
Semantic Decomposition of Question and SQL for Text-to-SQL Parsing. EMNLP (Findings) 2023: 13629-13645

SQL-Like Syntax Language

QDMR decomposes the original question into a number of atomic questions

- Each atomic question serves as an intermediate representation of the original question and can be translated into a set of small-scale formal operations involving tasks such as selecting entities, retrieving attributes, or aggregating information.

QDMR Step	Phrase-DB Linking	SQL
1. ships	1. SELECT(ship.id)	SELECT ship.id FROM ship;
2. injuries	2. SELECT(death.injured)	SELECT death.injured FROM death;
3. number of #2 for each #1	3. GROUP(count, #2, #1)	SELECT COUNT(death.injured) FROM ship, death WHERE death.caused_by_ship_id = ship.id GROUP BY ship.id;
4. #1 where #3 is highest	4. SUPER.(max, #1, #3)	SELECT ship.id FROM ship, death WHERE death.caused_by_ship_id = ship.id GROUP BY ship.id ORDER BY COUNT(death.injured) DESC LIMIT 1;
5. the name of #4	5. PROJECT(ship.name, #4)	SELECT ship.name FROM ship, death WHERE death.caused_by_ship_id = ship.id AND ship.id IN (#4);

Tomer Wolfson, Daniel Deutch, Jonathan Berant:
Weakly Supervised Text-to-SQL Parsing through Question Decomposition. NAACL-HLT (Findings) 2022: 2528-2542

SQL-Like Syntax Language

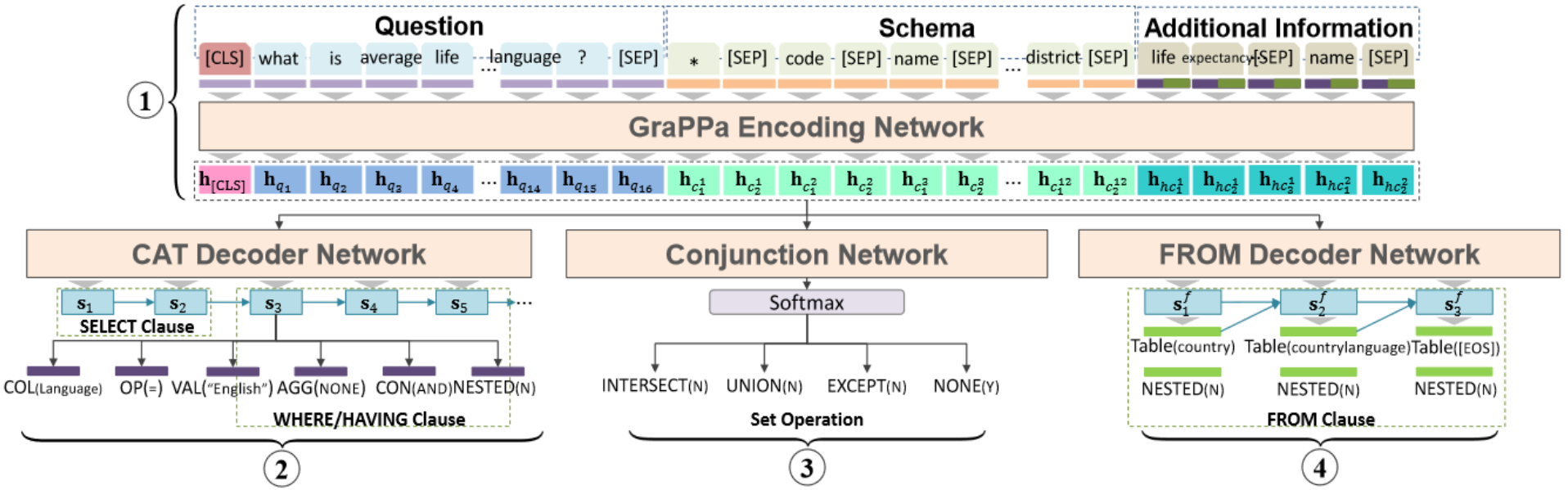
Limitations:

- High complexity and inadequate coverage of database structures
- Cost of manual design and deployment

SQL-Like Sketch Structure

CatSQL constructs a general template sketch with slots serving as initial placeholders.

- Its base model focus on the parsing of user queries to fill these placeholders, consequently decreasing the computational resource cost.
- It implements a novel semantic correction algorithm to assess the semantic accuracy of the resulting SQL queries and rectify any semantic issues detected in the generated queries.

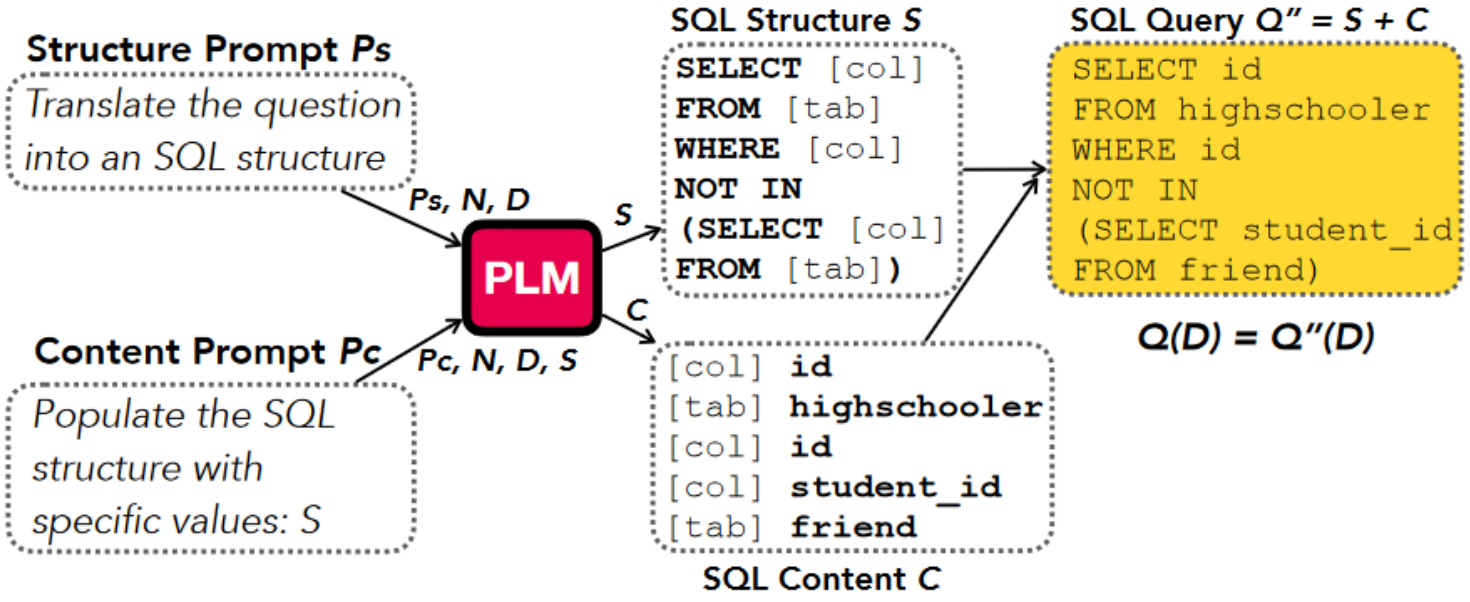


Han Fu, Chang Liu, Bin Wu, Feifei Li, Jian Tan, Jianling Sun:
CatSQL: Towards Real World Natural Language to SQL Applications. Proc. VLDB Endow. 16(6): 1534-1547 (2023)

SQL-Like Sketch Structure

SC-prompt utilizes a two-stage divide and conquer method for NL2SQL parsing

- It instructs PLM to generate specific SQL structures.
- In the subsequent phase, it directs the PLM to generate SQL structures containing actual values to fill the previously provided placeholders.

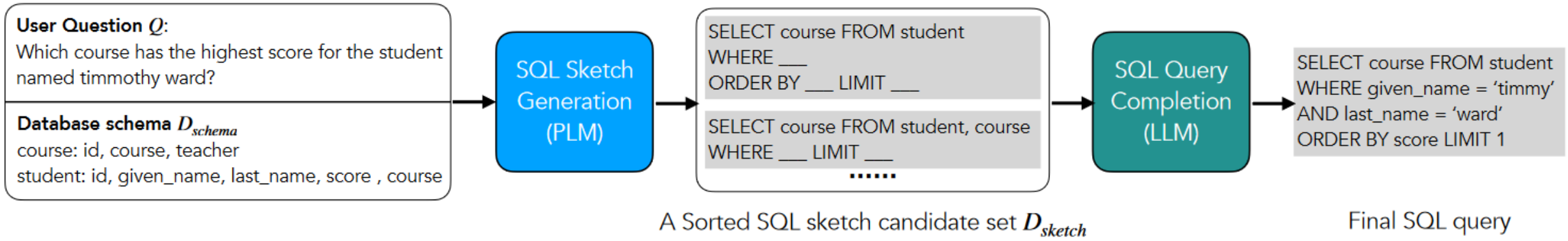


Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, Xiaoyong Du:
Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning. Proc. ACM Manag. Data 1(2): 147:1-147:28 (2023)

SQL-Like Sketch Structure

ZeroNL2SQL integrates the schema alignment capabilities of PLM with the complex reasoning capabilities of LLMs.

- Initially, it utilizes PLM to produce SQL sketches for achieving schema alignment and subsequently employs LLMs to execute complex content reasoning for populating missing information.
- Additionally, it also proposes a predicate calibration method for guiding the design of language models for SQL sketches based on database instances and selecting the optimal SQL query.



Zihui Gu, Ju Fan, Nan Tang, Songyue Zhang, Yuxin Zhang, Zui Chen, Lei Cao, Guoliang Li, Sam Madden, Xiaoyong Du: Interleaving Pre-Trained Language Models and Large Language Models for Zero-Shot NL2SQL Generation. CoRR abs/2306.08891 (2023)

SQL-Like Sketch Structure

Limitations:

- High computational cost
- Hard to construct accurate sketch when the query is much more complex

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
  - ⭐ Post-Processing
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

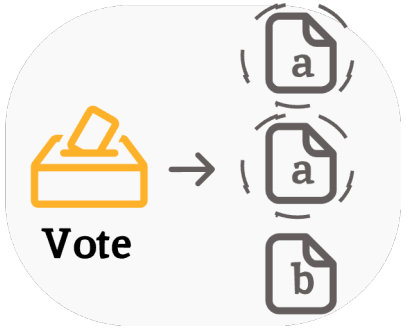
Post Processing

Motivation:

- Post-processing is a crucial step to refine the generated SQL queries, ensuring they meet user expectations more accurately. This involves enhancing the initial SQL output using various strategies.

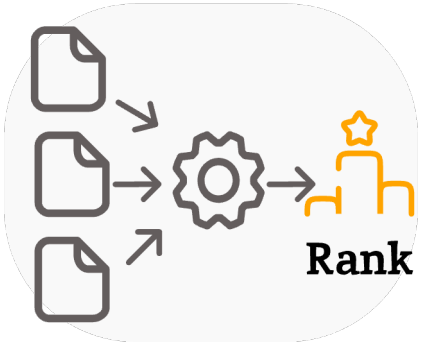
Correction

Consistency



Execution-Guided

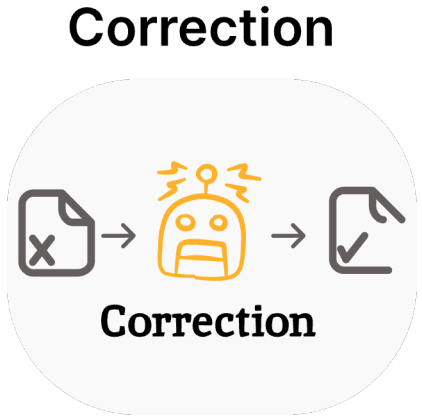
N-best Rerankers



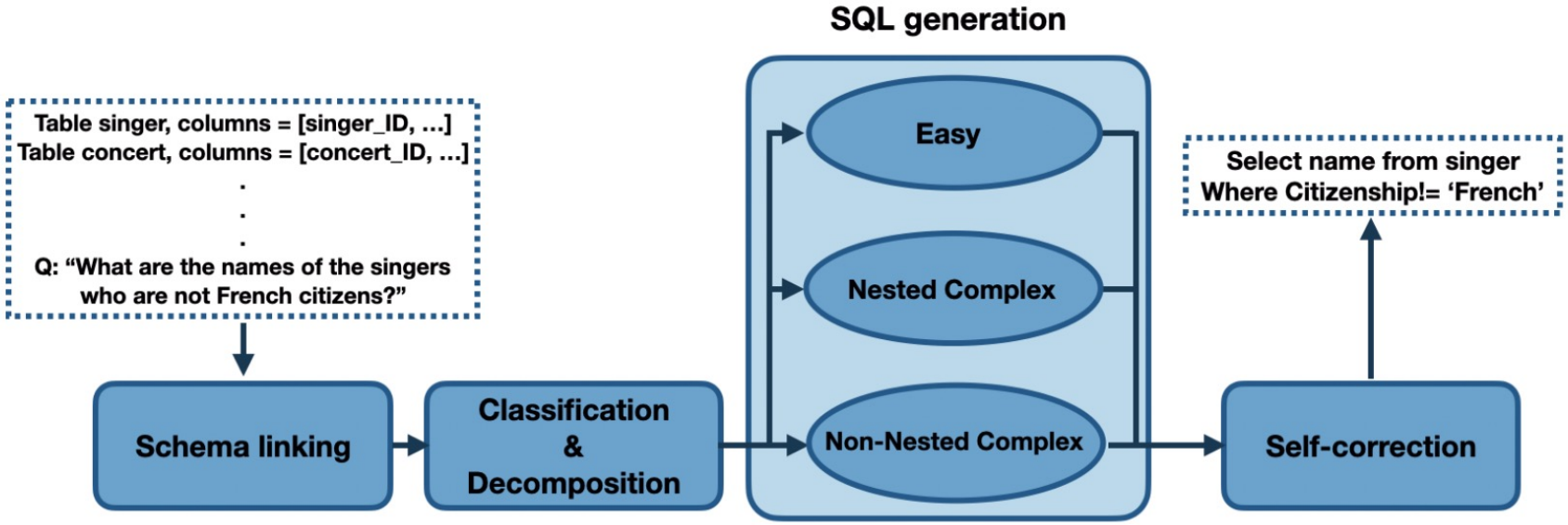
SQL Correction Strategies

Motivation:

- This module aims to identify and correct syntax errors in generated SQL queries.



DIN-SQL introduces a self-correction module to fix buggy SQL in a zero-shot setting, using different prompts for CodeX and GPT-4 models to identify and correct potential errors.



