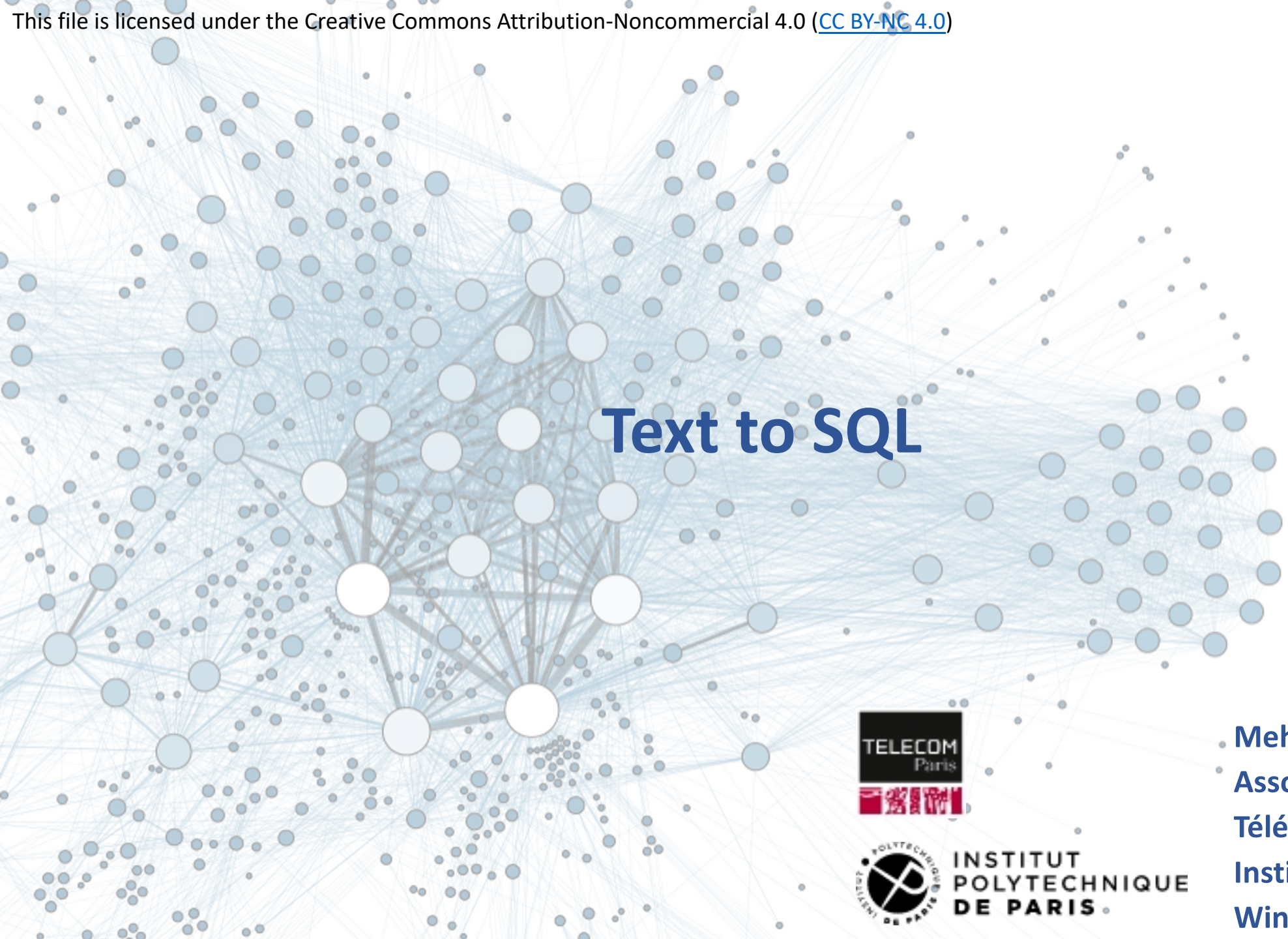


Text to SQL

Mehwish Alam
Associate Professor
Télécom Paris
Institut Polytechnique de Paris
Winter Semester 2024-2025

Outline

- What are the traditional approaches for T2S?
- Masked Language Models for T2S (TaBERT)
- Large Language Models for T2S
  - Preprocessing
  - Inference
  - Post-processing
  - Fine-tuning

Outline

- What are the traditional approaches for T2S?
- Masked Language Models for T2S (TaBERT)
- Large Language Models for T2S
  - Preprocessing
  - Inference
  - Post-processing
  - Fine-tuning

Parsing Approaches for T2S

Single-turn Parsing (context independent).

- Given a NL question Q and the corresponding database schema $S = < T, C >$, the goal is to generate a SQL query Y.
  - Question $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ is a sequence of $|Q|$ tokens.
  - The database schema consists of $|\mathcal{T}|$ tables $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$ and $|C|$ columns $C = \{ c_1, c_2, \dots, c_{|C|}\}$.

- Each table t_i is described by its name that contains multiple words $\left[t_{i,1}, t_{i,2}, \dots, t_{i,|t_i|}\right]$.

- Each table $c_j^{t_i}$ in the table t_i is represented by words (a phrase) $\left[c_{j,1}^{t_i}, c_{j,2}^{t_i}, \dots, c_{j,|c_j^{t_i}|}^{t_i}\right]$

- The whole input is given as $X = < Q, S >$

Source: https://arxiv.org/pdf/2208.13629

Parsing Approaches for T2S

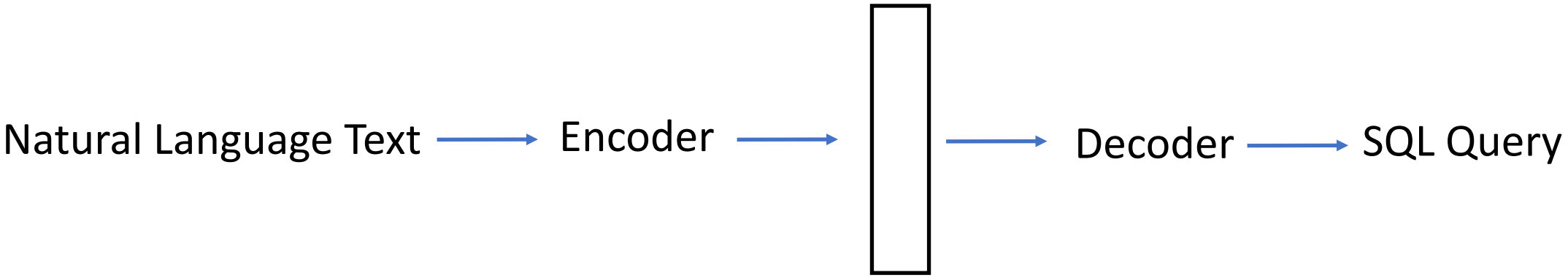
Multi-turn Parsing (context dependent)

- A sequence of NL questions to the corresponding SQL queries
- NL questions may contain ellipsis and anaphora
- Let $U = \{U_1, \ldots, U_T\}$ denote a sequence of utterances with T turns
- $U_t = (X_t, Y_t)$ represents the $t - th$ utterance
- U_t is the combination of an NL question X_i and a SQL query Y_i
- S = corresponding database schema
- At the $t - th$ turn, produce the SQL query Y_t conditioned on the current NL question X_t
- The historical utterances $\{U_i\}_{i=1}^{t-1}$, and the database schema S

