Text-to-SQL Semantic Parsing

Group 4
Nguyen Tan Viet
Ngo Quoc Bao

March 16, 2022

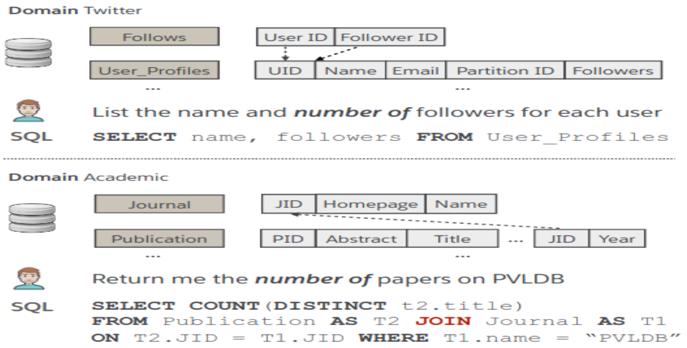
Problem

Definition of problem

Given a natural language Q and the schema $S = \langle \mathcal{T}, \mathcal{C} \rangle$ for a relational database, the parser needs to generate the corresponding SQL query Y. The schema consists of tables $\mathcal{T} = \{t_1, .., t_N\}$ and fields $\mathcal{C} = \{c_{11}, ..., c_{1|T_1|}, ..., c_{n1}, ..., c_{N|T_N|}\}$.Each table t_i and each field c_{ij} has a textual name. Some fields are primary keys, used for uniquely indexing each data record, and some are foreign keys, used to reference a primary key in a different table. In addition, each field has a data type, $\tau \in \{number, text, time, boolean, etc\}$

Problem

Definition of problem: SQL Cross domain database



List the name and ***number of*** followers for each user

```
SELECT name, followers FROM User_Profiles
```

Return me the ***number of*** papers on PVLDB

```
SELECT COUNT(DISTINCT t2.title)
FROM Publication AS T2 JOIN Journal AS T1
ON T2.JID = T1.JID WHERE T1.name = "PVLDB"
```

Figure 1: Two questions from the Spider dataset with similar intent resulted in completely different SQL logical forms on two DBs. In cross-DB text-to-SQL semantic parsing, the interpretation of a natural language question is strictly grounded in the underlying relational DB schema.

Literal review

Represent schema as Graph

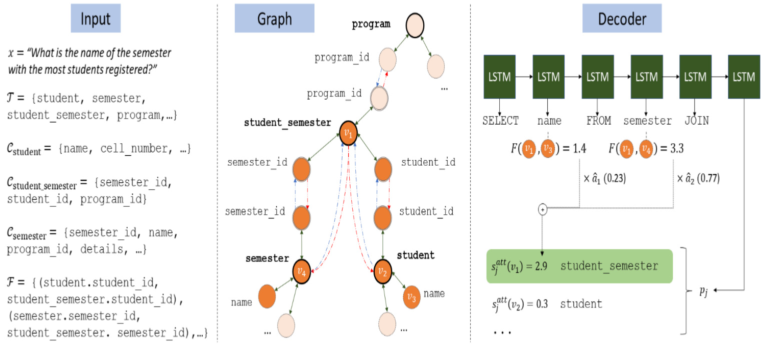

Input
x = "What is the name of the semester with the most students registered?"
𝒯 = {student, semester, student_semester, program,…}
𝒞student = {name, cell_number, …}
𝒞student_semester = {semester_id, student_id, program_id}
𝒞semester = {semester_id, name, program_id, details, …}
ℱ = {(student.student_id, student_semester.student_id), (semester.semester_id, student_semester. semester_id),…}
Graph
program
program_id
program_id
student_semester
semester_id
student_id
semester_id
student_id
semester
student
name
name
Decoder
LSTM
SELECT
name
FROM
semester
JOIN
F(v1,v3) = 1.4
F(v1,v4) = 3.3
× â1 (0.23)
× â2 (0.77)
s_j^att(v1) = 2.9 student_semester
s_j^att(v2) = 0.3 student
p_j

Figure 3: Left: DB schema and question. Middle: A graph representation of the schema. Bold nodes are tables, other nodes are columns. Dashed red (blue) edges are foreign (primary) keys edges, green edges are table-column edges. Right: Use of the schema by the decoder. For clarity, the decoder outputs tokens rather than grammar rules.