Pourreza, Mohammadreza, and Davood Rafiei. "Din-sql: Decomposed in-context learning of text-to-sql with self correction." Advances in Neural Information Processing Systems 36 (2024).

SQL Correction Strategies

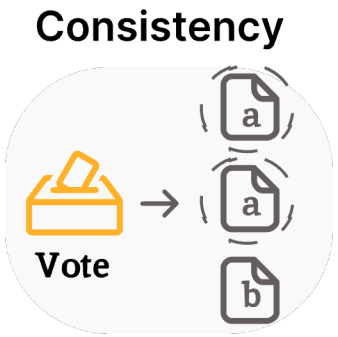
Limitations:

- Limited focus on syntax-specific corrections
- Lack of generalized error correction for other SQL errors
- Decreasing Relevance as LLM Capabilities Improve [????]

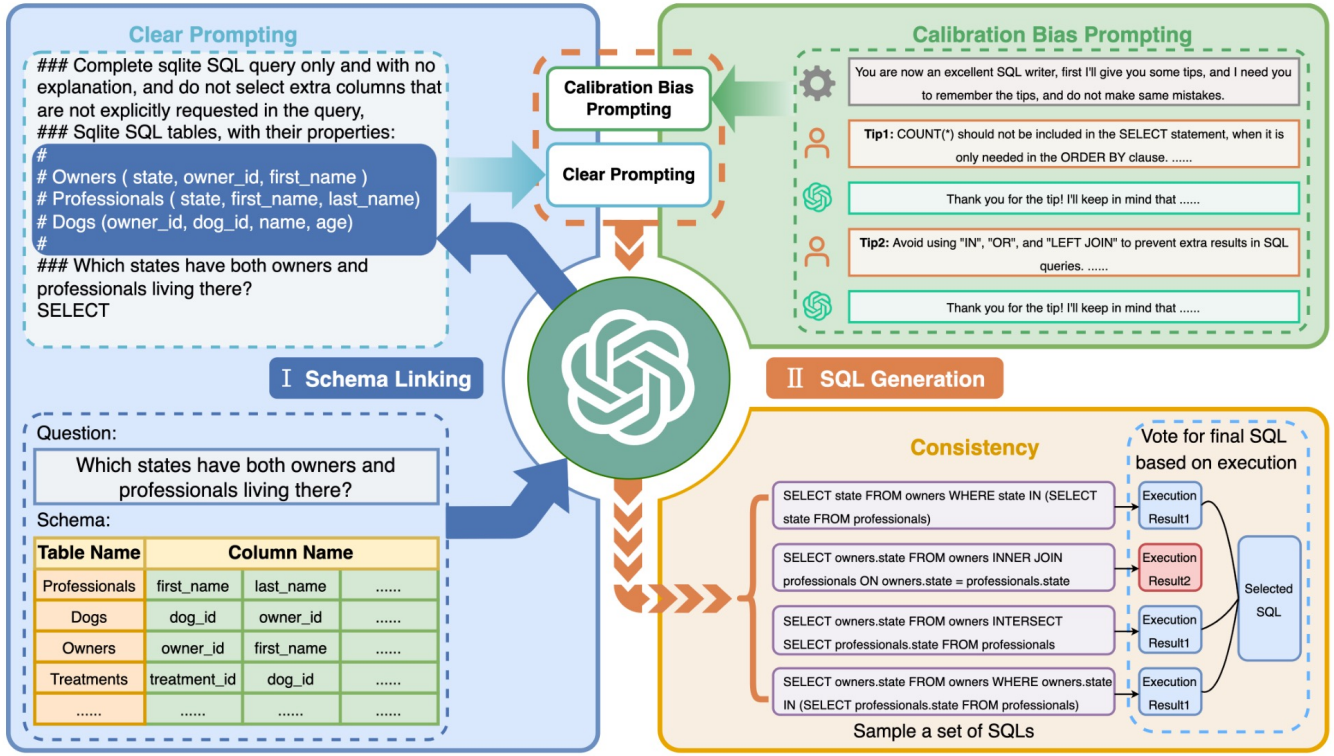
Output Consistency

Motivation:

This module ensures the uniformity of SQL queries by sampling multiple reasoning results and selecting the most consistent result.



The Consistency Output (CO) component in **C3-SQL** enhances zero-shot NL2SQL performance by generating multiple SQL queries, executing them, and using a voting mechanism on the results to select the most consistent SQL query.



Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Chen, lu, Lin, J. and Lou, D. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. arXiv.

Output Consistency

Limitations:

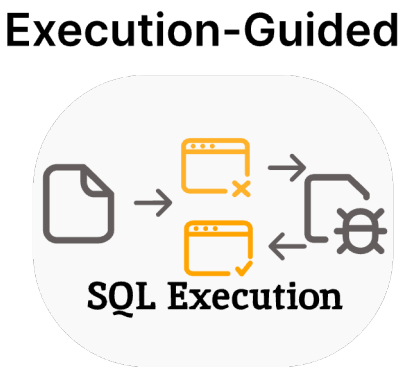
- Increased inference cost and time
- Reliance on multiple queries may not address all types of errors
- Effectiveness may decrease as LLM capabilities improve

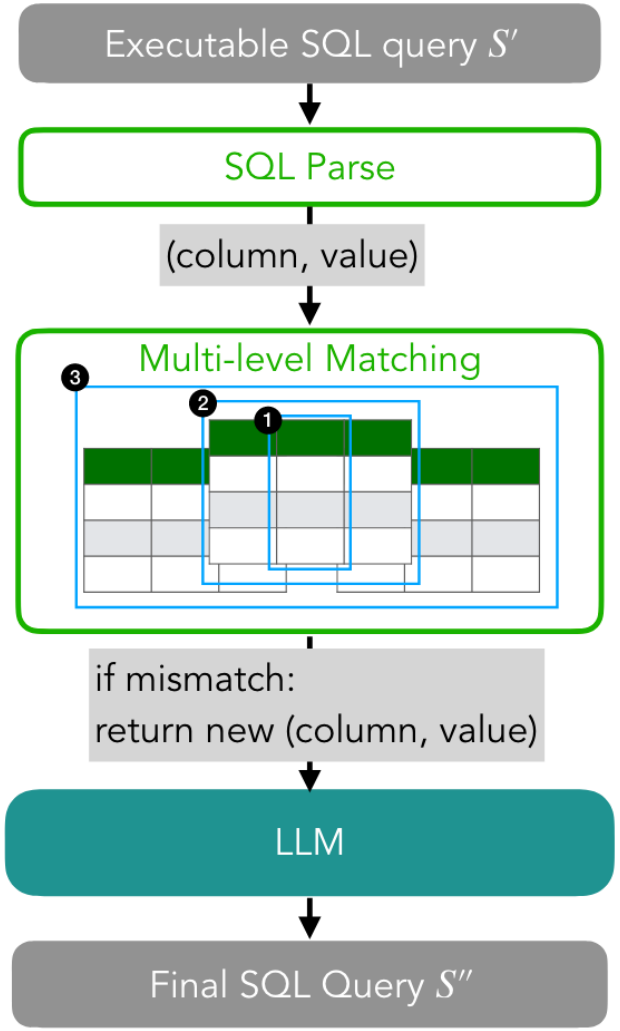
Execution-Guided Strategies

Motivation:

This module uses the execution results of SQL queries to guide subsequent refinements.

ZeroNL2SQL iteratively generates and refines SQL queries through feedback from executable checks, selecting the optimal query as the final output.





Gu, Z., Fan, J., Tang, N., Zhang, S., Zhang, Y., Chen, Z., Cao, L., Li, G., Madden, S. and Du, X. 2023. Interleaving Pre-Trained Language Models and Large Language Models for Zero-Shot NL2SQL Generation. arXiv.

Execution-Guided Strategies

Limitations:

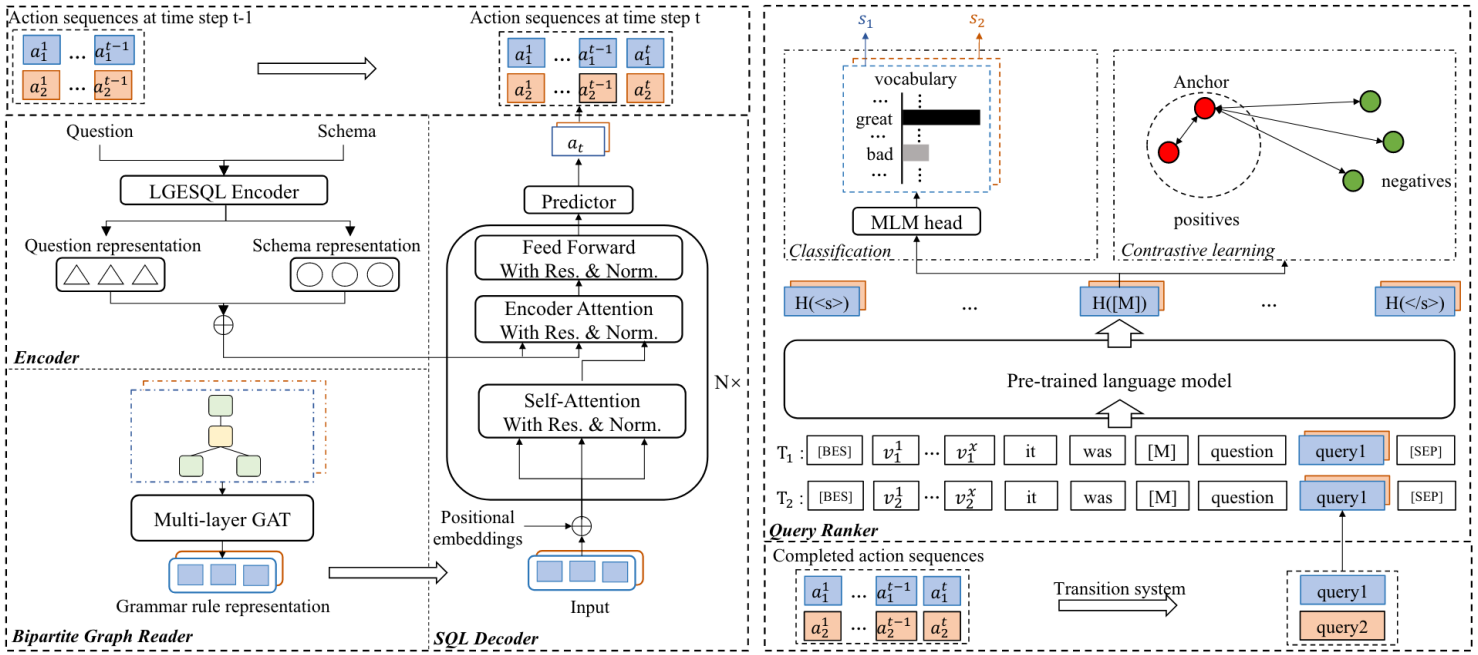
- Increased SQL generation time, especially with large databases
- Dependency on execution results may not address logical errors
- Limited scalability for complex or extensive databases

N-best Rankers Strategies

Motivation:

N-best re-ranking strategies reorder the top results from the model to improve query accuracy.

G3R introduces a feature-enhanced SQL reranker using hybrid prompt tuning and contrastive learning to improve query distinguishability without adding extra parameters.



Xiang, Y., Zhang, Q.-W., Zhang, X., Liu, Z., Cao, Y. and Zhou, D. G3R: A Graph-Guided Generate-and-Rerank Framework for Complex and Cross-domain Text-to-SQL Generation.

N-best Rankers Strategies

Limitations:

- Limited applicability to LLMs with stronger inference capabilities
- Reduced effectiveness as LLMs increase in size and sophistication
- Reliance on additional models or knowledge for effective filtering

Outline

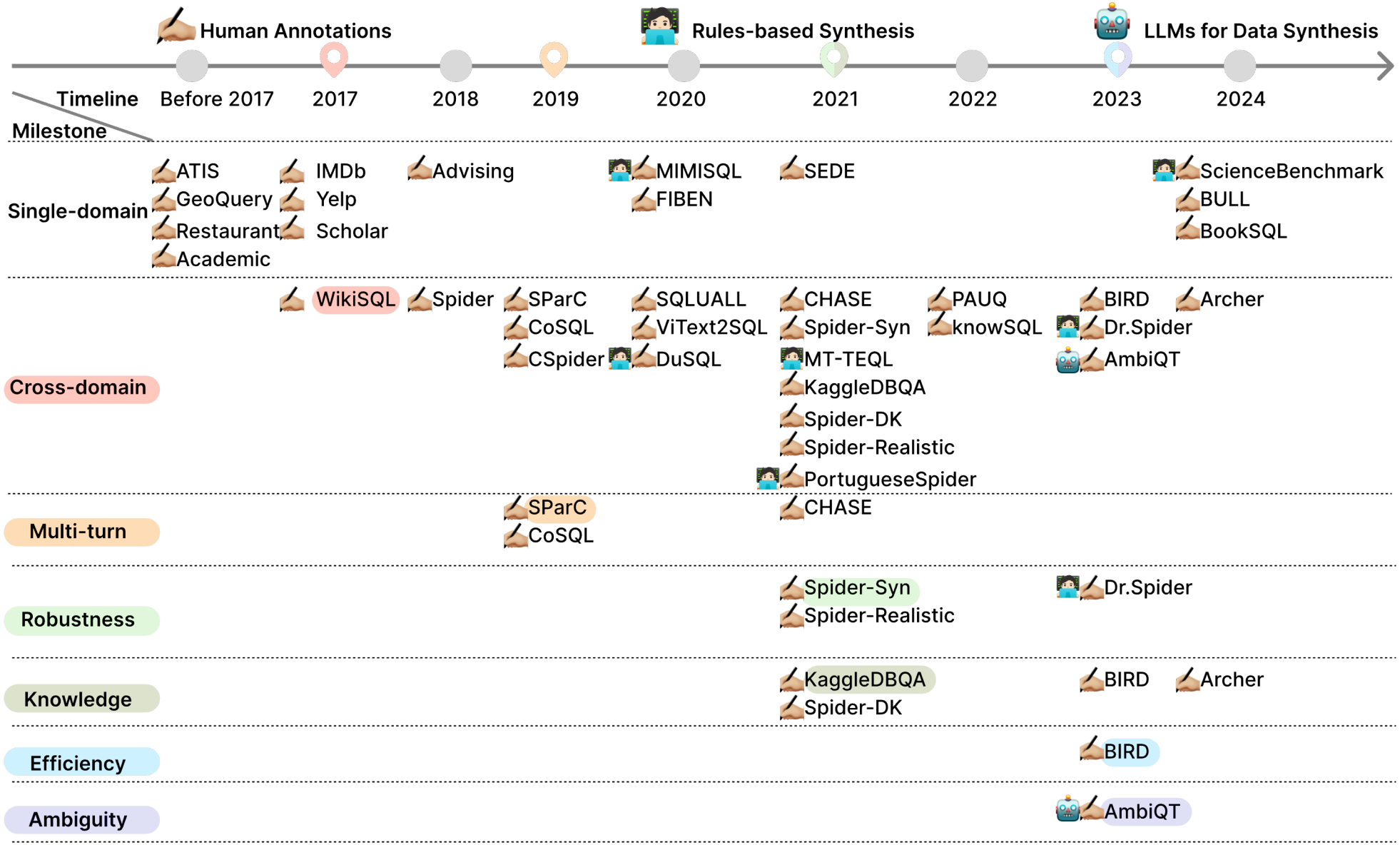
- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
  - Pre-Processing
  - NL2SQL Translation Methods
  - Post-Processing
- ⭐ NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Why We Need Benchmark?