Source: https://openreview.net/pdf?id=IZnrCGF9WI

Deep Learning based Approaches – Single Turn

Seq2Seq task



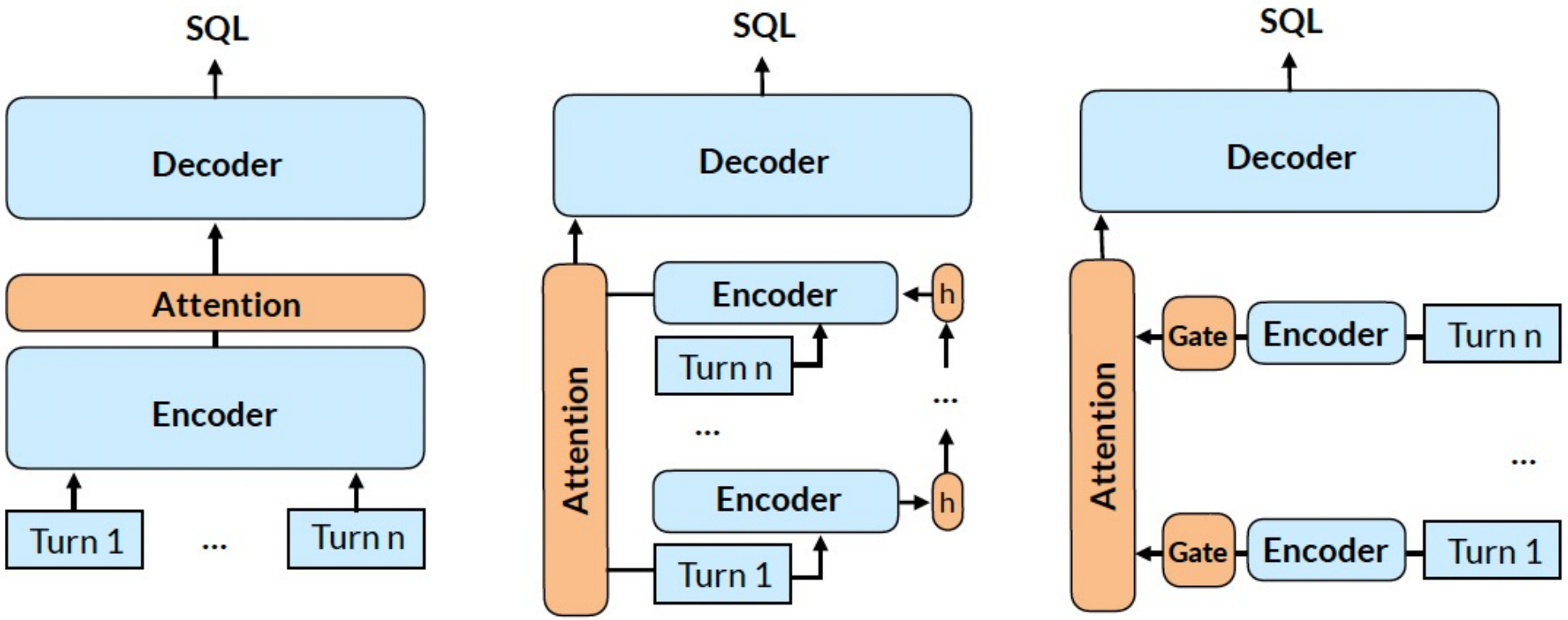
Encoders.

- Schema Linking – SL
- LSTM
- Transformers
- Graph Neural Networks

Decoders.

- LSTM
- Transformers
- Grammar
- Domain Specific Language - DSL
- Constrained Decoding
- Re-Ranking

Deep Learning based Approaches – Multi-turn



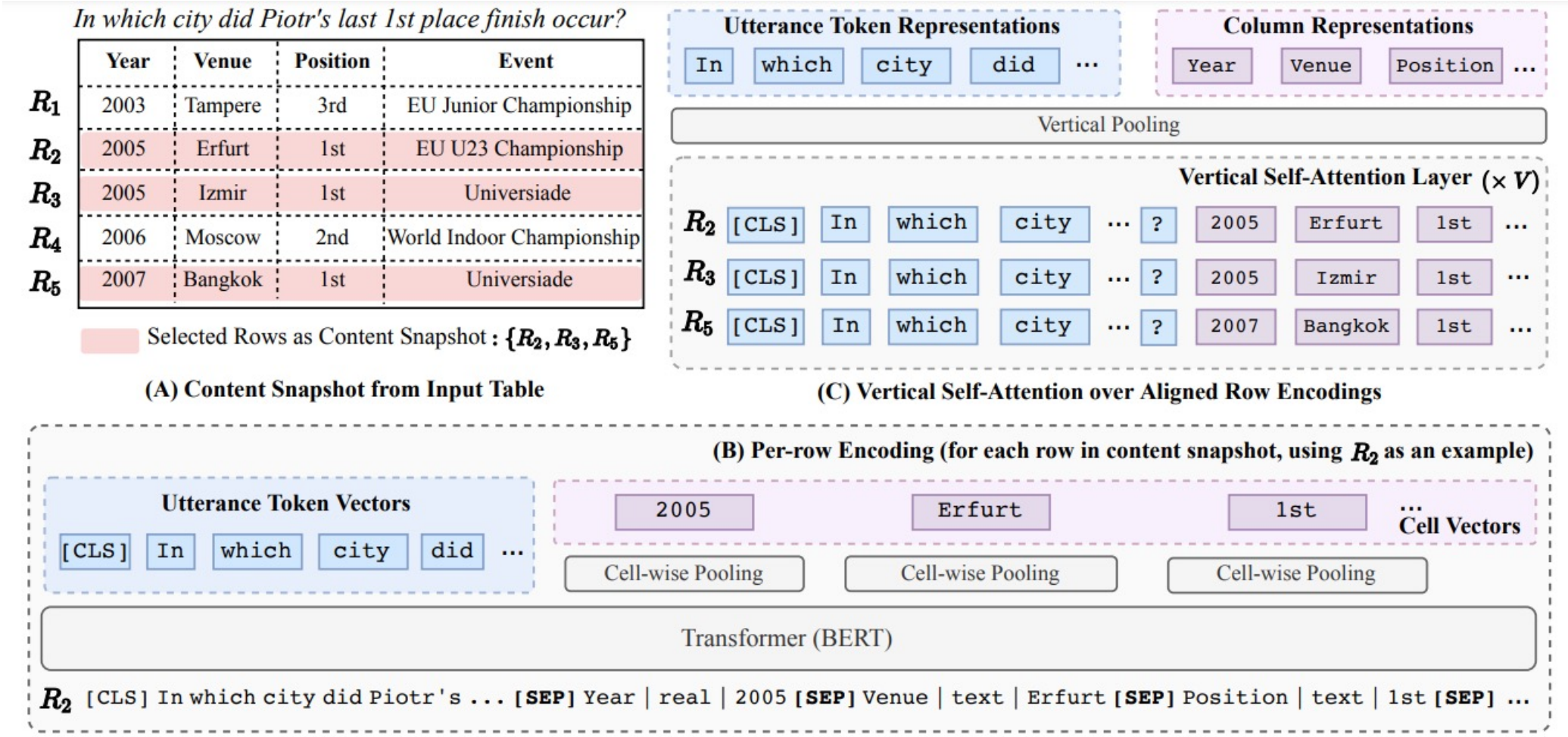
(a) Concat (b) Turn (c) Gate

Outline

- What are the traditional approaches for T2S?
- Masked Language Models for T2S (TaBERT)
- Large Language Models for T2S
  - Preprocessing
  - Inference
  - Post-processing
  - Fine-tuning

Masked Language Models for T2S

TaBERT – Learning Joint Representations over Textual and Tabular Data



Source: https://aclanthology.org/2020.acl-main.745.pdf

Masked Language Models for T2S - TaBERT

Computing Representations for NL Utterances and Table Schemas

- Content Snapshot
- Row Linearization
- Vertical Self-Attention Mechanism
- Utterance and Column Representations

