Global Reasoning over Database Structures for Text-to-SQL Parsing

Ben Bogin, Matt Gardner, Jonathan Berant
Tel-Aviv University/Allen Institute for AI, 2019

George Tolkachev (georgeto@seas.upenn.edu)
February 19, 2020

Penn Engineering

Contents

- What is SQL?
- Problem setup
- Base model
- New contributions
- Experiments and results
- Shortcomings/limitations
- Potential for future work

A Brief Overview of SQL

- Stands for "**S**tructured **Q**uery **L**anguage"
- Designed for querying data within a relational database management system (relation = table)
- Relies on relational data model and uses relational algebra (union, intersect, minus, etc.)
- DB schema
  - Set of DB tables
  - Set of columns for each table
  - Set of foreign key-primary key column pairs

DB Constants

A Simple Example

```
MariaDB [georgeto]> DESCRIBE Airports;
```

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	NULL
name	varchar(20)	YES		NULL
city	varchar(20)	YES		NULL
country	varchar(20)	YES		NULL
iata	char(3)	YES		NULL
icao	char(4)	YES		NULL
lat	decimal(8,6)	YES		NULL
lon	decimal(9,6)	YES		NULL
alt	int(11)	YES		NULL
timezone	decimal(3,1)	YES		NULL
dst	char(1)	YES		NULL
tz	varchar(20)	YES		NULL

12 rows in set (0.00 sec)

```
MariaDB [georgeto]> SELECT * FROM Airports WHERE city = 'New York';
```

id	name	city	country	iata	icao
3697	La Guardia	New York	United States	LGA	KLGA
3797	John F Kennedy Intl	New York	United States	JFK	KJFK
3993	Wall Street Heliport	New York	United States	JRB	KJRB
4032	East 34th Street Hel	New York	United States	TSS	NONE
6966	Penn Station	New York	United States	ZYP	NULL
7729	West 30th St. Helipo	New York	United States	JRA	KJRA
7767	Idlewild Intl	New York	United States	IDL	KIDL
7881	Port Authority Bus T	New York	United States		NYPA
8010	NEW YORK CRUISE TERM	NEW YORK	United States		NULL
8123	One Police Plaza Hel	New York	United States		NK39
8591	All Airports	New York	United States	NYC	NULL
9350	Grand Central Termin	New York	United States		NULL
9351	Tremont	New York	United States		NULL
9451	Port Authority	New York	United States		NULL

14 rows in set (0.01 sec)

Problem Setup

Input: $\{(x^{(k)}, y^{(k)}, S^{(k)})\}_{k=1}^{N}$

- $x^{(k)}$: question
- $y^{(k)}$: query
- $S^{(k)}$: DB schema

Goal: map question-schema pairs (x, S) to correct query y

Must be able to generalize to new schemas S that were not observed at training time

x: *What is the name and nation of artists with a song with the word 'Hey' in its name?*

$\hat{y}$: `SELECT ___?___`

(a) `singer.name` 48%
(b) `song.name` 48%
(c) `singer.country` 2%
...

Local similarities:	name	nation	song	'Hey'	...
`singer.name`	48%	3%	3%	2%	
`singer.country`	2%	94%	2%	1%	
`song.name`	48%	3%	91%	77%	

...

Challenges

- Choice between DB constants can be ambiguous
- Same English word can refer to different DB constants based on context
- Queries can be complex/nested
- May require merging multiple tables
  - Which tables to merge?
  - How/on which attribute?