Motivation:

- To **quantify the performance** of NL2SQL models
- To **provide sufficient data** for training NL2SQL models
- To **simulate the challenges** in real-world application scenarios

An Overview of NL2SQL Benchmarks



Single-domain Dataset

Motivation:

- Early NL2SQL datasets focused on specific domains, often featuring simple SQL query structures
- Recent single-domain datasets feature more complex databases and SQL queries specific to particular scenarios

Question:	Which funds have a fund establishment size exceeding 1 billion? And what is their three-year annualized return rate?
SQL Query:	SELECT a.**secuabbr**, a.**annualizedrrinthreeyear** FROM **mf_netvalueperformancehis** as a JOIN **mf_fundarchives** as b ON a.innercode = b.innercode Where b.**foundedsize** > 1000000000;

Table Name & Column Name	Table and Column Description
mf_netvalueperformancehis	Latest range performance of public fund net value
secuabbr	Fund abbreviation
annualizedrrinthreeyear	Three-year annualized return rate (%)
mf_fundarchives	Public fund overview
foundedsize	Fund establishment size (units)

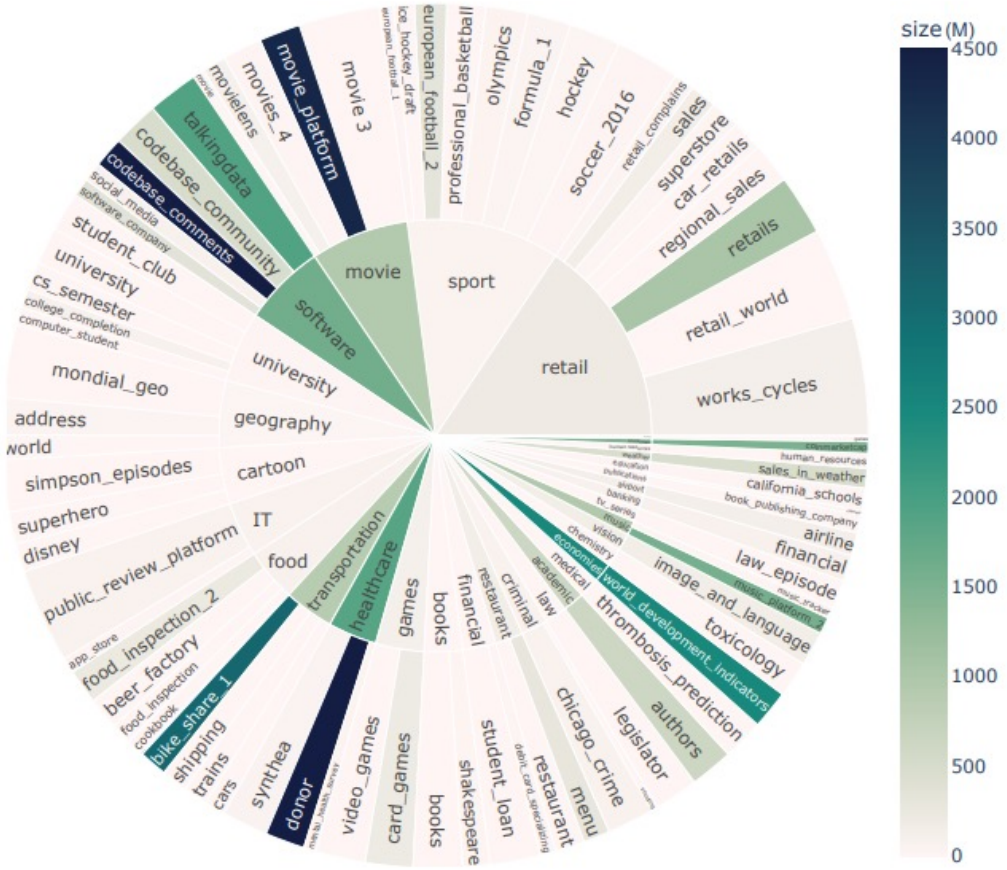
An example in **BULL** dataset

Zhang C, Mao Y, Fan Y, et al.
FinSQL: Model-Agnostic LLMs-based Text-to-SQL Framework for Financial Analysis[J]. arXiv preprint arXiv:2401.10506, 2024.

Cross-domain Dataset

Motivation:

- Cross-domain datasets challenge the generalization capabilities of NL2SQL systems across different domains, requiring these systems to generalize not only to new SQL queries but also to new databases.



Database domain distribution in BIRD dataset

Li J, Hui B, Qu G, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls[J]. Advances in Neural Information Processing Systems, 2024, 36.

Multi-turn Dataset

Motivation:

- In a real-world setting, users tend to ask a sequence of thematically related questions to learn about a particular topic or to achieve a complex goal.

D_1 : Database about student dormitory containing 5 tables.

C_1 : **Find the first and last names of the students who are living in the dorms that have a TV Lounge as an amenity.**

Q_1 : How many dorms have a TV Lounge?

S_1 :
```
SELECT COUNT(*) FROM dorm AS T1 JOIN has_amenity
AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity
AS T3 ON T2.amenid = T3.amenid WHERE
T3.amenity_name = 'TV Lounge'
```

Q_2 : What is the total capacity of these dorms?

S_2 :
```
SELECT SUM(T1.student_capacity) FROM dorm AS T1
JOIN has_amenity AS T2 ON T1.dormid = T2.dormid
JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid
WHERE T3.amenity_name = 'TV Lounge'
```

Q_3 : How many students are living there?

S_3 :
```
SELECT COUNT(*) FROM student AS T1 JOIN lives_in
AS T2 ON T1.stuid = T2.stuid WHERE T2.dormid IN
(SELECT T3.dormid FROM has_amenity AS T3 JOIN
dorm_amenity AS T4 ON T3.amenid = T4.amenid WHERE
T4.amenity_name = 'TV Lounge')
```

Q_4 : Please show their first and last names.

S_4 :
```
SELECT T1.fname, T1.lname FROM student AS T1 JOIN
lives_in AS T2 ON T1.stuid = T2.stuid WHERE
T2.dormid IN (SELECT T3.dormid FROM has_amenity
AS T3 JOIN dorm_amenity AS T4 ON T3.amenid =
T4.amenid WHERE T4.amenity_name = 'TV Lounge')
```

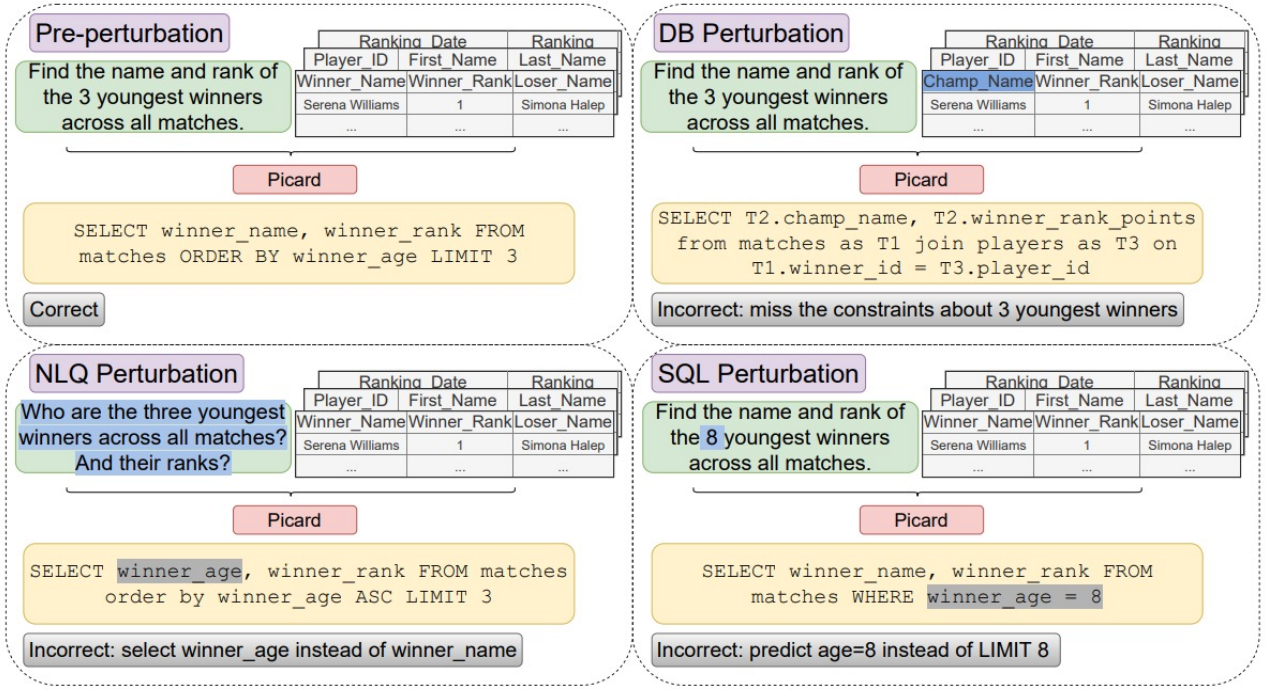
An example in SParC dataset

Yu T, Zhang R, Yasunaga M, et al. Sparc: Cross-domain semantic parsing in context[J]. arXiv preprint arXiv:1906.02285, 2019.

NL2SQL Datasets: Considering Robustness

Motivation:

- In real-world applications, NL2SQL systems need to handle diverse user groups and various database domains, making robustness a growing focus within the community.



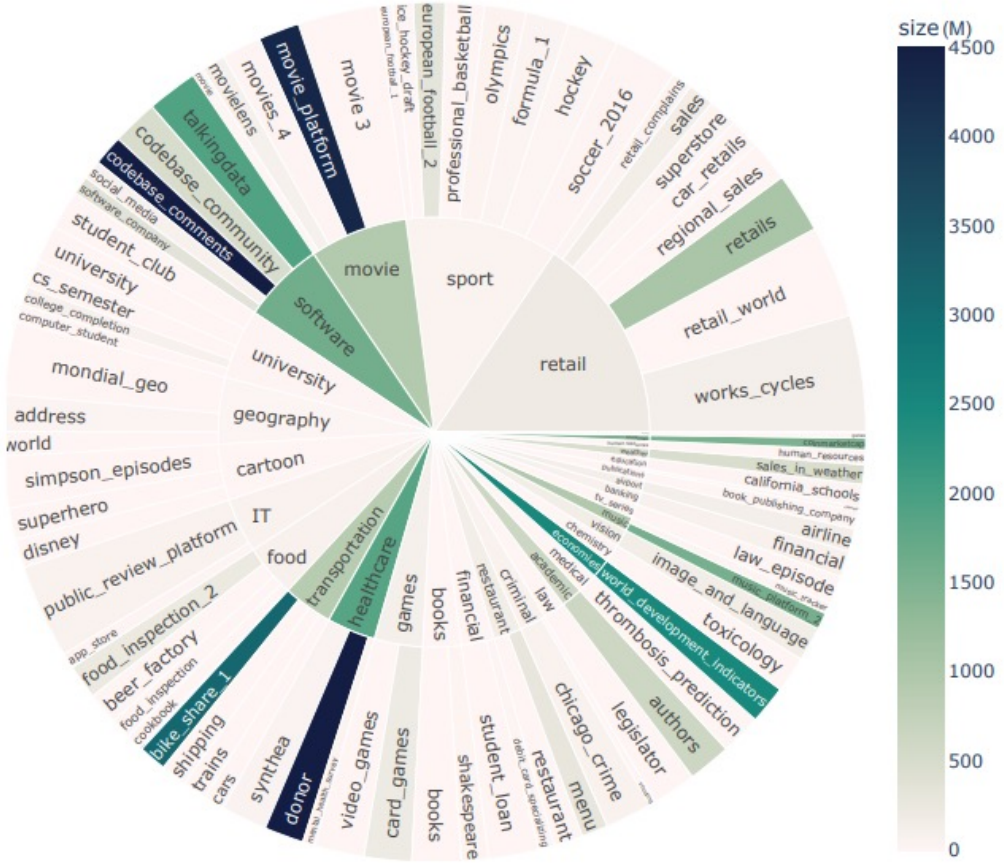
An example in Dr.Spider dataset

Chang S, Wang J, Dong M, et al.
Dr. spider: A diagnostic evaluation benchmark towards text-to-sql robustness[J]. arXiv preprint arXiv:2301.08881, 2023.

NL2SQL Datasets: Considering SQL Efficiency

Motivation:

- Databases in real-world scenarios often contain massive amounts of data, and a user question can be solved by multiple SQL queries.
- These SQL queries can vary in execution efficiency, which has attracted attention from the community.