Literal review

Represent schema as Graph

1. Use decoder uses graph structue to guide decoding
2. E.g: LSTM generates token by token when decoder generates token schema ν, the subsequent generated tokens can be considered as $\tau(\nu)$.
3. For example to generate SQL query SELECT FROM *student* WHERE $student.lastname == "Nguyen"$. Decoder first generates SELECT FROM then the next token must be table name according to grammar rules.

Literal review

Represent problem as Seq2seq models for text2SQL

1. Encoder encodes natural question,sequential schema and translates to SQL logical form.
2. Shaw et al.(2020) showed T5 model with 3 billion parameters achieves state of the art on spider.

Literal review

Use DB content

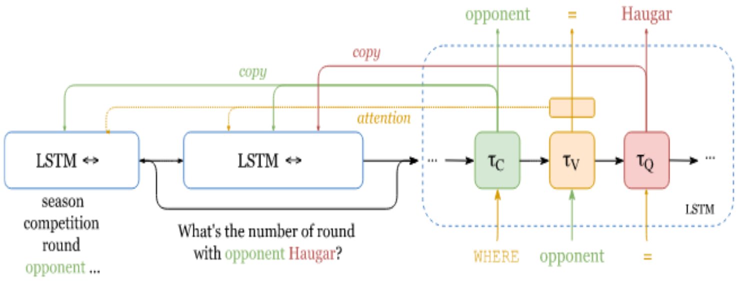
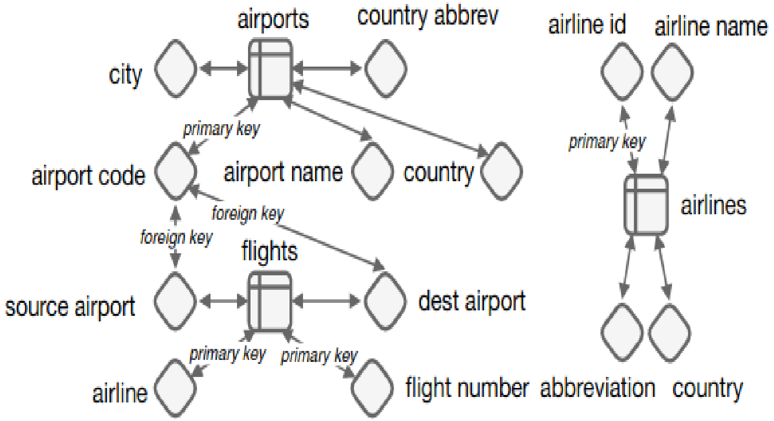


Figure 2: Overview of the base model. The model encodes table columns as well as the user question with a BiLSTM and then decodes the hidden state with a typed LSTM, where the decoding action for each cell is statically determined.

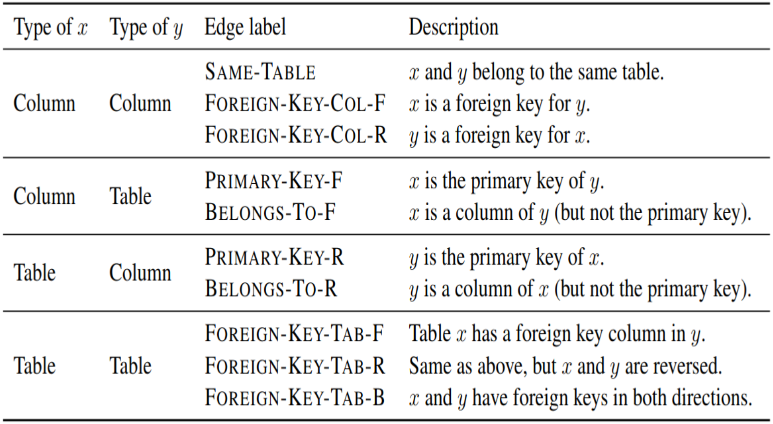
Literal review

Use DB content- RAT SQL



Literal review

Use DB content- RAT SQL



Type of x	Type of y	Edge label	Description
Column	Column	SAME-TABLE	x and y belong to the same table.
		FOREIGN-KEY-COL-F	x is a foreign key for y.
		FOREIGN-KEY-COL-R	y is a foreign key for x.
Column	Table	PRIMARY-KEY-F	x is the primary key of y.
		BELONGS-TO-F	x is a column of y (but not the primary key).
Table	Column	PRIMARY-KEY-R	y is the primary key of x.
		BELONGS-TO-R	y is a column of x (but not the primary key).
Table	Table	FOREIGN-KEY-TAB-F	Table x has a foreign key column in y.
		FOREIGN-KEY-TAB-R	Same as above, but x and y are reversed.
		FOREIGN-KEY-TAB-B	x and y have foreign keys in both directions.