Source: https://aclanthology.org/2020.acl-main.745.pdf

Masked Language Models for T2S - TaBERT

Computing Representations for NL Utterances and Table Schemas

➢ Content Snapshot

- TaBERT uses the table contents in addition to column names
- Ambiguity in column names
  - e.g., “Venue” can be ambiguous “City” may be more representative
- **Drawback.** Large number of rows in the table
- **Solution.**
  - Content Snapshot containing only relevant rows to input utterance
  - Calculate content-sensitive column representations from cell values
- **Strategy.**
  - Top-k based on relevance between utterance and rows
  - Highest n-gram overlap ratio
  - For k=1, synthetic row by selecting the cell values for each column with highest n-gram ratio

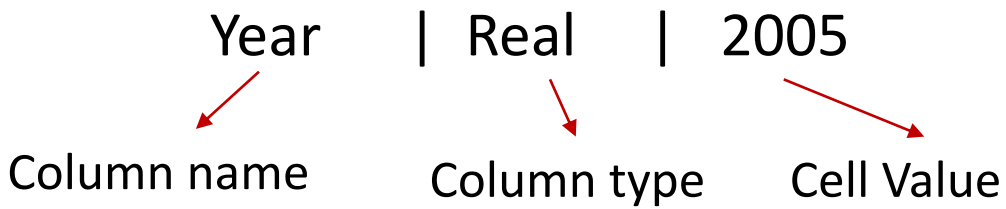
Source: https://aclanthology.org/2020.acl-main.745.pdf

Masked Language Models for T2S - TaBERT

Computing Representations for NL Utterances and Table Schemas

➢ Row Linearization

- Creates a linearized sequence of each row in content snapshot
- Input to the transformer
- A concatenation of the utterance, columns, and their cell values
- Each cell is represented as follows:



[CLS] In which city did Piotr's ... [SEP] Year | real | 2005 [SEP] Venue | text | Erfurt [SEP] Position | text | 1st [SEP]

Source: https://aclanthology.org/2020.acl-main.745.pdf

Masked Language Models for T2S - TaBERT

Computing Representations for NL Utterances and Table Schemas

➢ Vertical Self-Attention Mechanism

- Allows for information flow across cell representations
- A fixed length initial vector for each cell at position $< i, j >$ which is given by mean pooling over the sequence of the Transformer's output vectors
- The sequence of word vectors for the NL utterance are concatenated with the cell vectors as initial inputs to the vertical attention layer

Source: https://aclanthology.org/2020.acl-main.745.pdf

Masked Language Models for T2S - TaBERT

Computing Representations for NL Utterances and Table Schemas

➢ Utterance and Column Representation

- A representation $\boldsymbol{c}_j$ is computed for each column c_j by mean-pooling over its vertically aligned cell vectors
- A representation for each utterance token, x_j, is computed similarly over the vertically aligned token vectors

Source: https://aclanthology.org/2020.acl-main.745.pdf

Masked Language Models for T2S - TaBERT

Unsupervised Learning Objective

- **Masked Column Prediction (MCP)** objective encourages the model to recover the names and data types of masked columns
- **Cell Value Recovery (CVR)** objective to ensure information of representative cell values in content snapshots is retained after additional layers of vertical self-attention

Source: https://aclanthology.org/2020.acl-main.745.pdf

Outline

- What are the traditional approaches for T2S?
- Masked Language Models for T2S (TaBERT)
- Large Language Models for T2S
  - Preprocessing
  - Inference
  - Post-processing
  - Fine-tuning

LLMs for Text2SQL - Preprocessing

Question Representation.

- Natural Language Problem
- Related information from databases

Layouts.

```
### SQLite SQL tables, with their
    properties
#
# schools(SchID, city)
# teachers(TechID, SchID)
# students(StuID, SchID, TechID)
# course(CourseID, SchID)
# grade(StuID, CourseID, score)
#
#
### {Question}
SELECT
```

OpenAI Template

```
### SQLite SQL tables, with their
    properties
CREATE TABLE schools(
    SchID int primay key,
    city text
);
CREATE TABLE teachers(
    TechID int primay key,
    SchID int references schools(TechID)
);
### {Question}
SELECT
```

Create Table Template

LLMs for Text2SQL - Preprocessing

- Add sample data to the prompt, i.e., the SQL query
- Combination of layout and sample data

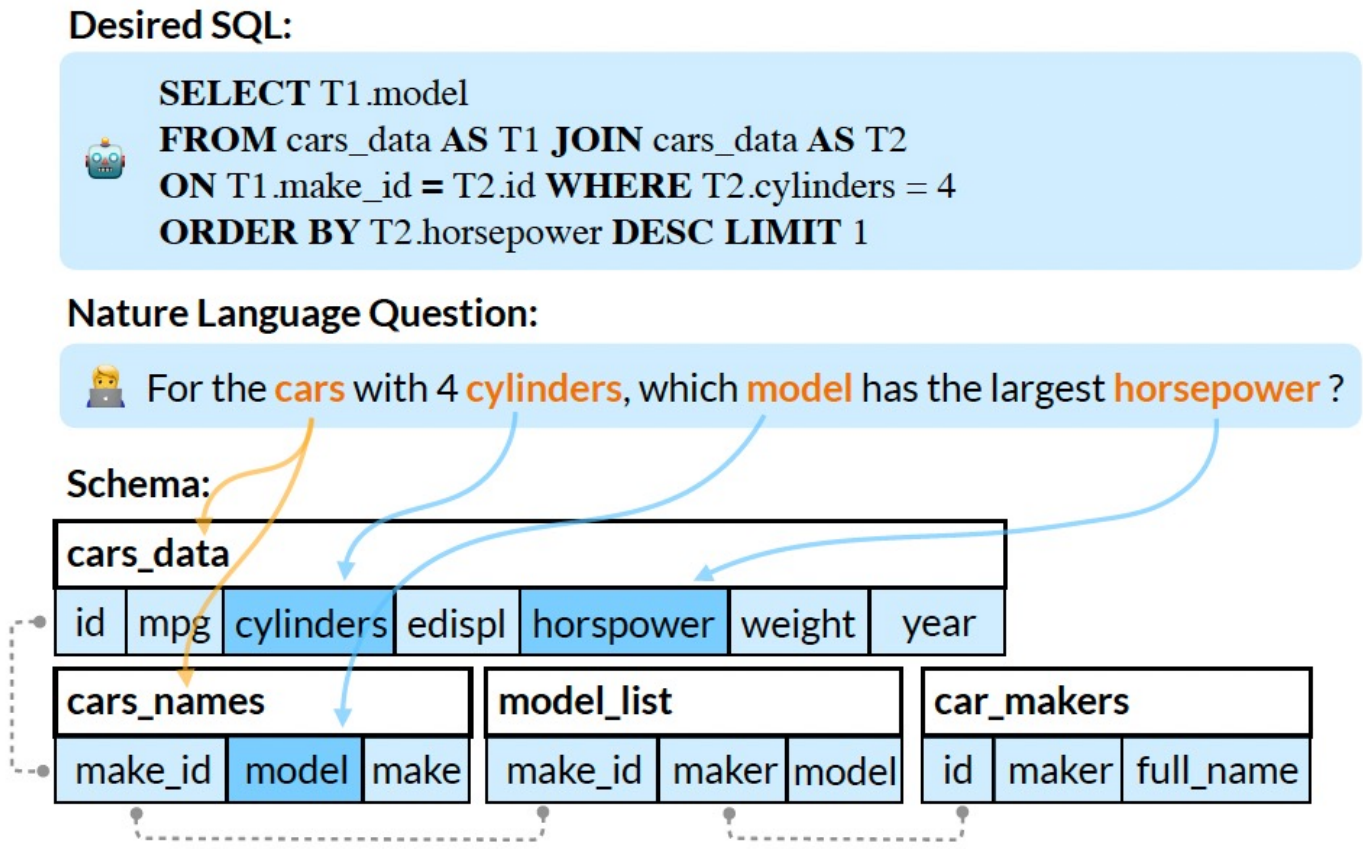
Observations on Question Formulations.