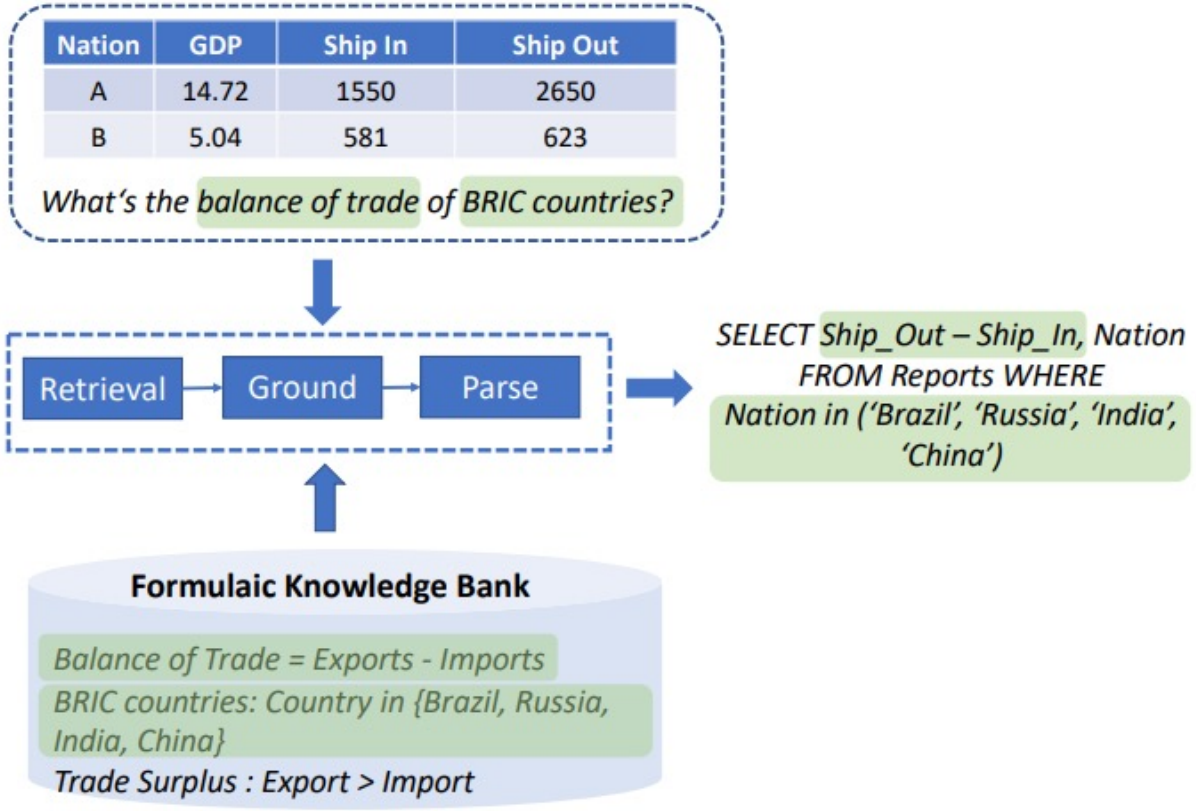
Database domain distribution with size in BIRD dataset

Li J, Hui B, Qu G, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls[J]. Advances in Neural Information Processing Systems, 2024, 36.

Knowledge-Augmented NL2SQL Datasets

Motivation:

- NL2SQL systems often require domain-specific knowledge to effectively perform the NL2SQL task in real-world applications within specific domains.



An example in KnowSQL

Dou L, Gao Y, Liu X, et al.
Towards knowledge-intensive text-to-SQL semantic parsing with formulaic knowledge[J]. arXiv preprint arXiv:2301.01067, 2023.

NL2SQL Datasets: Considering NL Ambiguity

Motivation:

- In real-world NL2SQL tasks, various ambiguities often arise, such as semantic ambiguities in NL and overlapping database schema.

Kind of ambiguity	Count	Example		
		Question Text	SQL #1	SQL #2
Column Ambiguity (C)	1240	List the ids of all students.	`SELECT roll_number FROM students`	`SELECT admission_number FROM students`
Table Ambiguity (T)	1417	How many singers do we have?	`SELECT COUNT(*) FROM artist`	`SELECT COUNT(*) FROM performer`
Join Ambiguity (J)	288	What are the makers and models?	`SELECT maker, model FROM model`	`SELECT t2.maker, t1.model FROM model AS t1 JOIN model_maker AS t2 ON t1.model_id = t2.model_id`
Precomputed Aggregates (P)	101	Find the average weight for each pet type.	`SELECT AVG(weight), pettype FROM pets GROUP BY pettype`	`SELECT avg_weight, pettype FROM pets_weight`

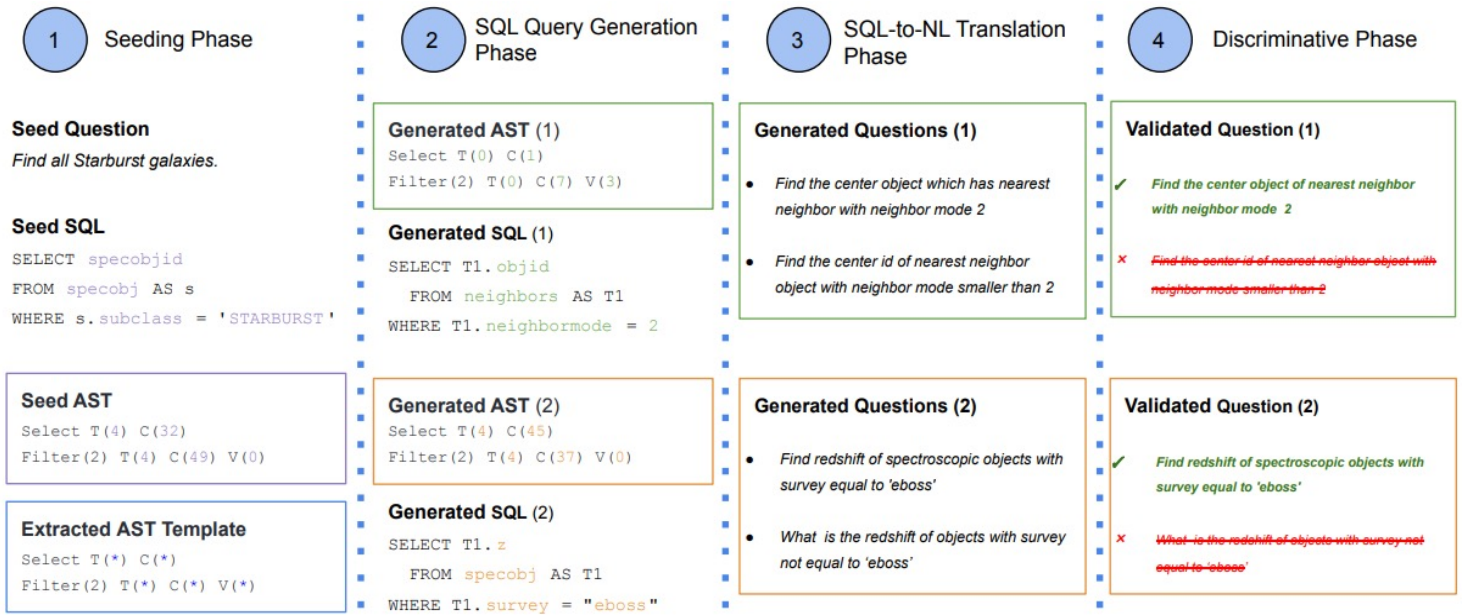
An example in AmbiQT

Bhaskar A, Tomar T, Sathe A, et al.
Benchmarking and improving text-to-sql generation under ambiguity[J]. arXiv preprint arXiv:2310.13659, 2023.

Synthetic NL2SQL Datasets

Motivation:

- Despite the fact that many datasets have been manually annotated so far, the rapid development of LLMs in the NL2SQL field has led to a growing demand for NL2SQL data.



End-to-end architecture for automatic training data generation in ScienceBenchmark

Zhang Y, Deriu J, Katsogiannis-Meimarakis G, et al.
Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. VLDB 2024.

The Statistics of NL2SQL Benchmarks

TABLE II: Statistics of NL2SQL Benchmarks.

Dataset	Redundancy Measure			DB Complexity					Query Complexity				
	#-Questions	#-Unique Queries	#-Questions / #-Queries	#-DBs	#-Tables	#-Tables / DB	#-Cols / Table	#-Records / DB	Tables	Selects	Agg	Scalar Func	Math Comp
ATIS [117]	5280	947	5.6	1	25	25	5.24	162243	8.39	1.79	0.22	0	0
GeoQuery [118]	877	246	3.6	1	7	7	4.14	937	2.22	2.19	0.92	0	0.01
Restaurants [119]	378	23	16.4	1	3	3	4.00	19295	2.43	1.17	0.35	0	0
Academic [108]	196	185	1.1	1	17	17	3.12	58249674	3.48	1.04	0.54	0	0
IMDb [120]	131	89	1.5	1	17	17	3.94	40147386	2.91	1.01	0.30	0	0
Yelp [120]	128	110	1.2	1	8	8	5	4823945	2.41	1	0.45	0	0
Scholar [121]	817	193	4.2	1	10	10	2.50	147416275	3.38	1.02	0.68	0	0.02
WikiSQL [40]	80654	80257	1	26531	26531	1	6.34	17	1	1	0.28	0	0
Advising [122]	4387	205	21.4	1	15	15	7.40	332596	3.41	1.21	0.40	0	0.11
Spider [31]	11840	6448	1.8	206	1056	5.13	5.01	8980	1.83	1.17	0.54	0	0
SParC [123]	10228	8981	1.1	166	876	5.28	5.14	9665	1.58	1.10	0.44	0	0
CoSQL [124]	8350	8007	1	166	876	5.28	5.14	9665	1.54	1.11	0.42	0	0
CSpider [125]	11840	6408	1.8	206	1056	5.13	5.01	8980	1.83	1.17	0.54	0	0
MIMICSQL [126]	20000	10000	2	-	-	-	-	-	1.74	1	0.84	0	0
SQUALL [127]	11276	8296	1.4	2108	4028	1.91	9.18	71	1.22	1.29	0.40	0.03	0.16
FIBEN [128]	300	233	1.3	1	152	152	2.46	11668125	5.59	1.56	0.97	0	0.04
ViText2SQL [129]	9693	5223	1.9	166	876	5.28	5.14	9665	1.17	1.12	0.54	0	0
DuSQL [130]	25003	20308	1.2	208	840	4.04	5.29	20	1.49	1.25	0.73	0	0.30
PortugueseSpider [131]	9693	5275	1.8	166	876	5.28	5.14	9665	1.85	1.17	0.54	0	0
CHASE [132]	15408	13900	1.1	350	1609	4.60	5.19	4594	1.81	1.16	0.31	0	0
Spider-Syn [133]	1034	550	1.9	166	876	5.28	5.14	9665	1.68	1.17	0.59	0	0
Spider-DK [134]	535	283	1.9	169	887	5.25	5.14	9494	1.71	1.16	0.54	0	0
Spider-Realistic [135]	508	290	1.8	166	876	5.28	5.14	9665	1.79	1.21	0.50	0	0
KaggleDBQA [136]	272	249	1.1	8	17	2.12	10.53	595075	1.25	1.05	0.69	0	0.04
SEDE [137]	12023	11421	1.1	1	29	29	7.28	-	1.90	1.29	0.94	0.49	0.49
MT-TEQL [138]	489076	4525	108.1	489076	3279004	6.70	5.51	-	1.69	1.15	0.53	0	0
PAUQ [139]	9876	5497	1.8	166	876	5.28	5.14	9693	1.82	1.17	0.53	0	0
knowSQL [140]	28468	-	-	488	-	-	-	-	-	-	-	-	-
Dr.Spider [141]	15269	3847	4	549	2197	4	5.54	28460	1.81	1.19	0.52	0	0
BIRD [49]	10962	10840	1	80	611	7.64	7.14	4585335	2.07	1.09	0.61	0.20	0.27
AmbiQT [142]	3046	3128	1	166	876	5.28	5.14	9665	1.85	1.17	0.51	0	0.01
ScienceBenchmark [143]	5031	3654	1.4	-	-	-	-	-	1.45	1	0.24	0	0.07
BULL [47]	7932	5864	1.4	3	78	26	14.96	85631	1.22	1	0.18	0.42	0.05
BookSQL [144]	78433	39530	2	1	7	7	8.86	1012948	1.25	1.12	0.78	0.39	0.22
Archer [145]	518	260	2	10	68	6.8	6.81	31365.3	3.89	3.07	1.77	0.1	3.55

NL2SQL Benchmark Discussion & Insights

- From the Redundancy Measure perspective
  - We observe a trend from early datasets to recent ones where datasets have grown in size, including increases in the number of questions and unique queries
- From the Database Complexity perspective
  - The number of databases (and tables) in datasets correlates with the tasks (e.g., Single-domain vs. Robustness) they serve
- From the Query Complexity perspective
  - Recent datasets show a growing emphasis on Scalar Functions and Mathematical Computations in SQL queries, which introduces challenges in SQL generation structure not seen in earlier datasets

The Next For NL2SQL Benchmark

- Despite the growing number of datasets proposed by the NL2SQL community, we find that current datasets still exhibit a gap in SQL complexity compared to realworld scenarios
  - Recent datasets generally feature lower counts of SELECT keyword, indicating fewer nested SQL queries or complex set operations
  - Challenges related to Scalar Functions (e.g., ROUND) and Mathematical Computations (e.g., Addition) also need further attention

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
  - ⭐ Evaluation Metrics
  - Evaluation Toolkit
  - Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

NL2SQL Evaluation Metrics

- Accuracy
  - Execution Accuracy (EX)
  - String-Match Accuracy (SM)
  - Component-Match Accuracy (CM)
  - Exact-Match Accuracy (EM)
- Efficiency
  - Valid Efficiency Score (VES)
- Robustness
  - Query Variance Testing (QVT)

NL2SQL Evaluation Metrics

- **Accuracy**
  - Execution Accuracy (EX)

$$EX = \frac{\sum_{i=1}^{N} 1\left(V_i = \hat{V}_i\right)}{N}$$

- N : the size of dataset
- V_i : the execution result set of the i-th ground-truth SQL query
- $\hat{V}_i$: the execution result set of the i-th predicted SQL query
- $1(\bullet)$: an indicator function that equals 1 if the condition inside is satisfied, and 0 otherwise

Yu T, Zhang R, Yang K, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task[J]. arXiv preprint arXiv:1809.08887, 2018.

NL2SQL Evaluation Metrics

- **Accuracy**
  - String-Match Accuracy (SM)

$$SM = \frac{\sum_{i=1}^{N} 1\left(Y_i = \hat{Y}_i\right)}{N}$$

- N : the size of dataset
- Y_i : the i-th ground-truth SQL query
- $\hat{Y}_i$: the i-th predicted SQL query
- $1(\bullet)$: an indicator function that equals 1 if the condition inside is satisfied, and 0 otherwise

Zhong V, Xiong C, Socher R. Seq2sql: Generating structured queries from natural language using reinforcement learning[J]. arXiv preprint arXiv:1709.00103, 2017.

NL2SQL Evaluation Metrics

- **Accuracy**
  - Component-Match Accuracy (CM)

$$CM^C = \frac{\sum_{i=1}^{N} 1\left(Y_i^C = \hat{Y}_i^C\right)}{N}$$

SQL Components:
- SELECT
- WHERE
- GROUP
- ……

- N : the size of dataset
- Y_i^C : the component of the i-th ground-truth SQL query Y_i
- $\hat{Y}_i^C$: the component of the i-th predicted SQL query $\hat{Y}_i$
- $1(\bullet)$: an indicator function that equals 1 if the condition inside is satisfied, and 0 otherwise

Yu T, Zhang R, Yang K, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task[J]. arXiv preprint arXiv:1809.08887, 2018.