x : *Find the age of students who **do not have** a cat pet.*

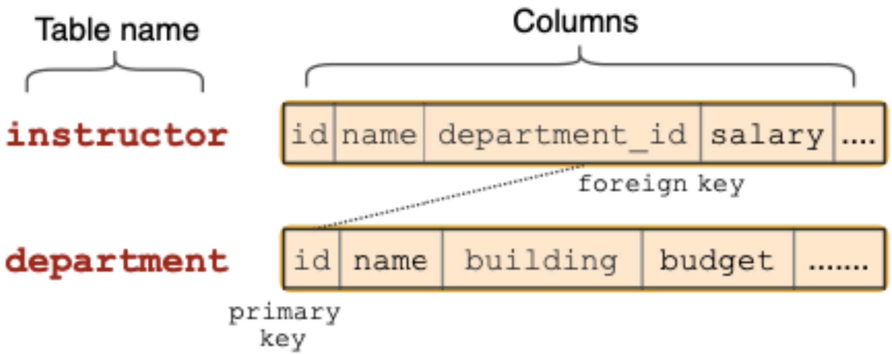
y : `SELECT age FROM student WHERE student NOT IN (SELECT ... FROM student JOIN has_pet ... JOIN pets ... WHERE ...)`

x : *What are the names of teams that **do not have** match season record?*

y : `SELECT name FROM team WHERE team_id NOT IN (SELECT team FROM match_season)`

- Similar questions can map to different queries, depending on the schema
- <u>Existing semantic parsers are auto-regressive</u>
  - DB constants are selected one at a time rather than as a set
  - Local similarity function between words and DB constants
  - How to take global context into account?

Example of a Complex Query



Complex question

What are the name and budget of the departments with average instructor salary greater than the overall average?

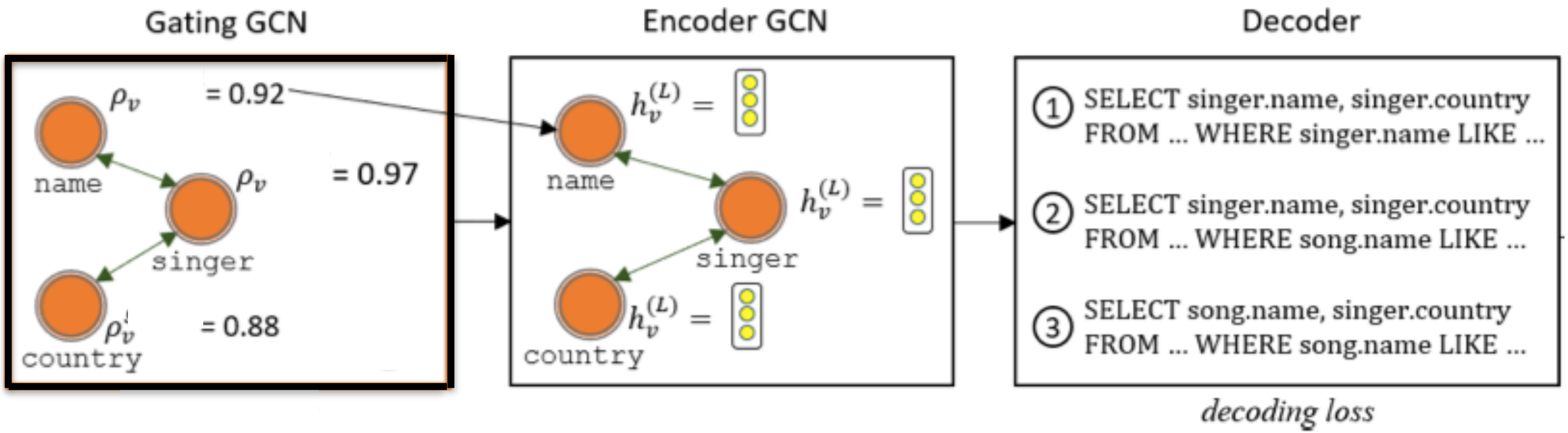
Complex SQL

```
SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as
T2 ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) >
     (SELECT avg(salary) FROM instructor)
```

Base model: Top-down zero-shot semantic parser

- Top-down zero-shot semantic parser
- Input question $(x_1, x_2, \ldots, x_{|x|})$ encoded with a BiLSTM, where hidden state e_i is a contextualized representation of word x_i
- Output query y decoded with another LSTM using a SQL grammar
- Focus of this paper: decoding of DB constants
- Major drawback: parsing is auto-regressive