- **Layout.**
  - Significant drop in performance when changing from layout to unstructured format
- **Sample Data.**
  - With each of the layout sample data is plug-able and beneficial
  - More sample data is not always better
  - More sample data could be risky in case of context length limitation
- **Primary & Foreign Keys.**
  - Removing foreign keys significantly decreases the performance
  - SQLfuse adds primary keys as a second contributor to performance gain
  - Some LLMs see drop while adding the FKs

LLMs for Text2SQL - Preprocessing

Schema Linking.

Specify the tables and columns in the database that correspond to the phrases in the given query



LLMs for Text2SQL - Preprocessing

Schema Linking. Divide & Prompt. Clause by Clause DnP (CC-DnP).

Input

These are tables(columns):
Products(Product_ID, Product_Type_Code…),
…
This is the text: Show the names of products that are in at least two events.
Translate the text to SQL query base on tables(columns).
1. Write the FROM clause: FROM Products AS T1 JOIN Products_in_Events AS T2 ON T1.Product_ID = T2.Product_ID ,
2. Write the GROUP BY clause: GROUP BY T1.Product_Name,
3. Writer the HAVING clause: HAVING COUNT(*) >= 2,
4. Write the SELECT clause: SELECT T1.Product_Name
5. Final SQL query: SELECT T1.Product_Name FROM Products AS T1 JOIN Products_in_Events AS T2 ON T1.Product_ID = T2.Product_ID GROUP BY T1.Product_Name HAVING COUNT(*) >= 2;

These are tables(columns):
stadium(stadium_id, location…),
singer(singer_id, name, country…),
…
This is the text: Show the name and the release year of the song by the youngest singer.
Translate the text to SQL query base on tables and columns.

Model output

1. Writer the FROM clause: FROM singer,
2. Writer the subquery: SELECT MIN(age) FROM singer,
3. Writer the WHERE clause: WHERE age = (SELECT MIN(age) FROM singer),
4. Writer the SELECT clause: SELECT song_name, song_release_year,
5. The final SQL is:

```
SELECT song_name, song_release_year
FROM singer
WHERE age = (SELECT MIN(age) FROM singer)
```

LLMs for Text2SQL - Preprocessing

Divide & Prompt. Schema Linking DnP (SL-DnP).

Input

These are tables(columns):
Products(Product_ID, Product_Type_Code…),
…
This is the text: Show the names of products that are in at least two events.
Translate the text to SQL query base on tables(columns).
1. Identify the relevant tables:
Products(Product_ID, Product_Type_Code, Product_Name, Product_Price),
Products_in_Events(Product_in_Event_ID, Event_ID, Product_ID)
2. Final SQL query: SELECT T1.Product_Name FROM Products AS T1 JOIN Products_in_Events AS T2 ON T1.Product_ID = T2.Product_ID GROUP BY T1.Product_Name HAVING COUNT(*) >= 2;

These are tables(columns):
stadium(stadium_id, location…),
singer(singer_id, name, country…),
…
This is the text: Show the name and the release year of the song by the youngest singer.
Translate the text to SQL query base on tables and columns.

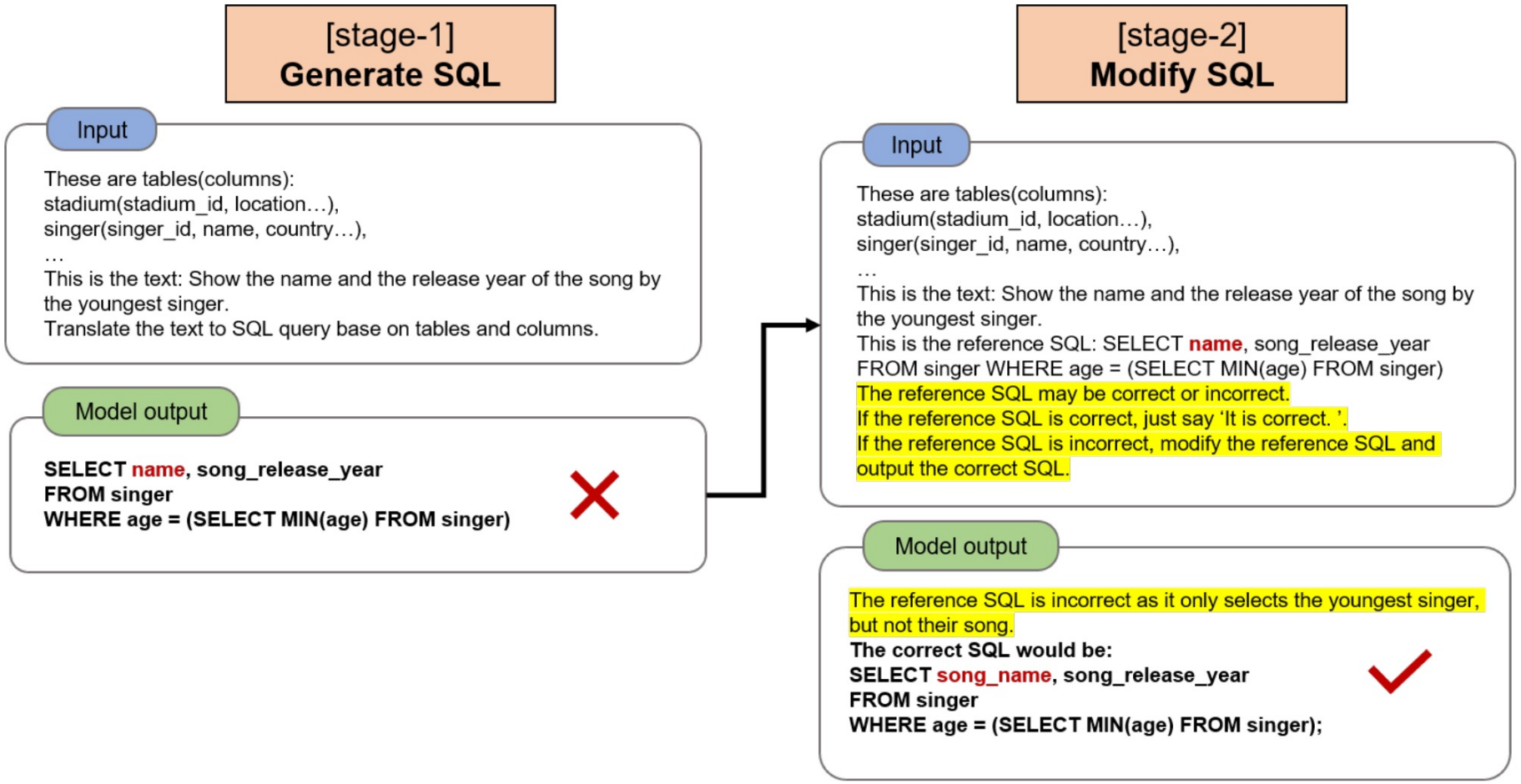
Model output

1. identify the relevant tables: singer(singer_id, name, country, song_name, song_release_year, age, is_male).
2. The final SQL is:

```
SELECT song_name, song_release_year
FROM singer
WHERE age = (SELECT MIN(age) FROM singer)
```