NL2SQL Evaluation Metrics

- **Accuracy**
  - Exact-Match Accuracy (EM)

$$EM = \frac{\sum_{i=1}^{N} 1\left(\wedge_{C_k \in C} Y_i^{C_k} = \hat{Y}_i^{C_k}\right)}{N}$$

- N : the size of dataset
- C_k: the k-th component of SQL query (e.g., WHERE)
- $Y_i^{C_k}$: the k-th component of the i-th ground-truth SQL query
- $\hat{Y}_i^{C_k}$: the k-th component of the i-th predicted SQL query
- $1(\bullet)$: an indicator function that equals 1 if the condition inside is satisfied, and 0 otherwise

- Yu T, Zhang R, Yang K, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task[J]. arXiv preprint arXiv:1809.08887, 2018.

NL2SQL Evaluation Metrics

- **Efficiency**
  - Valid Efficiency Score (VES)

$$VES = \frac{\sum_{i=1}^{N} 1\left(V_i = \hat{V}_i\right) \cdot R\left(Y_i, \hat{Y}_i\right)}{N} \qquad R\left(Y_i, \hat{Y}_i\right) = \sqrt{\frac{E\left(Y_i\right)}{E\left(\hat{Y}_i\right)}}$$

- N : the size of dataset
- V_i : the execution result set of the i-th ground-truth SQL query Y_i
- $\hat{V}_i$: the execution result set of the i-th predicted SQL query $\hat{Y}_i$
- E(•) : measures the efficiency of specific SQL query, which can be refer to execution time

Li J, Hui B, Qu G, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls[J]. Advances in Neural Information Processing Systems, 2024, 36.

NL2SQL Evaluation Metrics

- **Robustness**
  - Query Variance Testing (QVT)

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

- N : the size of dataset
- $\phi\left(Q_{ij}\right)$: the predicted SQL query for the j-th NL variation of Y_i
- m_i : denotes the number of different NL variations for the SQL query Y_i

Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?[J]. arXiv preprint arXiv:2406.01265, 2024.

Outline

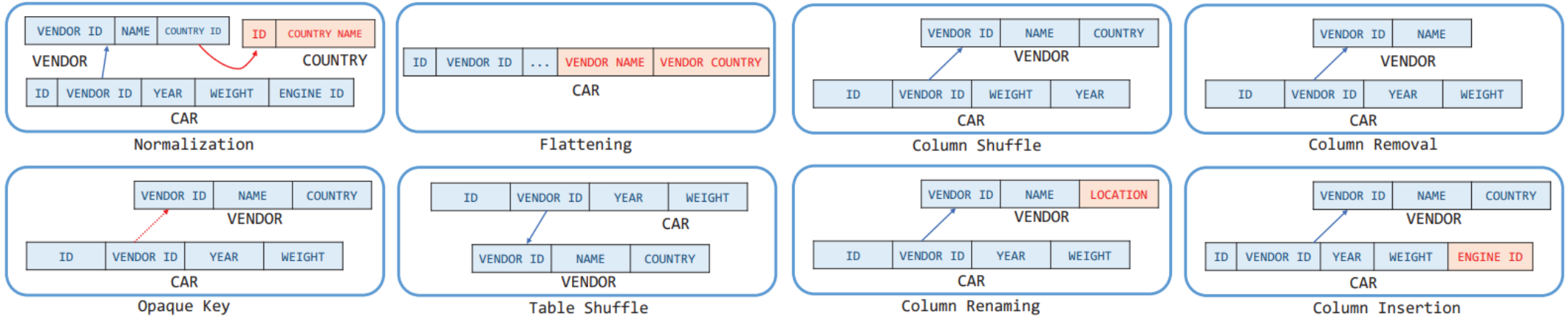
- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
  - Evaluation Metrics
  - ⭐ Evaluation Toolkit
  - Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

MT-TEQL

- MT-TEQL
  - NL Utterance Variations
  - Database Schema Variations

Type	Illustrative Examples
Common Interrogative Prefix	what is/are, which is/are
Common Declarative Prefix	tell me, return, find, list
Special Interrogative Prefix	when, where, how many
Special Declarative Prefix	count

Categorizations of frequently-used prefixes in typical NLIDB utterances

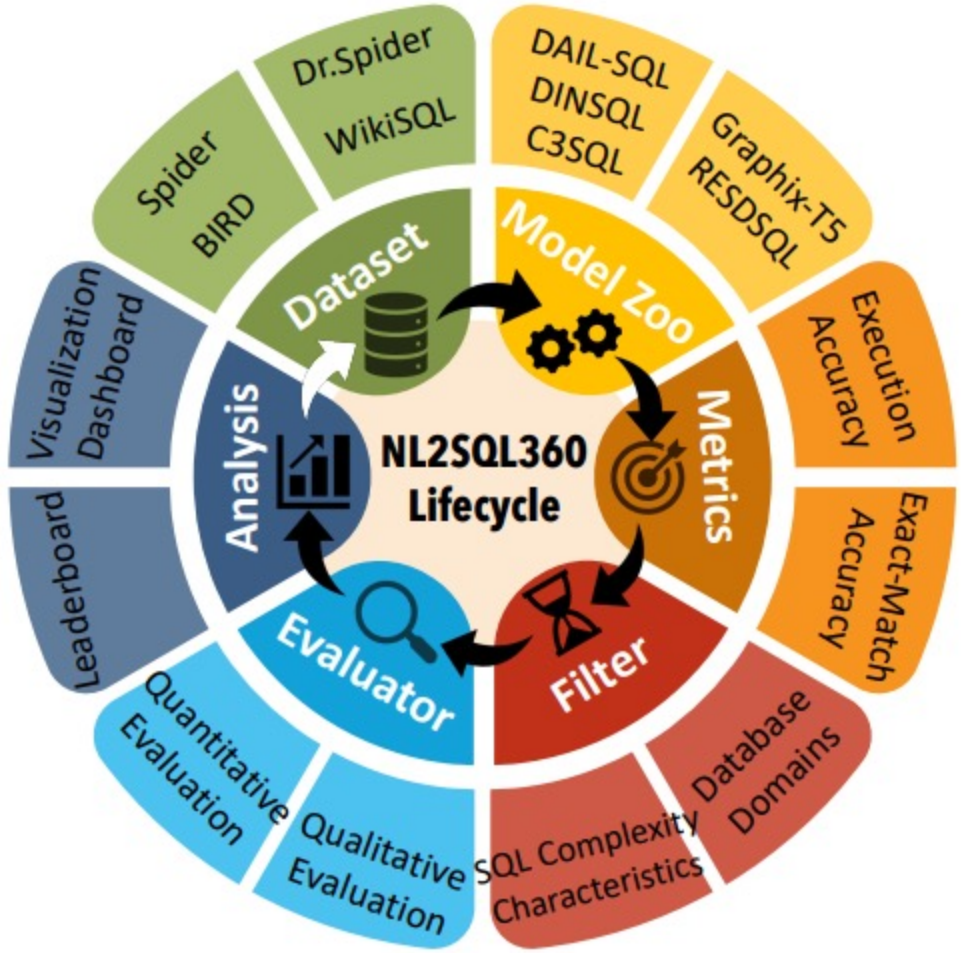


Illustrative examples of schema-oriented MRs

Ma P, Wang S. Mt-teql: evaluating and augmenting neural nlidb on real-world linguistic and schema variations[J]. Proceedings of the VLDB Endowment, 2021, 15(3): 569-582.

NL2SQL360

- NL2SQL360
  - SQL Complexity
  - SQL Characterisitics (e.g., JOIN)
  - Database Domain Adaption
  - NL Query Variance Testing
  - Economy
  - Efficiency
- Features
  - Multi-angle & Fine-grained Evaluation
  - Scenario-based Evaluation



Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?[J]. arXiv preprint arXiv:2406.01265, 2024.

NL2SQL360: User Interface

Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?. VLDB 2024.

NL2SQL360: User Interface

Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?. VLDB 2024.

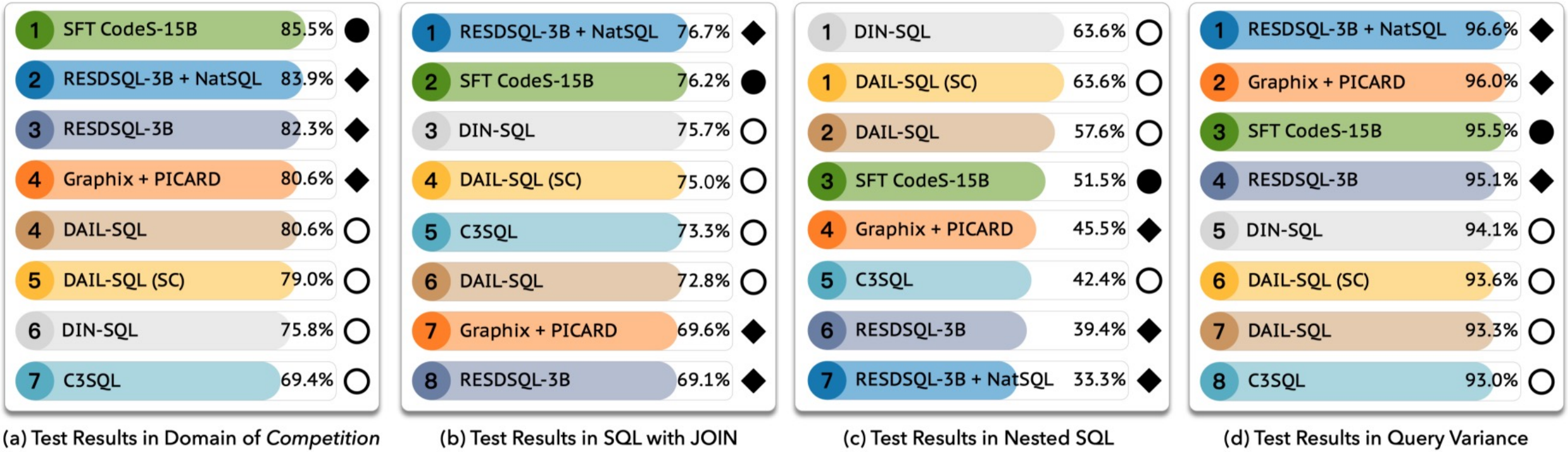
NL2SQL360 Case Study

- Dataset: Spider
- Model:
  - LLM-based methods:
    - DAILSQL, DAILSQL(SC)
    - DINSQL
    - C3SQL
    - SFT CodeS-1B/3B/7B/15B
  - PLM-based methods:
    - RESDSQL-3B, RESDSQL-3B + NatSQL
    - Graphix + PICARD

- Metric:
  - Execution Accuracy (EX)
  - Exact-Match Accuracy (EM)
  - Valid Efficiency Score (VES)
  - Query Variance Testing (QVT)

- Multi-Angle:
  - SQL Characteristics
  - SQL Complexity
  - NL Variance
  - Database Domain Adaption
  - Efficiency
  - SQL Efficiency
  - Economy (Cost)

NL2SQL360: Leaderboard



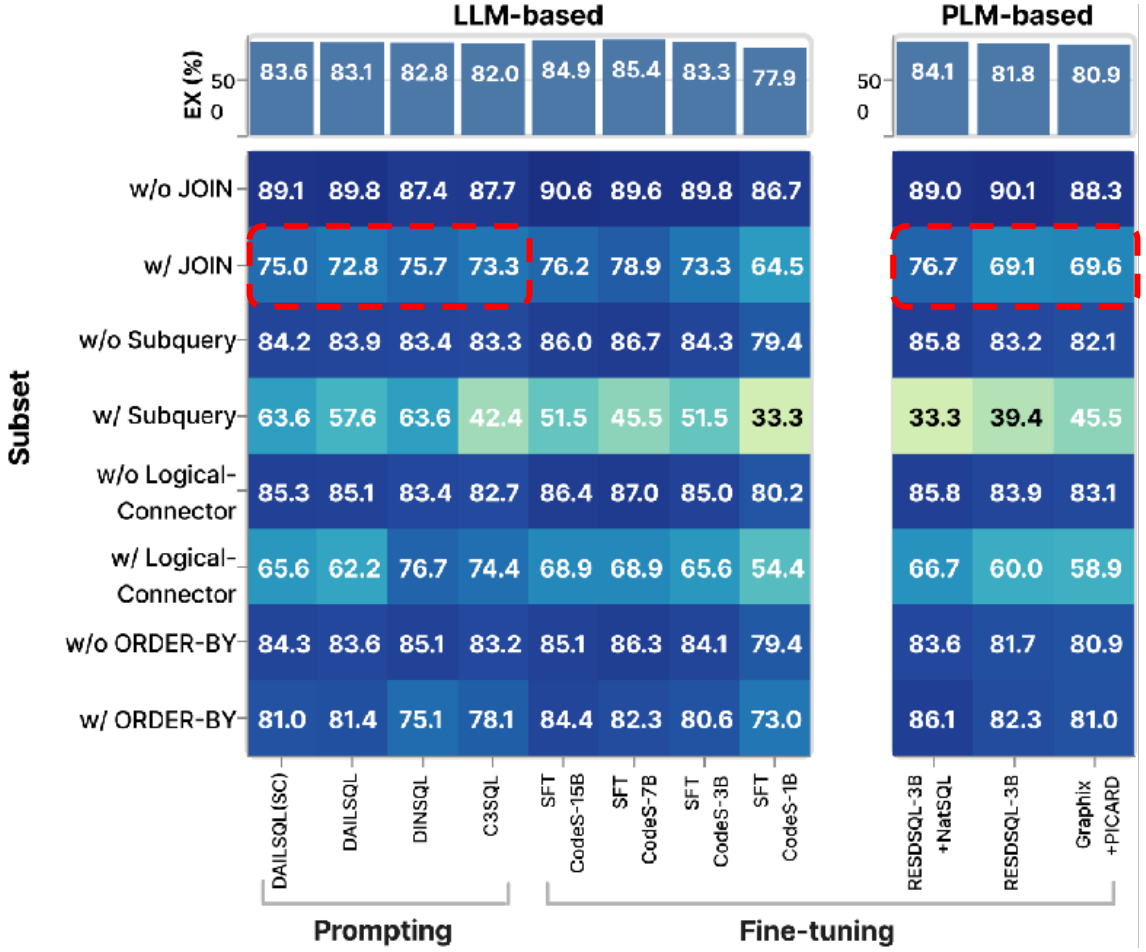
(a) Test Results in Domain of *Competition* (b) Test Results in SQL with JOIN (c) Test Results in Nested SQL (d) Test Results in Query Variance

Figure 3: NL2SQL Models on Spider from Different Angles (○: Prompting LLM, ●: Fine-tuning LLM, ◆: Fine-tuning PLM).

Leaderboard Link: https://nl2sql360.github.io/

Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?. VLDB 2024.