Literal review

Use DB content- RAT SQL

1. RAT-SQL uses relational transformers to encode relations of tokens.
$e_{ij}^{(h)} = \frac{x_i W_Q^{(h)} (x_j W_K^h + r_{ij}^K)^T}{\sqrt{d_z/H}}$
$z_i^{(h)} = \sum_{j=1}^n \alpha_{ij}^{(h)} (x_j W_V^{(h)} + r_{ij}^V)$
2. RAT-SQL performs schema-linking, name-based linking, value-based linking based on relation types above.
3. Encode schema and question is independent.

New point in research

1. Model has acess to value of each field called picklists (e.g *Property_type_code* can have one of the following values {"*Apartment*","*Field*","*House*","*Shop*","*Other*"})
   $\rightarrow$ protect individual data record and sensitive fields such as User IDs or credit numbers.
2. Use LSTM-based pointer generator with multihead-attention as decoder

New point in research

Question-schema serialization and encoding

1. $X = [CLS], Q, [SEP], [T], t_1, [C], c_{11}, ..., c_{1|T_1|}, [T], t_2, [C], c_{21}, ..., c_{N|T_N|}, [SEP]$
2. X is encoded with Bert
3. Folowed by a directional LSTM $\mathbf{h}_X \in \mathbb{R}^{|X|xn}$
4. The question $\mathbf{h}_X$ is passed through another bi-LSTM $\mathbf{h}_Q \in \mathbb{R}^{|Q|xn}$
5. Dense lookup features to represent meta-data of schema $f_{pri} \in \mathbb{R}^{2xn}$, $f_{for} \in \mathbb{R}^{2xn}$, $f_{type} \in \mathbb{R}^{|\tau|xn}$

New point in research

Question-schema serialization and encoding

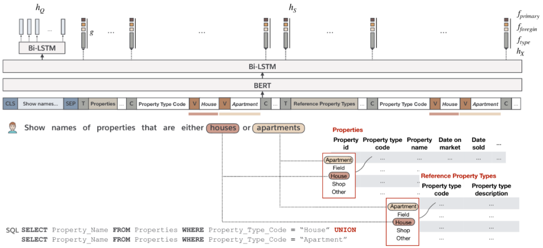
Meta-data features and base encoding are fused via a feed-forward layer $g(\mathbb{R}^{4n} \rightarrow \mathbb{R}^{n})$:

1. $\mathbf{h}_S^{t_i} = g([\mathbf{h}_X^p; \mathbf{0}; \mathbf{0}; \mathbf{0}])$
2. $\mathbf{h}_S^{c_{ij}} = g([\mathbf{h}_X^q; f_{pri}^u, f_{for}^v, f_{type}^w])$
   $= ReLU(\mathbf{W}_g[\mathbf{h}_X^q; f_{pri}^u, f_{for}^v, f_{type}^w] + \mathbf{b}_g)$
3. $\mathbf{h}_S = [\mathbf{h}^{t_1}, ..., \mathbf{h}^{t_{|\tau|}}, \mathbf{h}^{c_{11}}, ..., \mathbf{h}^{c_{N|T_N|}}] \in \mathbb{R}^{|S|xn}$

New point in research

Bridging

Anchor text, perform fuzzy matching between question Q and the picklists of each field, if found the matched values in DB, the mathched field values are inserted in X, succeeding corresponding field name and seperated by token $[V]$.



```
SQL SELECT Property_Name FROM Properties WHERE Property_Type_Code = "House" UNION
    SELECT Property_Name FROM Properties WHERE Property_Type_Code = "Apartment"
```

Figure 2: The BRIDGE encoder. The two phrases "houses" and "apartments" in the input question both matched to two DB fields. The matched values are appended to the corresponding field names in the hybrid sequence.

New point in research

Decoder

The decoder starts from the final state of the question encoder. At each step, the decoder performs one of the follow- ing actions: generating a token from the vocabulary V, copying a token from the question Q or copying a schema component from S.
Mathematically, at each step t, given the decoder state s_t and the encoder representation $[h_Q; h_S] \in \mathbb{R}^{(|Q|+|S|)xn}$

$$e_{tj}^{(h)} = \frac{s_t W_U^{(h)} (h_j W_V^{(h)})^\top}{\sqrt{n/H}}; \alpha_{tj}^{(h)} = \underset{j}{softmax}\{e_{tj}^{(h)}\}$$

$$z_t^{(h)} = \sum_{j=1}^{|Q|+|S|} \alpha_{tj}^{(h)} (h_j W_V^{(h)}); z_t = [z_t^{(1)}, ..., z_t^{(H)}]$$

$$p_{gen}^t = sigmoid(s_t W_{gen}^s + z_t W_{gen}^z + b_{gen})$$

$$p_{out}^t = p_{gen}^t P_{\mathcal{V}}(y_t) + (1 - p_{gen}^t) \sum_{j:\tilde{X}_j = y_t} \alpha_{tj}^{(H)})$$