LLMs for Text2SQL - Preprocessing

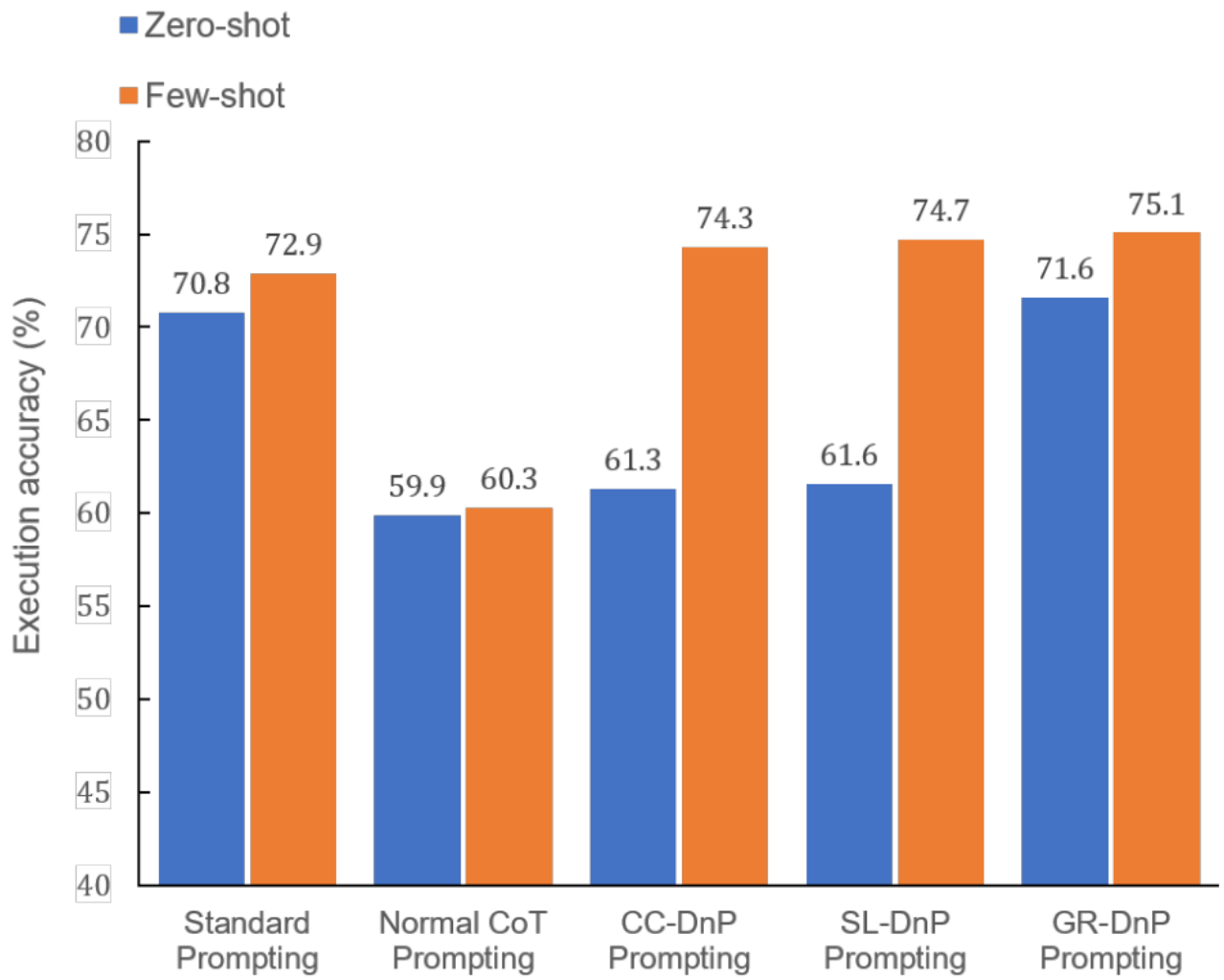
Divide & Prompt. Generate and Refine (GR-DnP).



Source: https://arxiv.org/pdf/2304.11556v1

LLMs for Text2SQL - Preprocessing

Divide & Prompt. Results.



Source: https://aclanthology.org/2023.emnlp-main.327.pdf

LLMs for Text2SQL - Inference

Decomposition. QDecomp

Model Input (API Doc Format)

```
### SQLite SQL tables, with their properties:
#
# allergy_type (allergy, allergytype)
# has_allergy (stuid, allergy)
# student (stuid, lname, fname, age, sex, major, advisor, city_code)
#
### Show first name, last name, age for all female students. Their sex is F.
```

Single-Pass Prompting

A. Chain-of-Thought

```
# This query chooses the records from the student table, followed by a WHERE clause that selects records where the sex column is equal to F. It then selects the fname, lname, and age columns

# Thus, the answer for the question is: Show first name, last name, age for all female students. Their sex is F.
SELECT fname, lname, age FROM student WHERE sex = 'F'
```

Iterative Prompting

B. Least-to-Most Prompting

Problem Reduction:

<Model Input> → **LLM** → # To answer the question "Show first name, last name, age for all female students. Their sex is F.", we need to know: "Show first name, last name, age for all students."

Problem Solving:

<Model Input>

Q: Show first name, last name, age for all students.

→ **LLM** → A: `SELECT fname, lname, age FROM student`

<Model Input>

Q: Show first name, last name, age for all students.
A: `SELECT fname, lname, age FROM student`

Q: Show first name, last name, age for all female students. Their sex is F.

→ **LLM** → A: `SELECT fname, lname, age FROM student WHERE sex = 'F'`

Source: https://aclanthology.org/2023.emnlp-main.327.pdf

LLMs for Text2SQL - Inference

Decomposition. QDecomp

Model Input (API Doc Format)

```
### SQLite SQL tables, with their properties:
#
# allergy_type (allergy, allergytype)
# has_allergy (stuid, allergy)
# student (stuid, lname, fname, age, sex, major, advisor, city_code)
#
### Show first name, last name, age for all female students. Their sex is F.
```

C. QDecomp

1. Show first name, last name, age for all students.
2. Show first name, last name, age for all female students. Their sex is F.

Thus, the answer for the question is: Show first name, last name, age for all female students. Their sex is F.

```
SELECT fname, lname, age FROM student WHERE sex = 'F'
```

D. QDecomp + InterCOL

1. Show first name, last name, age for all students.
SQL table (column): student (fname, lname, age)
2. Show first name, last name, age for all female students. Their sex is F.
SQL table (column): student (sex)