NL2SQL360 Case Study – JOIN Finding



We find that:
DINSQL outperforms in LLMs with prompting.
RESDSQL + NatSQL outperforms in PLMs.

Commonality:
Both methods use **NatSQL** as the intermediate representation (IR) of SQL.

Finding:
Taking **NatSQL** reduces the complexity of predicting JOIN operations and potentially enhances the model performance.

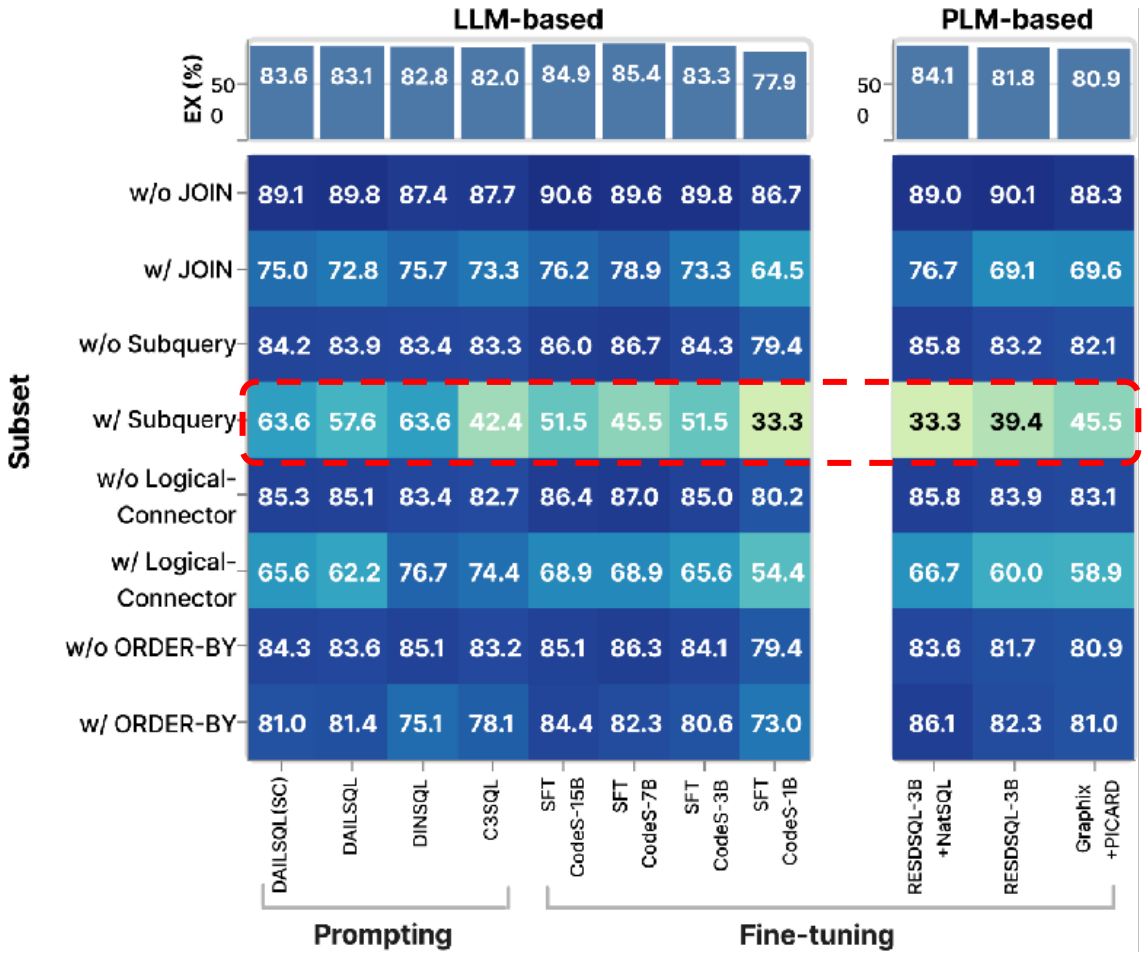
Question :
Find the name of students who have a pet

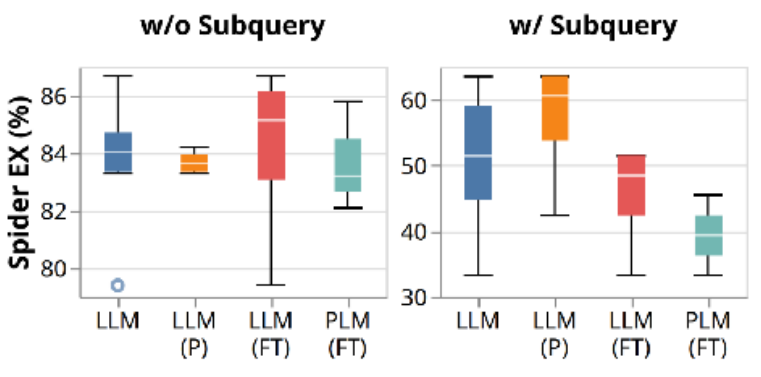
SQL :
```
SELECT  T1.name FROM  student AS T1
JOIN has_pet AS T2 ON T1.stuid=T2.stuid
```

NatSQL : (Extend FROM clause)
```
SELECT student.name FROM student, has_pet
```

Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?. VLDB 2024.

NL2SQL360 Case Study – Subquery Finding





(P): Prompt-based
(FT): Fine-tuning-based

In scenarios involving subqueries, LLM-based methods outperform PLM-based methods overall, with methods using GPT-4 showing particularly better performance.

The inherent reasoning ability of these models is likely crucial for success in predicting the subqueries.

NL2SQL360 Case Study – QVT

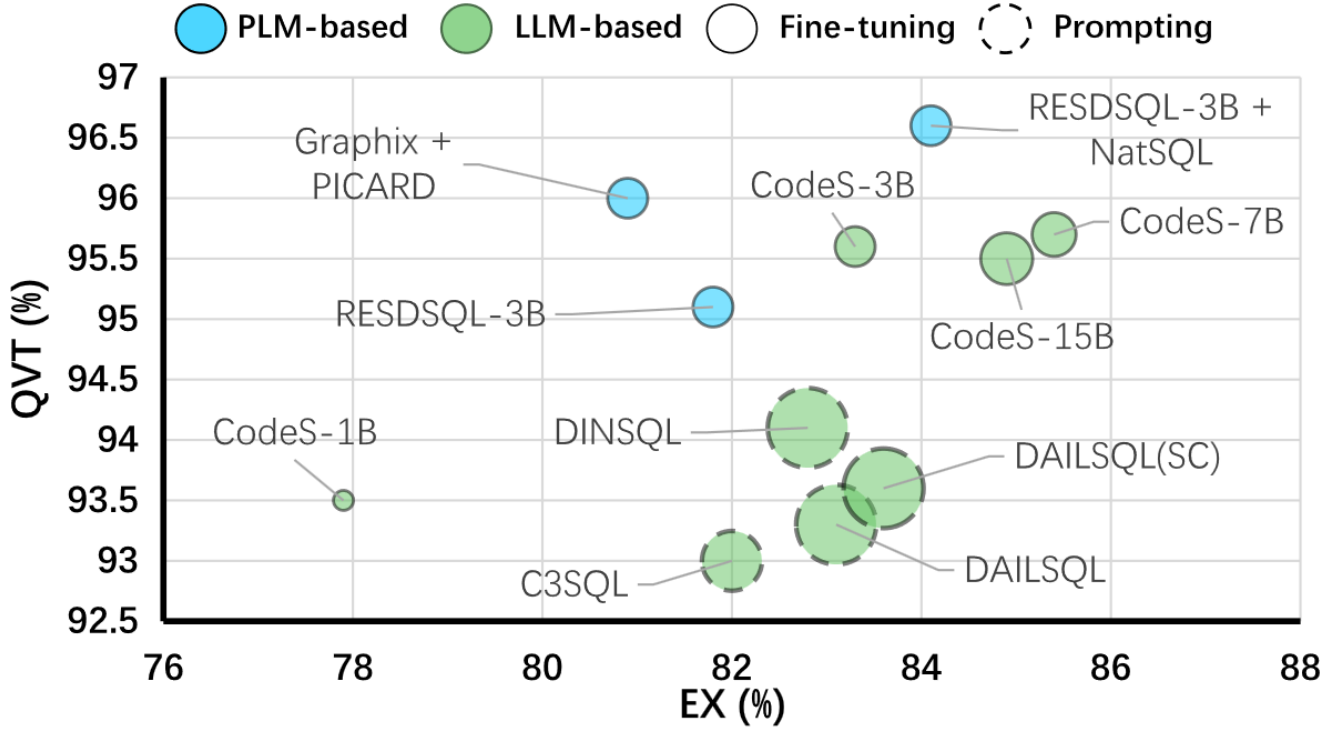
- To further evaluate the robustness and flexibility of NL2SQL solutions in handling variations in natural language queries, we propose a new metric called **Query Variance Testing (QVT)**. This metric assesses how well the models can adapt to different forms of NL queries.

Query Variance Testing (QVT): Given a SQL query Q_i, there typically exist multiple corresponding NL queries, denoted as pairs $\{(N_1, Q_i), (N_2, Q_i), \ldots, (N_m, Q_i)\}$. The formula for computing QVT accuracy is defined as follows:

$$QVT = \frac{1}{M} \sum_{i=1}^{M} \left(\frac{\sum_{j=1}^{m_i} \mathbb{1}\left(\mathcal{F}(N_{ij}) = Q_i\right)}{m_i} \right)$$

- M is the total number of SQL queries in the test set.
- m_i is the number of natural language query variations corresponding to the SQL query Q_i .
- $F(N_{ij})$ represents the SQL query generated by the model for the *j*-th natural language query variation of Q_i.

NL2SQL360 Case Study – QVT



- Fine-tuned LLMs generally exhibit higher QVT than prompting LLMs.
- Notably, although the Graphix+PICARD method underperforms in overall EX compared to all prompt-based LLMs, it surpasses them in QVT.
- **Fine-tuning the model with task-specific datasets may help stabilize its performance against NL variations.**

Li B, Luo Y, Chai C, et al. The Dawn of Natural Language to SQL: Are We Fully Ready?. VLDB 2024.

Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
  - Evaluation Metrics
  - Evaluation Toolkit
  - ⭐ Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- Open Problems

Error Analysis

Motivation:

- Error Analysis provides valuable insights into the limitations and challenges faced by current models.

Goal:

- By systematically examining errors, researchers can identify specific areas for improvement, enhance model robustness, and develop more effective training strategies.

Existing Taxonomies for NL2SQL Errors

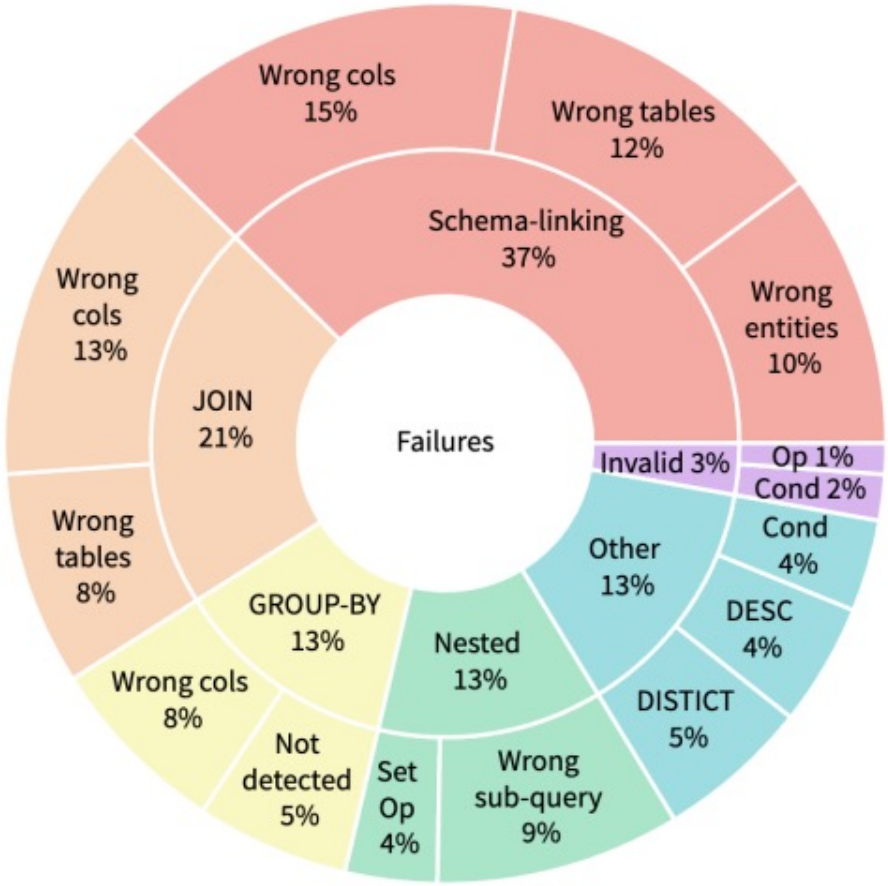
- Schema Linking
- Misunderstanding Database
- Misunderstanding Knowledge
- Reasoning
- Syntax-Related Errors
- Database related Errors

SQL-PaLM

Ruoxi Sun, Sercan Ö. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, Tomas Pfister:SQLPaLM: Improved Large Language Model Adaptation for Text-to-SQL. CoRR abs/2306.00739 (2023)

Existing Taxonomies for NL2SQL Errors

- Schema Linking
- JOIN
- GROUP BY
- Queries with Nesting and Set Operations
- Invalid SQL
- Miscellaneous



DIN-SQL

Mohammadreza Pourreza, Davood Rafiei:
DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. NeurIPS 2023

Existing Taxonomies for NL2SQL Errors

Limitations:

- Designed for specific datasets
- Limited scalability
- Due to inconsistent taxonomies, the annotated data cannot be shared or compared between different studies

Therefore, it is imperative to develop **a standardized and effective taxonomy** for analyzing NL2SQL Errors