New point in research

Schema-Consistency Guided Decoding

a simple pruning strategy for sequence decoders, based on the fact that the DB fields ap- peared in each SQL clause must only come from the tables in the FROM clause.

Lemma 1 Let Y_{exec} be a SQL query with clauses arranged in execution order, then any table field in Y_{exec} must appear after the table.

As a result, we adopt a binary attention mask ξ

$$\overset{\sim}{\alpha}_t^{(H)} = \alpha_t^{(H)}.\xi$$

which initially has entries corresponding to all fields set to 0 Once a table t_i is decoded, all entries in ξ corresponding to that table to 1, allows the decoder to only search in the space specified by the condition in Lemma 1 with little overhead in decoding speed.

Experiment result

Dataset

1. Spider: Train/dev/test databases not overlap, test set is hidden from public
2. WikiSQL: 49.6% of its dev tables and 45.1% of its test tables are not found in the train set
3. Both databases have the ability of models to generalize to unseen schema in train set

Experiment result

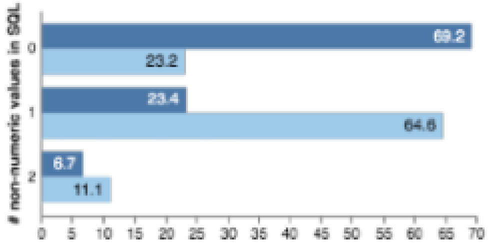
Evaluation Metrics

1. Exact Match (EM): Checks if the predicted SQL exactly matches the ground truth SQL
2. Exact Set Match (E-SM): Structural correctness of the predicted SQL. Check orderless set match
3. Execution Accuracy (EA): Execution results have the same result

Experiment result

Implementation

1. Anchor Text selection: Fuzzy matching *k* from DB field picklist and exclude number due to not effectively discriminate between different fields.
2. Training:
   - Loss: cross-entropy
   - Optimizer: Adam-SGD with mini-batch size of 32
   - Model: BERT-large + 1-layer LSTMs + 8-head attention



Experiment result

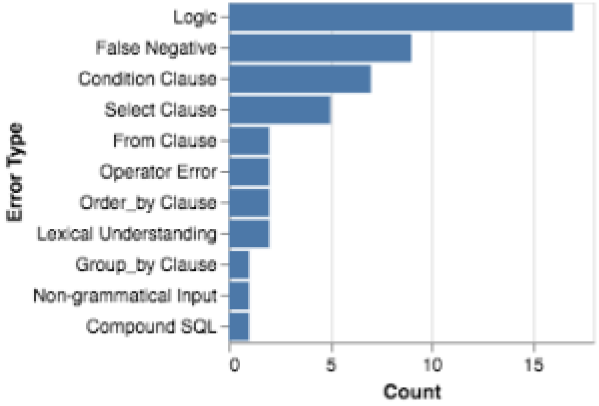
Result (Spider database, E-SM)

Model	Dev	Test
Global-GNN (Bogin et al., 2019b)	52.7	47.4
EditSQL + BERT (Zhang et al., 2019)	57.6	53.4
GNN + Bertrand-DR (Kelkar et al., 2020)	57.9	54.6
IRNet + BERT (Guo et al., 2019)	61.9	54.7
RAT-SQL v2 (Wang et al., 2019)	62.7	57.2
RYANSQL + BERTL (Choi et al., 2020)	66.6	58.2
SmBoP + BART (Rubin and Berant, 2020)	66.0	60.5
RYANSQL v2 + BERTL	70.6	60.6
RAT-SQL v3 + BERTL (Wang et al., 2019)	69.7	65.6
BRIDGE v1 (Lin et al., 2020)	65.5	59.2
BRIDGE L (ours)	70.0	65.0
BRIDGE L (ours, ensemble)	71.1	67.5