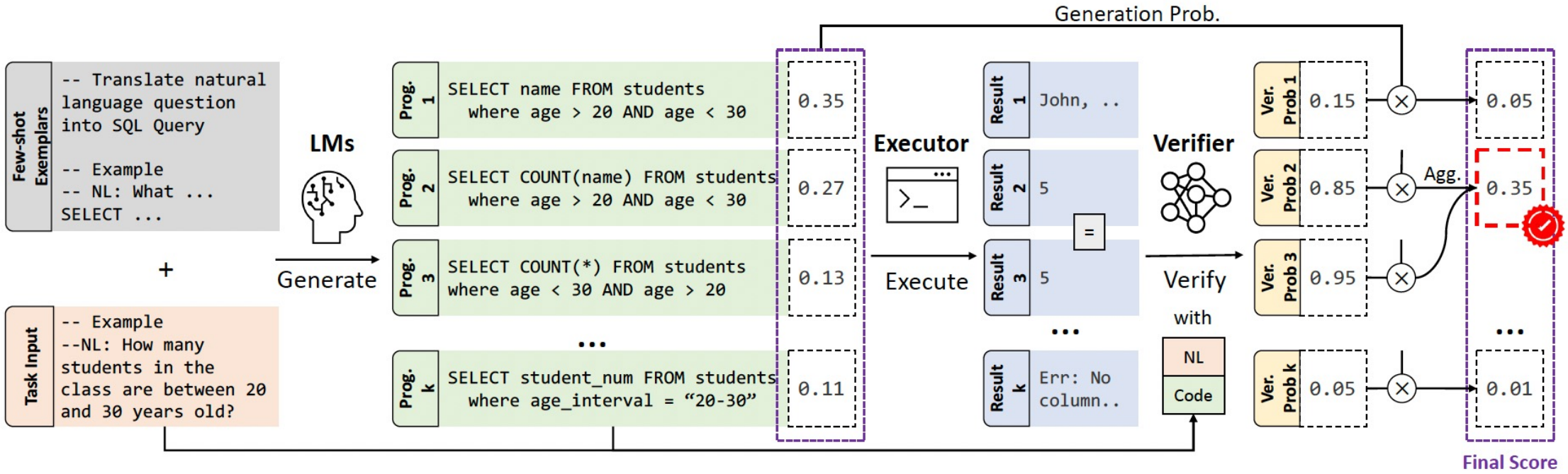
Thus, the answer for the question is: Show first name, last name, age for all female students. Their sex is F.

```
SELECT fname, lname, age FROM student WHERE sex = 'F'
```

Source: https://proceedings.mlr.press/v202/ni23b/ni23b.pdf

LLMs for Text2SQL - Postprocessing

Consistency. Adopts a majority voting strategy, allowing the same LLM to generate multiple answers to the same question by setting a certain temperature.



LLMs for Text2SQL - Postprocessing

LEVER.

Language to Code Generation

$$P_{\boldsymbol{LM}}(y|x) = P(y \mid prompt(x, \{(x_i, y_i)\}_{i<m}))$$

$$\hat{y}_{greedy} \approx \arg max_y \; P_{LM}(y|x)$$

LLMs for Text2SQL - Postprocessing

LEVER.

Reranking. Reranking of Program Candidates

$$\hat{y}_{rerank} = \arg max_{\hat{y} \in S} \; R(x, \hat{y})$$

LLMs for Text2SQL - Postprocessing

LEVER.

Reranking. Program Sampling from Code LLMs

- k programs from $P_{LM}\ (y|x)$ with temperature sampling, i.e, $\{\hat{y}_i\}_{i=1}^{k} \sim P_{LM}\ (y|x)$
- Deduplication is performed since the same program can be sampled more than once

$$S = \{\hat{y}_i\}_{i=1}^{n} \text{ where } n \leq k$$

- Sampling is performed instead of beam search because
  - Beam search for code generation results in worst performance due to degenerated programs
  - Beam search is not available or efficiently implemented for all LLMs used in the work

LLMs for Text2SQL - Postprocessing

LEVER.

Reranking. Verification with Execution

Reranking probability

Joint probability

$$P_R(\hat{y}, v_{=1}|x) = P_{\mathbf{LM}}(\hat{y}|x) \cdot P_\theta(v_{=1}|x, \hat{y}, \mathcal{E}(\hat{y}))$$

generation

Passing the verification

x = Input
$\hat{y}$ = Candidate Program
$\varepsilon(\hat{y})$ = Execution Results

LLMs for Text2SQL - Postprocessing

LEVER.

Reranking. Execution Result Aggregation

$$R(x, \hat{y}) = P_R(\mathcal{E}(\hat{y}), v_{=1}|x) = \sum_{y \in S, \mathcal{E}(y)=\mathcal{E}(\hat{y})} P_R(y, v_{=1}|x)$$

Likelihood of the best output for the input

LLMs for Text2SQL - Postprocessing

LEVER.

Learning the Verifiers. Learning Objective

Negative log-likelihood

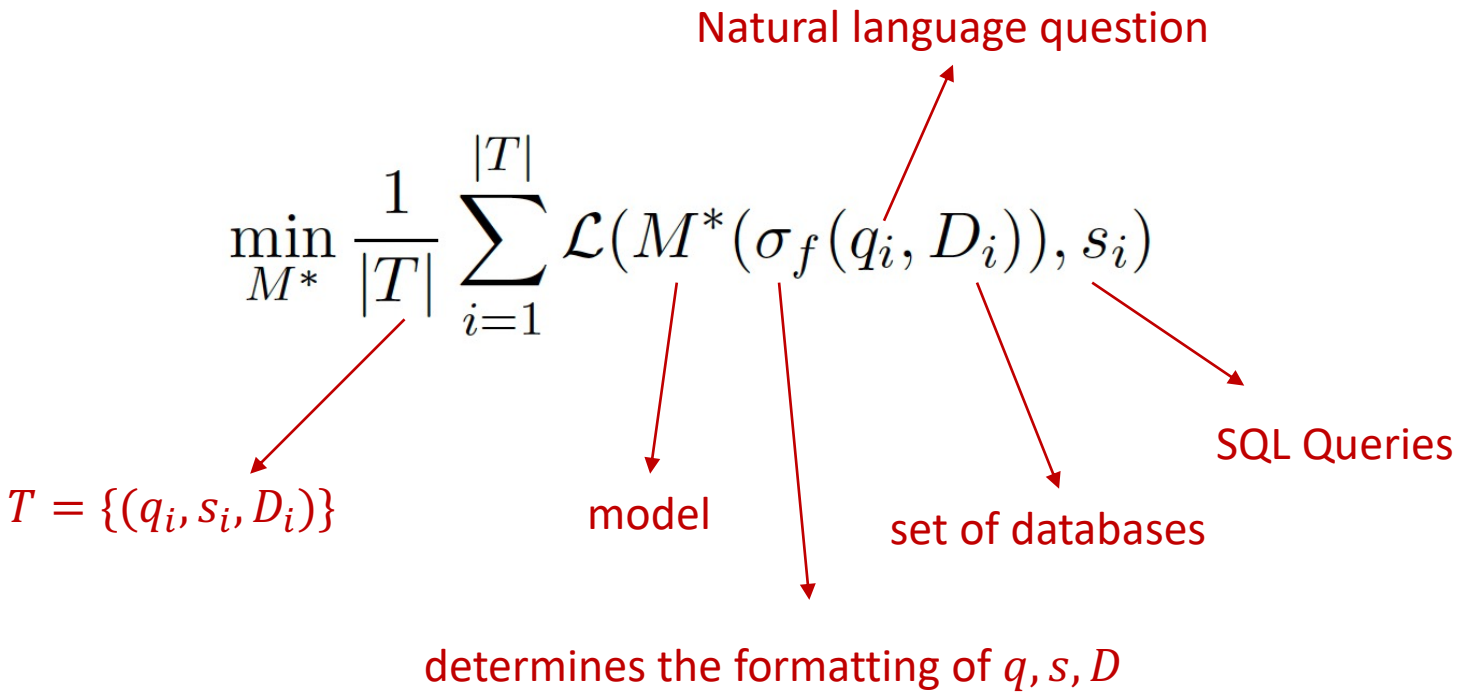
$$\mathcal{L}_\theta(x, S) = -\frac{1}{|S|} \cdot \sum_{\hat{y}_i \in S} \log P_\theta(v_i | x, \hat{y}_i, \hat{z}_i)$$