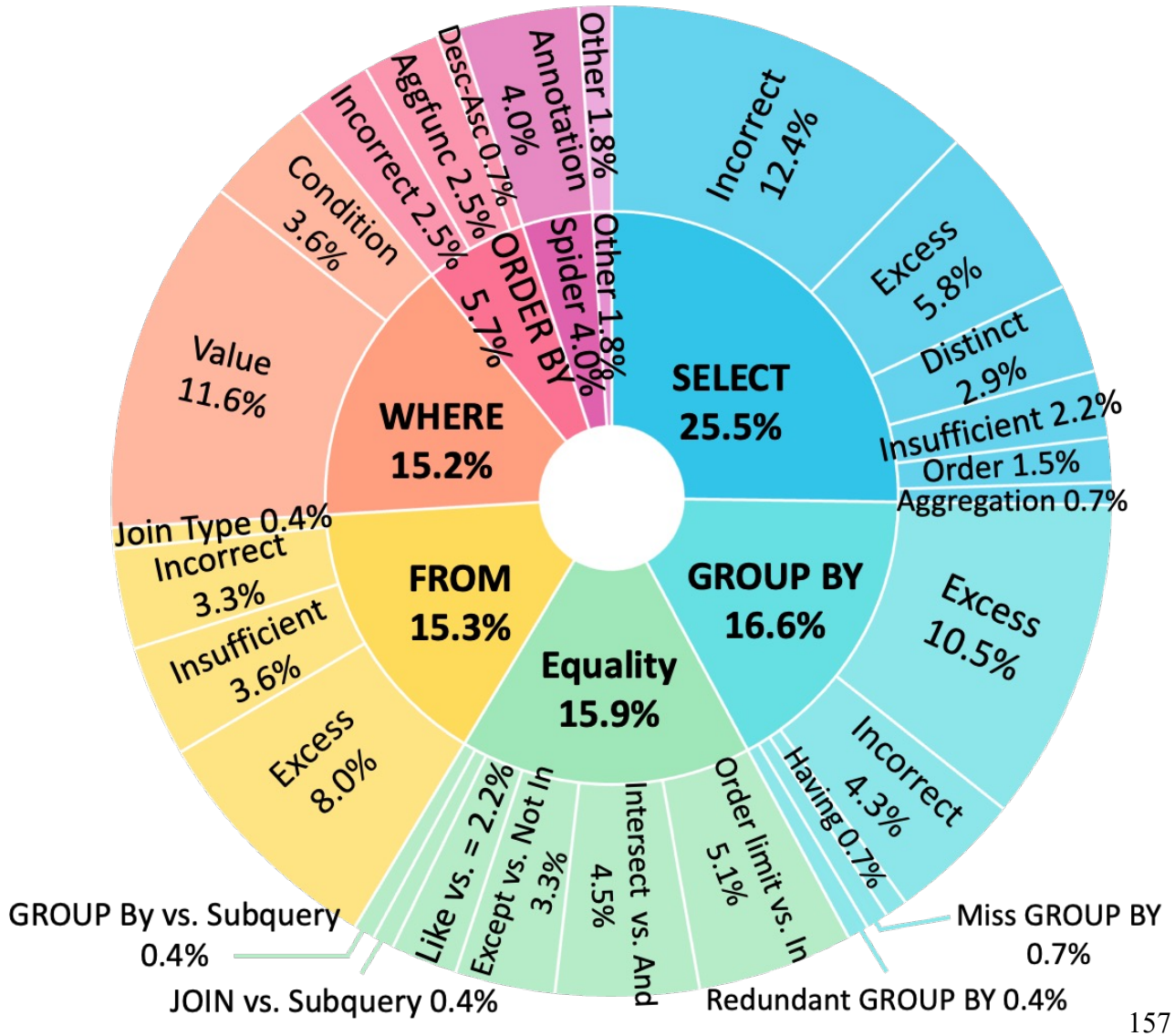
Principles for NL2SQL Errors Taxonomy

- **Comprehensiveness:** The taxonomy should encompass all potential errors that could occur during the NL2SQL conversion process.
- **Mutual Exclusivity:** Each error type should be clearly distinct with no overlap, to avoid ambiguity in error classification.
- **Extensibility:** The taxonomy should be adaptable to incorporate new error types as NL2SQL technologies and methodologies evolve.
- **Practicality:** The taxonomy should be practical and applicable in real-world settings, aiding developers in diagnosing and correcting errors effectively

Our Two-level Taxonomy

- **Error Localization**: Focus on identifying the specific parts of the SQL where errors occur

- **Cause of Error**: Focus on understanding why the model is wrong when generating SQL.



Outline

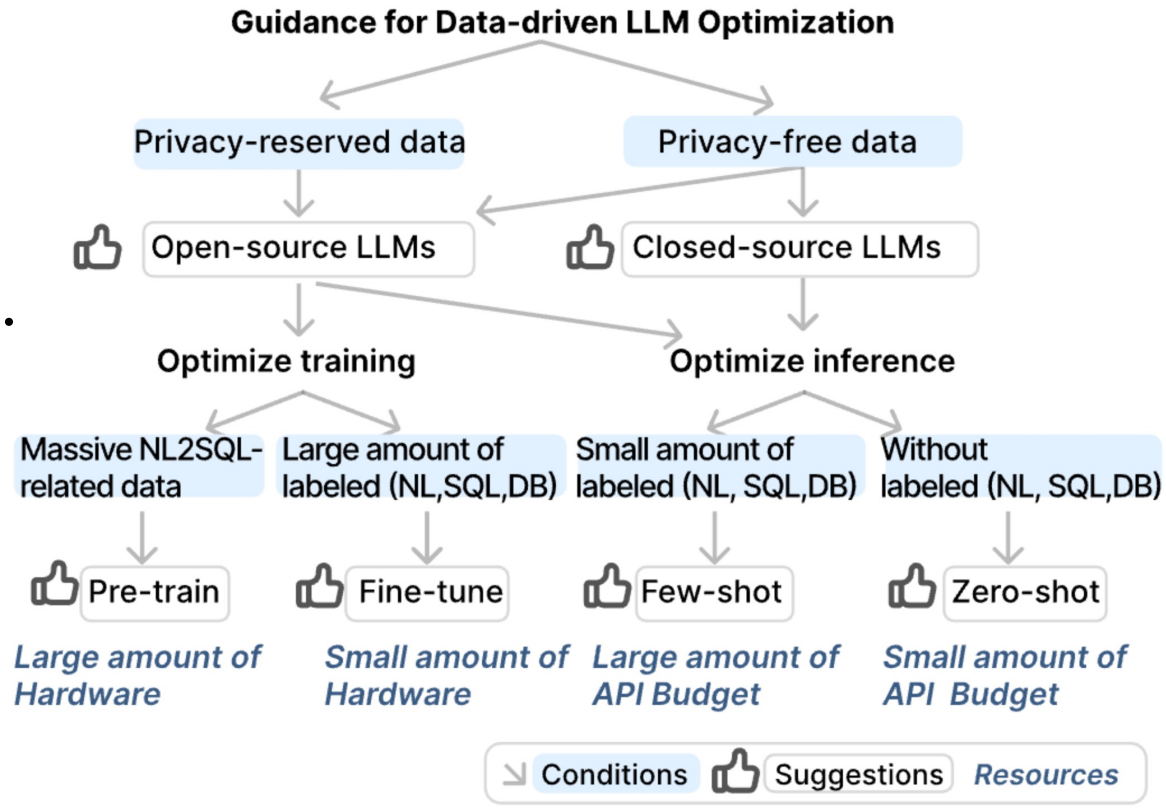
- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
  - ⭐ A Data-Driven Roadmap of Optimizing LLMs for NL2SQL
    - Decision Flow of Selecting NL2SQL Modules
- Open Problems

A Roadmap of Optimizing LLMs for NL2SQL

Condition 1: Data Privacy.

Condition 2: Training Data Volume.

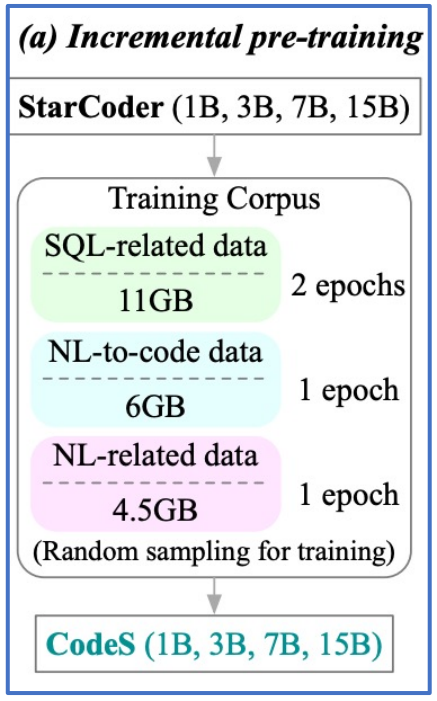
Note that also need consider hardware resources and API costs

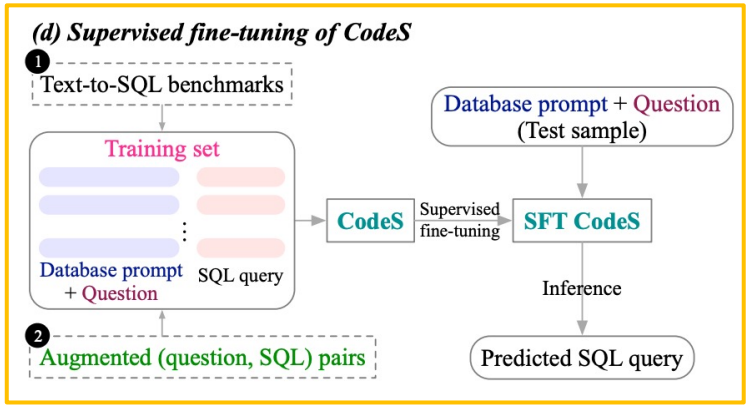


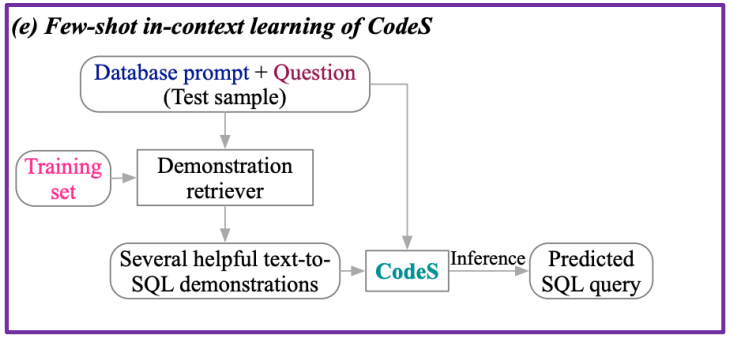
A Roadmap of Optimizing LLMs for NL2SQL

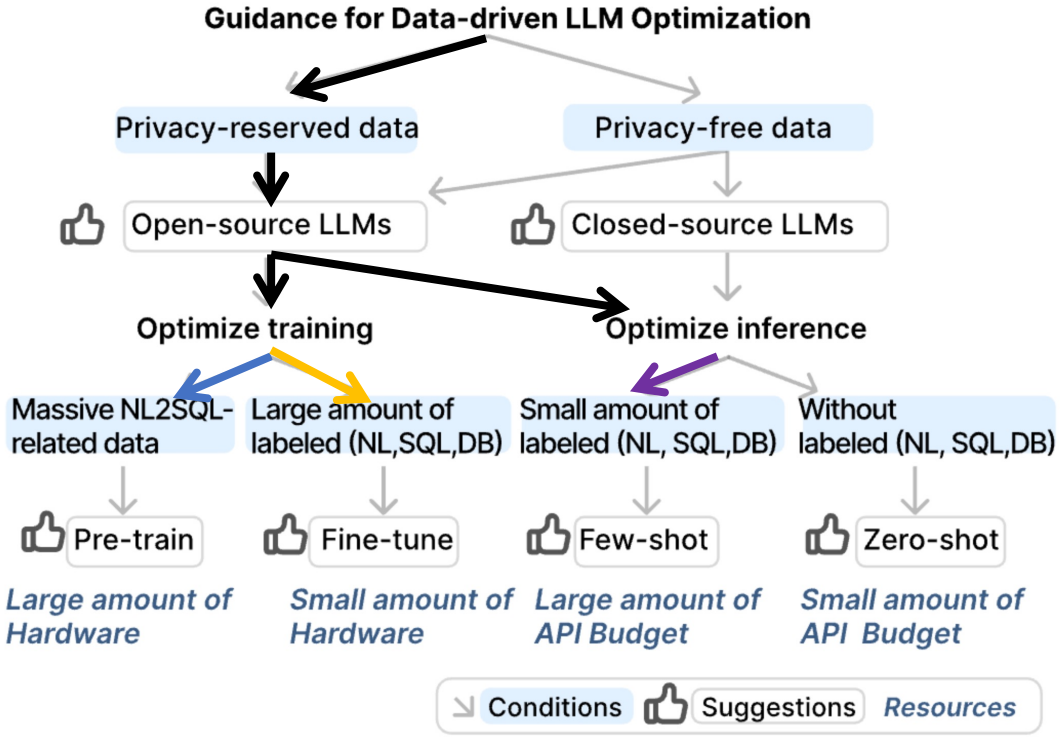
A Case Study

CodeS designs open-source NL2SQL LLM to solve data privacy risks when using GPT-4







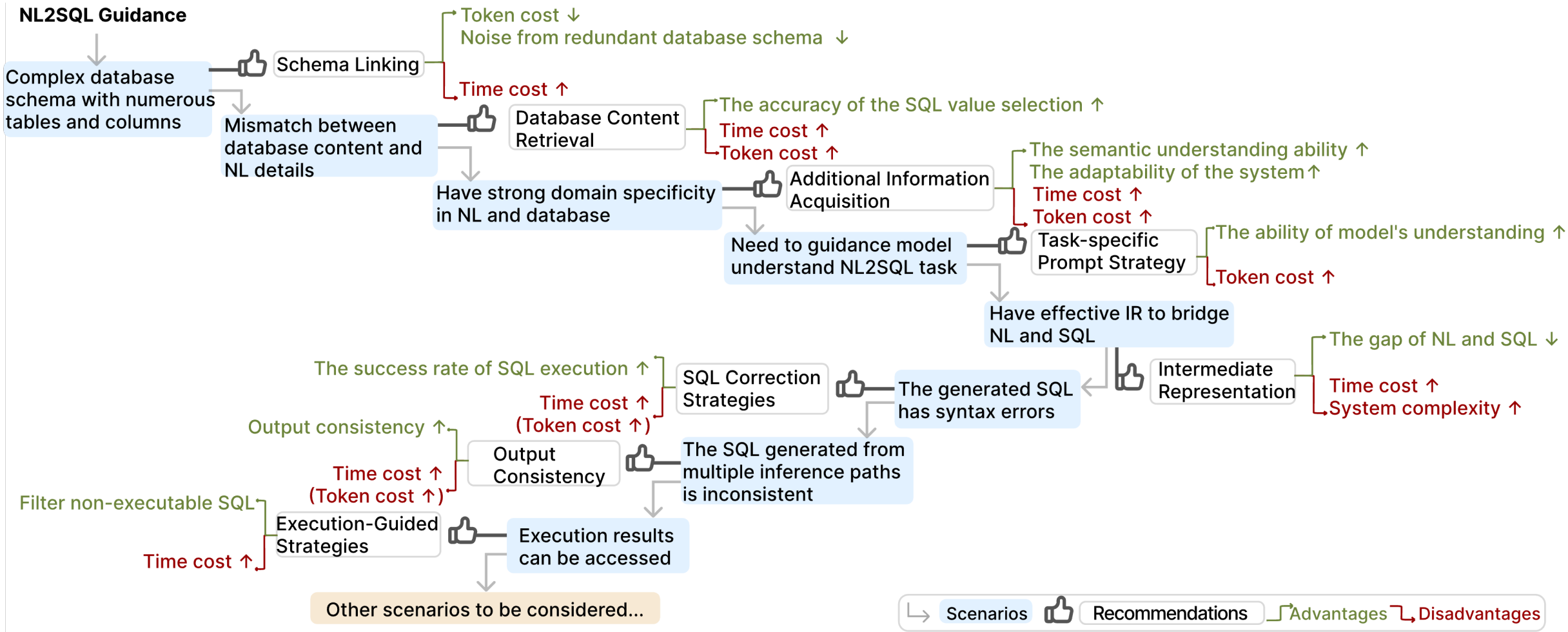


Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, Hong Chen:CodeS: Towards Building Open-source Language Models for Text-to-SQL. Proc. ACM Manag. Data 2(3): 127 (2024)