Error analysis

Evaluation

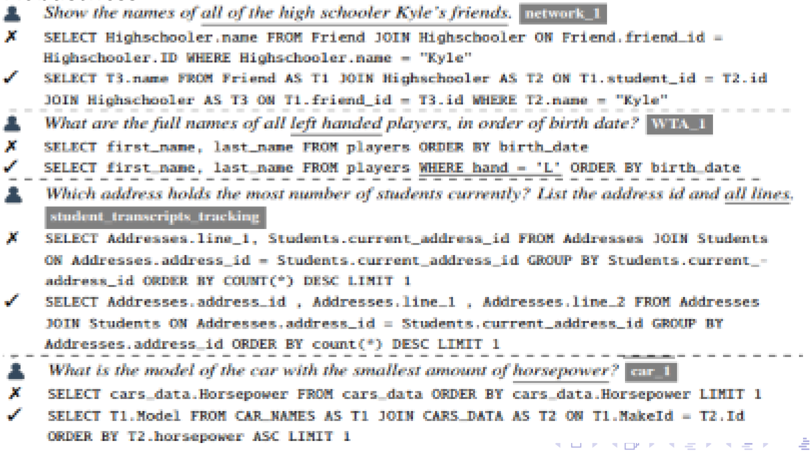
50 ramdom Spider dev set failed samples



Error analysis

Evaluation

4 common errors: Logical, Lexical Understanding, Commonsense, Robustness



👤 *Show the names of all of the high schooler Kyle's friends.* network_1

✗ `SELECT Highschooler.name FROM Friend JOIN Highschooler ON Friend.friend_id = Highschooler.ID WHERE Highschooler.name = "Kyle"`

✓ `SELECT T3.name FROM Friend AS T1 JOIN Highschooler AS T2 ON T1.student_id = T2.id JOIN Highschooler AS T3 ON T1.friend_id = T3.id WHERE T2.name = "Kyle"`

👤 *What are the full names of all left handed players, in order of birth date?* WTA_1

✗ `SELECT first_name, last_name FROM players ORDER BY birth_date`

✓ `SELECT first_name, last_name FROM players WHERE hand = 'L' ORDER BY birth_date`

👤 *Which address holds the most number of students currently? List the address id and all lines.* student_transcripts_tracking

✗ `SELECT Addresses.line_1, Students.current_address_id FROM Addresses JOIN Students ON Addresses.address_id = Students.current_address_id GROUP BY Students.current_address_id ORDER BY COUNT(*) DESC LIMIT 1`

✓ `SELECT Addresses.address_id , Addresses.line_1 , Addresses.line_2 FROM Addresses JOIN Students ON Addresses.address_id = Students.current_address_id GROUP BY Addresses.address_id ORDER BY count(*) DESC LIMIT 1`

👤 *What is the model of the car with the smallest amount of horsepower?* car_1

✗ `SELECT cars_data.Horsepower FROM cars_data ORDER BY cars_data.Horsepower LIMIT 1`

✓ `SELECT T1.Model FROM CAR_NAMES AS T1 JOIN CARS_DATA AS T2 ON T1.MakeId = T2.Id ORDER BY T2.horsepower ASC LIMIT 1`

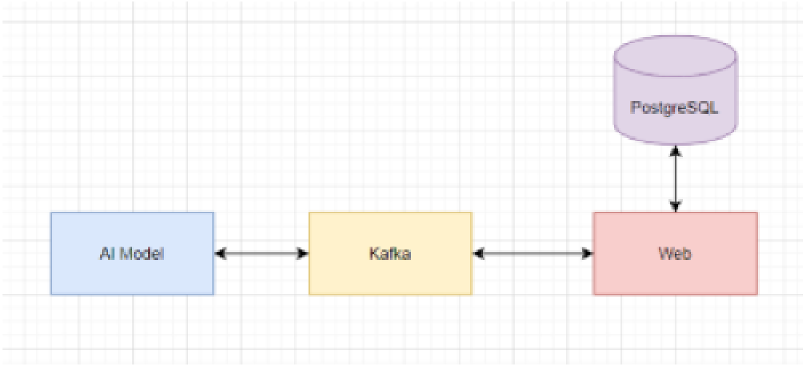
Application

1. Translation
2. Question-answer like Siri, Alexa (integrate with speech-to-text)
3. Smarthome Iot devices
4. Extension to support user in Web 3.0 generation. Applying semantics in the Web would enable machines to decode meaning and emotions by analyzing data. Consequently, internet users will have a better experience driven by enhanced data connectivity.

Demo

Question-Answer Website

Resources: 6 cores, 56GB RAM, 380GB storage - 1 x NVIDIA Tesla M60



References

1. https://github.com/salesforce/TabularSemanticParsing