Source: https://aclanthology.org/2024.findings-emnlp.481.pdf

LLMs for Text2SQL - Finetuning

DTS-SQL. Supervised Fine-tuning

Next token prediction loss

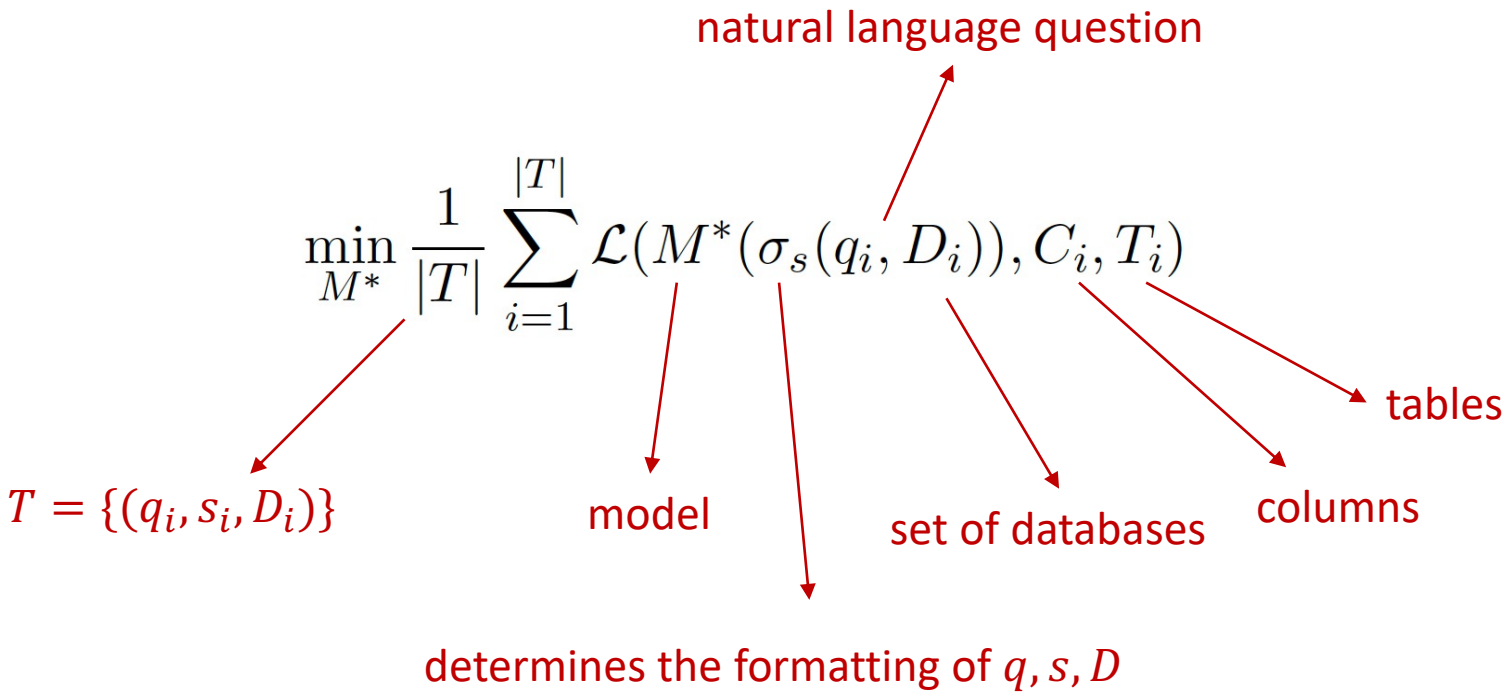


Source: https://aclanthology.org/2024.findings-emnlp.481.pdf

LLMs for Text2SQL - Finetuning

DTS-SQL. Decomposed Supervised Fine-tuning

Schema Linking Fine-Tuning

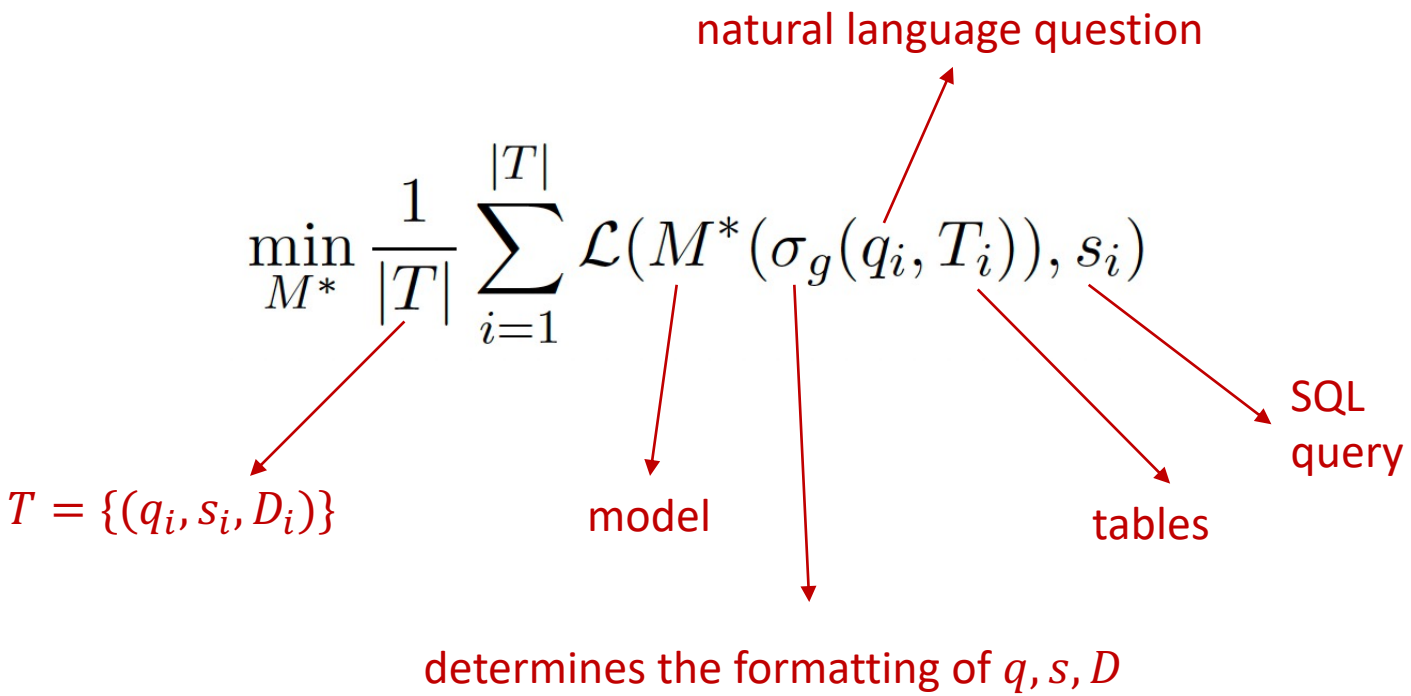


Source: https://aclanthology.org/2024.findings-emnlp.481.pdf

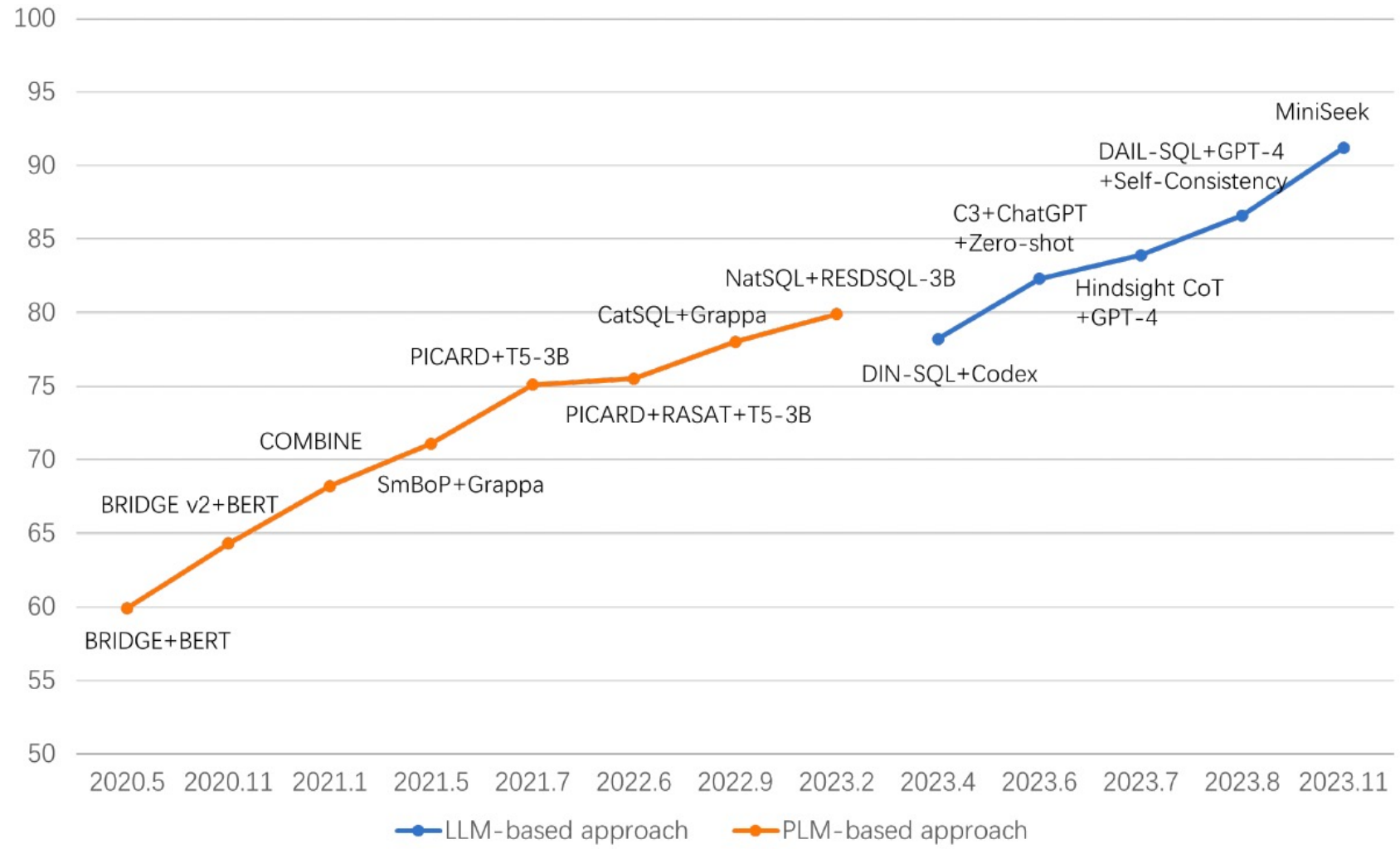
LLMs for Text2SQL - Finetuning

DTS-SQL. Decomposed Supervised Fine-tuning

SQL Generation Fine-Tuning



Accuracy for T2S over time



Benchmark Datasets for T2S - Traditional

WikiSQL.

- A large crowd-sourced dataset for developing NL interfaces for relational databases.
- 80654 hand-annotated questions and SQL queries across 24241 tables from Wikipedia.

Spider 1.0.

- Parallel data of utterances and queries
- 10,181 examples across 200 DBs
- Each example consists of an occurance and a DB with one or more tables and an annotated query

KaggleDBQA.

- Cross-domain and complex evaluation dataset of real Web databases,
- with domain-specific data types, original formatting, and unrestricted questions
- Databases are pulled from real-world data sources and not normalized
- Provides database documentation

Source (KaggleDBQA): https://aclanthology.org/2021.acl-long.176.pdf

Benchmark Datasets for T2S - Traditional

Spider-Realistic.

- Creates scenarios where the explicit appearance of the database schema in question is eliminated

Spider-SYN.

- Replaces schema-related words in questions with manually selected synonyms

SPIDER-CG.

- Splits the sentences in Spider into sub-sentences,
- Annotates each sub-sentence with the corresponding SQL clause,

Cspider.

- Translates Spider into Chinese to test the performance of Chinese text-to-SQL.

ADVETA.

- Adds adversarial table perturbation to deceive Text-to-SQL parsers

Benchmark Datasets for T2S – LLM Era

ScienceBenchmark

- Adaptation of spider dataset to domain specific queries is limited
- Sciencebenchmark is created with the help of SQL experts and domain specific researchers

BIRD

- Existing datasets focus on the database schema with few rows of database values
- Dataset of 37 professional domains
- Dirty and noisy database values
- External knowledge grounding between NL questions and database values
- Efficiency (in the context of massive databases)