Outline

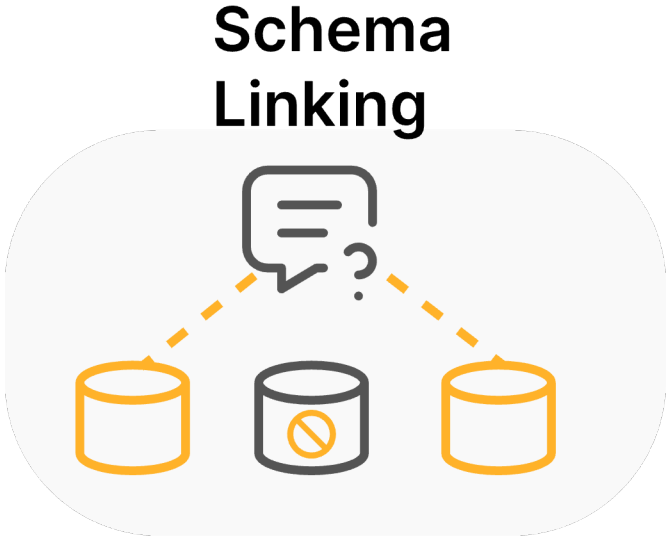
- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
  - A Data-Driven Roadmap of Optimizing LLMs for NL2SQL
  - ⭐ Decision Flow of Selecting NL2SQL Modules
- Open Problems

Decision Flow of Selecting Modules



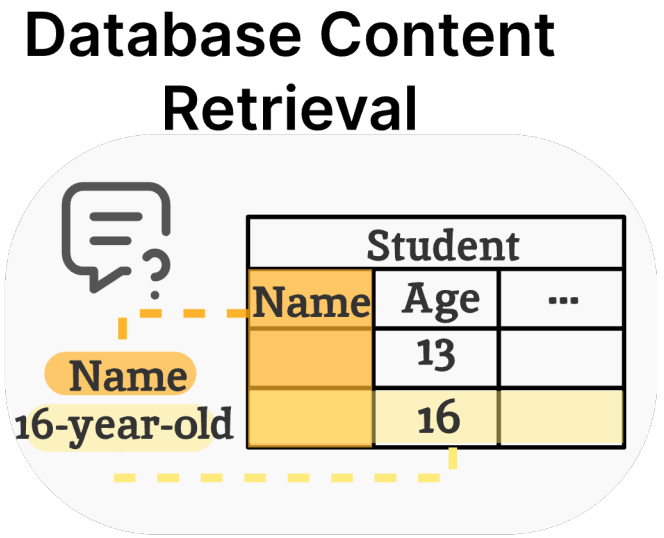
Decision Flow of Selecting Modules

- **Scenario 1:**
  - Complex Database Schema with Numerous Tables and Columns.
- **Recommendation:**
  - Schema Linking.
- **Advantages:**
  - Reduced token costs.
  - Decreased noise from the redundant database schema.
- **Disadvantages:**
  - Increased time costs.



Decision Flow of Selecting Modules

- **Scenario 2:**
  - Mismatch between Database Content and NL.
- **Recommendation:**
  - Database Content Retrieval.
- **Advantages:**
  - Increased the accuracy of SQL value selection.
- **Disadvantages:**
  - Increased time costs.
  - Increased token costs.



Outline

- NL2SQL Problem and Background
- Language Model-Powered NL2SQL Solutions
- NL2SQL Benchmarks
- Evaluation and Error Analysis
- Practical Guidance for Developing NL2SQL Solutions
- ⭐ Open Problems

Open Problem- Open NL2SQL Problem

- **Problem & Goal:**
  - Open NL2SQL Problem: Querying Multiple Databases.
- **Motivation:**
  - In many real-world scenarios, such as government open data platforms, answering complex questions often requires querying multiple databases and aggregating results.
- **Key Idea:**
  - Database Retrieval
  - Handling Heterogeneous Schemas
  - Answer Aggregation
  - Domain Adaptation
  - Scalability and Efficiency
  - Evaluating and Benchmarking

Open Problem- Cost-effective NL2SQL Methods

- **Problem & Goal:**
  - Cost-effective NL2SQL Methods: Balancing Performance and Costs.
- **Motivation:**
  - LLM-based NL2SQL methods show potential but face challenges due to high token consumption, which impacts both costs and inference times. Conversely, PLM-based methods excel at complex SQL query processing and schema interpretation.
- **Key Idea:**
  - Modular NL2SQL Solutions
  - Multi-agent Framework

Open Problem- Trustworthy NL2SQL Solutions

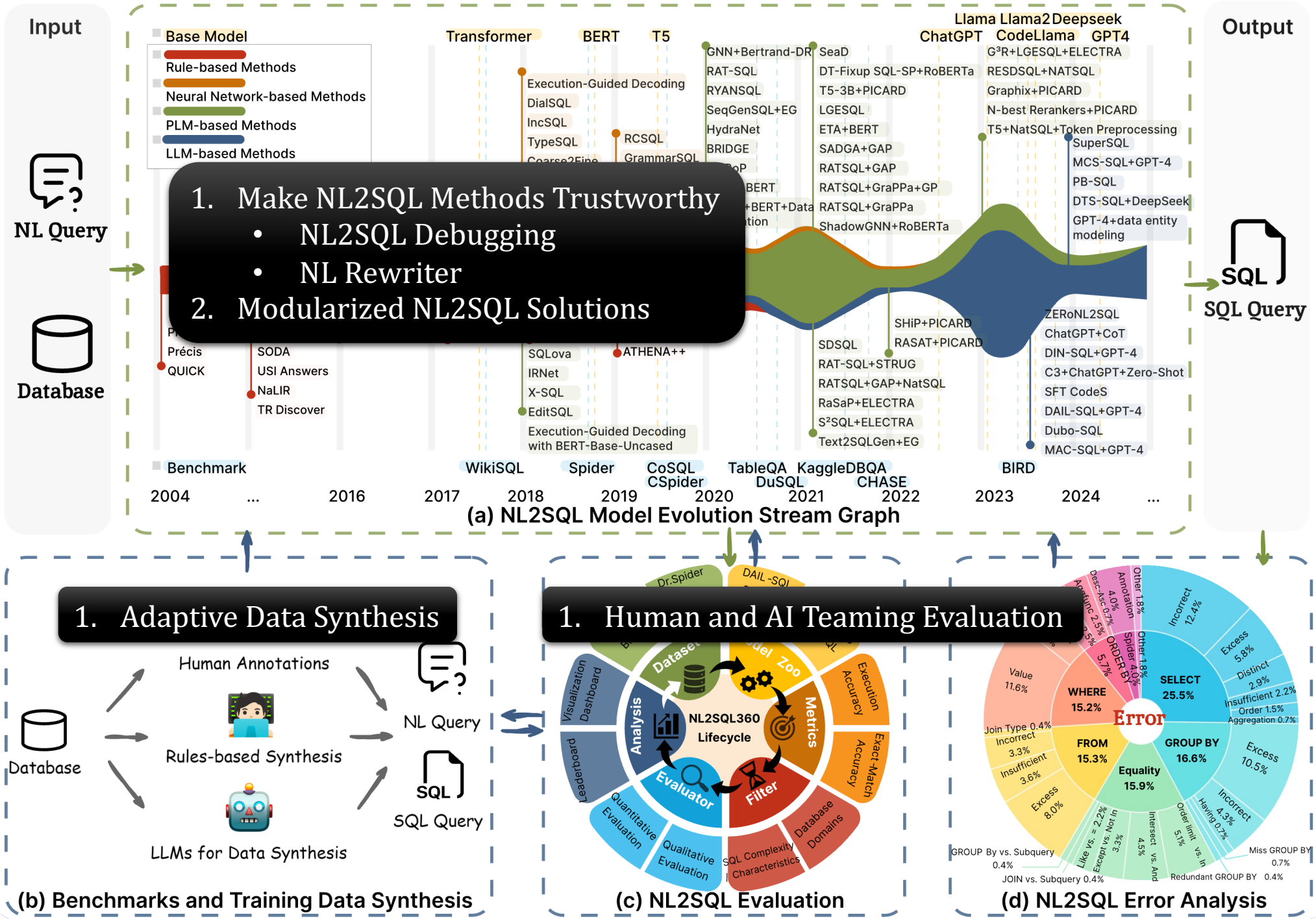
- **Problem & Goal:**
  - Trustworthy NL2SQL Solutions: Enhancing Accuracy and Reliability.
- **Motivation:**
  - Ensuring the accuracy and reliability of generated SQL queries is essential to reduce risks and minimize manual intervention.
- **Key Idea:**
  - Interpreting NL2SQL Solutions
  - Multi-agent Framework
  - NL2SQL Debugging Tools

Open Problem- NL2SQL with Ambiguous and Unspecified NL Queries

- **Problem & Goal:**
  - NL2SQL with Ambiguous and Unspecified NL Queries: Addressing Vague Inputs.
- **Motivation:**
  - Ambiguous or unspecified queries are common in real-world scenarios and pose challenges for NL2SQL systems.
- **Key Idea:**
  - NL Query Rewriter.
  - NL Query Auto-completion.
  - Training with Ambiguous Queries.

Open Problem- Adaptive Training Data Synthesis

- **Problem:**
  - Adaptive Training Data Synthesis: Improving Model Generalization.
- **Motivation:**
  - Learning-based NL2SQL methods often struggle with new domains and require high-quality, diverse training data for effective performance.
- **Key Idea:**
  - Adaptive Data Synthesis.
  - Incorporating Feedback.


Input
NL Query
Database
Output
SQL
SQL Query
Base Model
Rule-based Methods
Neural Network-based Methods
PLM-based Methods
LLM-based Methods
Transformer
BERT
T5
Llama Llama2 Deepseek
ChatGPT
CodeLlama
GPT4
1. Make NL2SQL Methods Trustworthy
• NL2SQL Debugging
• NL Rewriter
2. Modularized NL2SQL Solutions
Execution-Guided Decoding
DialSQL
IncSQL
TypeSQL
RCSQL
GrammarSQL
GNN+Bertrand-DR
RAT-SQL
RYANSQL
SeqGenSQL+EG
HydraNet
BRIDGE
SeaD
DT-Fixup SQL-SP+RoBERTa
T5-3B+PICARD
LGESQL
ETA+BERT
SADGA+GAP
RATSQL+GAP
RATSQL+GraPPa+GP
RATSQL+GraPPa
ShadowGNN+RoBERTa
G³R+LGESQL+ELECTRA
RESDSQL+NATSQL
Graphix+PICARD
N-best Rerankers+PICARD
T5+NatSQL+Token Preprocessing
SuperSQL
MCS-SQL+GPT-4
PB-SQL
DTS-SQL+DeepSeek
GPT-4+data entity modeling
Précis
QUICK
SODA
USI Answers
NaLIR
TR Discover
SQLova
IRNet
X-SQL
EditSQL
Execution-Guided Decoding with BERT-Base-Uncased
ATHENA++
SDSQL
RAT-SQL+STRUG
RATSQL+GAP+NatSQL
RaSaP+ELECTRA
S²SQL+ELECTRA
Text2SQLGen+EG
SHiP+PICARD
RASAT+PICARD
ZERoNL2SQL
ChatGPT+CoT
DIN-SQL+GPT-4
C3+ChatGPT+Zero-Shot
SFT CodeS
DAIL-SQL+GPT-4
Dubo-SQL
MAC-SQL+GPT-4
Benchmark
WikiSQL
Spider
CoSQL
CSpider
TableQA
DuSQL
KaggleDBQA
CHASE
BIRD
2004
...
2016
2017
2018
2019
2020
2021
2022
2023
2024
...
(a) NL2SQL Model Evolution Stream Graph
1. Adaptive Data Synthesis
Human Annotations
Database
Rules-based Synthesis
LLMs for Data Synthesis
NL Query
SQL
SQL Query
(b) Benchmarks and Training Data Synthesis
1. Human and AI Teaming Evaluation
Dr.Spider
DAIL -SQL
Dataset
Model Zoo
Visualization Dashboard
Execution Accuracy
Analysis
NL2SQL360 Lifecycle
Metrics
Exact-Match Accuracy
Leaderboard
Evaluator
Filter
Quantitative Evaluation
Qualitative Evaluation
SQL Complexity Characteristics
Database Domains
(c) NL2SQL Evaluation
Error
SELECT 25.5%
WHERE 15.2%
FROM 15.3%
GROUP BY 16.6%
ORDER BY 5.7%
Equality 15.9%
Incorrect 12.4%
Excess 5.8%
Distinct 2.9%
Insufficient 2.2%
Order 1.5%
Aggregation 0.7%
Excess 10.5%
Incorrect 4.3%
Having 0.7%
Miss GROUP BY 0.7%
Redundant GROUP BY 0.4%
Order limit 5.1%
vs. In
Intersect vs. And 4.5%
Except vs. Not In 3.3%
Like vs. = 2.2%
GROUP By vs. Subquery 0.4%
JOIN vs. Subquery 0.4%
Excess 8.0%
Insufficient 3.6%
Incorrect 3.3%
Join Type 0.4%
Value 11.6%
Annotation
Other 1.8%
Spider 4.0%
Desc-Asc 0.7%
Agg-func 2.5%
(d) NL2SQL Error Analysis

Thanks!

Yuyu LUO (骆昱宇)
Assistant Professor, HKUST (Guangzhou)/HKUST

yuyuluo@hkust-gz.edu.cn

https://arxiv.org/abs/2408.05109

https://github.com/HKUSTDial/NL2SQL_Handbook