- GPT-4 achieved 54.89% accuracy, human result 92.96%

Benchmark Datasets for T2S – LLM Era

Dr.Spider

- Existing models are sensitive to task specific perturbations
- 17 perturbations in T2S tasks
- Leveraged PLMs to simulate human behaviors in creating natural questions

Spider2.0

- Real scenarios involve complex cloud or local data across DB systems
- Multiple SQL queries in various dialects
- Operations from data transformation to analytics

Evaluation Metrics – Single Turn T2S

Exact Set Match Accuracy (EM)

- Compare ground truth SQL query with predicted SQL query
- SQL clauses: SELECT, GROUP BY, WHERE, ORDER BY, KEYWORDS

$$score(\hat{Y}, Y) = \begin{cases} 1, & \hat{Y} = Y \\ 0, & \hat{Y} \neq Y \end{cases}$$

Where $\hat{Y} = \{(\widehat{k^i}, \widehat{v^i}), i \in (1, m)\}$ and $Y = \{(k^i, v^i), i \in (1, m)\}$

$$EX = \frac{\sum_{n=1}^{N} score\ (\widehat{Y_n}, Y_n)}{N}$$

Drawback. Often underestimates the prediction accuracy of the model

k = SQL clause
v = corresponding value

Evaluation Metrics – Single Turn T2S

Execution Accuracy (EX).

- Comparison of the execution results
- the ground-truth SQL query and the predicted SQL query

$$score(\hat{V}, V) = \begin{cases} 1, & \hat{V} = V \\ 0, & \hat{V} \neq V \end{cases}$$

$$EX = \frac{\sum_{n=1}^{N} score\ (\widehat{Y_n}, Y_n)}{N}$$

Drawback.

- SQL with different logic may yield the same results when executed
- EX overestimates the prediction accuracy of the model

Evaluation Metrics – Multiturn T2S

Question Match Accuracy (QM).

- EM score over all questions
- Its value is 1 for each question only if all predicted SQL clauses are correct.

$$score(\hat{Y}, Y) = \begin{cases} 1, & \hat{Y} = Y \\ 0, & \hat{Y} \neq Y \end{cases}$$

$$QM = \frac{\sum_{m=1}^{M} score\ (\widehat{Y_m}, Y_m)}{M}$$

M = total number of question

Evaluation Metrics

Valid Efficiency Score (VES).

- SQL execution efficiency in the evaluation scope
- The higher the correctness rate of the generated SQL, the higher the execution efficiency of the generated SQL, and the higher the VES value

$$VES = \frac{\sum_{n=1}^{N} 1\left(V_n, \hat{V}_n\right). R(Y_n, \hat{Y}_n)}{N}, R\left(Y_n, \hat{Y}_n\right) = \sqrt{\frac{E(Y_n)}{E(\hat{Y}_n)}}$$

Source: https://dl.acm.org/doi/10.5555/3666122.3667957

Summary

- What are the traditional approaches for T2S?
- Masked Language Models for T2S (TaBERT)
- Large Language Models for T2S
  - Preprocessing
  - Inference
  - Post-processing
  - Fine-tuning