GCN (Graph Convolutional Network)



Base Model Pseudocode

Input: schema S, question $x = (x_1, x_2, \ldots, x_{|x|})$

For every DB constant v in S:

Create $\mathbf{r_v}$, a learned embedding of v

For every question word x_i:

Compute local similarity score $s_{\text{link}}(v, x_i)$ from learned embeddings of v and x_i

Define distribution $p_{\text{link}}(v \mid x_i) \propto \exp(s_{\text{link}}(v, x_i))$

Using gating GCN, calculate relevance probability of v as $p_v = \max_i p_{\text{link}}(v \mid x_i)$

Using encoder GCN, calculate initial representation of v as $\mathbf{h_v}^{(0)} = p_v \cdot \mathbf{r_v}$

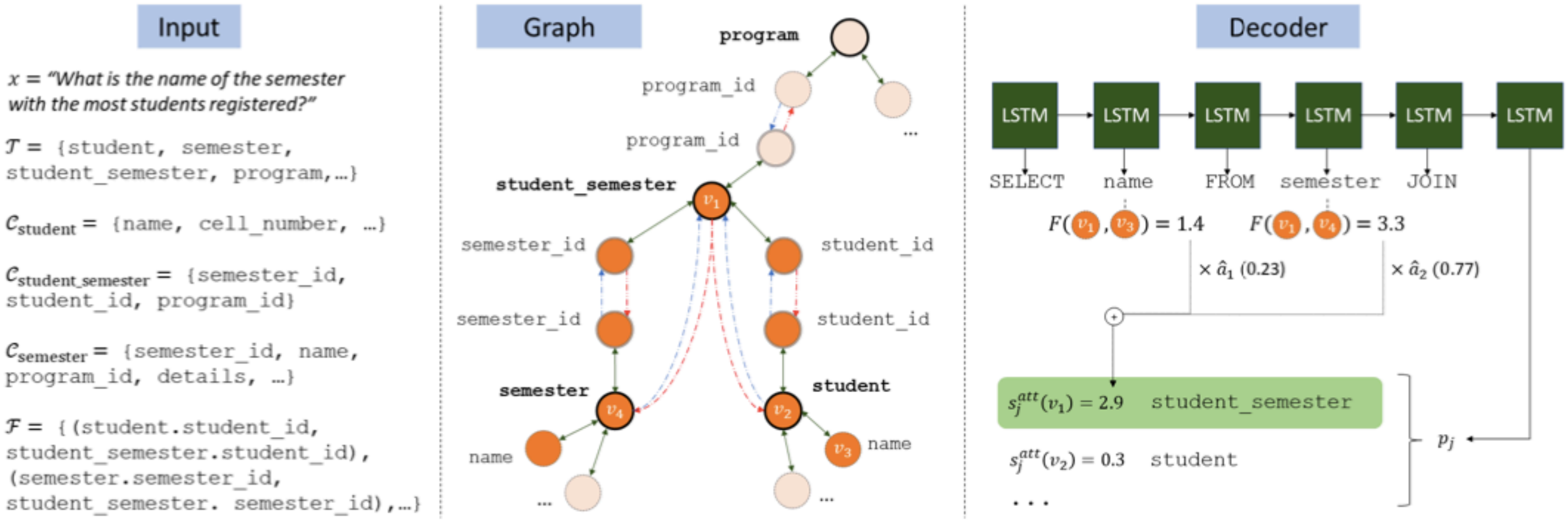
Apply GCN recurrence L times; final representation of v is $\mathbf{h_v} = (p_v \cdot \mathbf{r_v})^{(L)}$

Using $\mathbf{h_v}$, compute an attention distribution α_i over all words x_i

Score for v is $s_v = \sum_i \alpha_i s_{\text{link}}(v, x_i)$

Output: DB constant v with highest score s_v (output distribution is softmax($\{s_v\}_{v \in V}$))

Example of Decoding



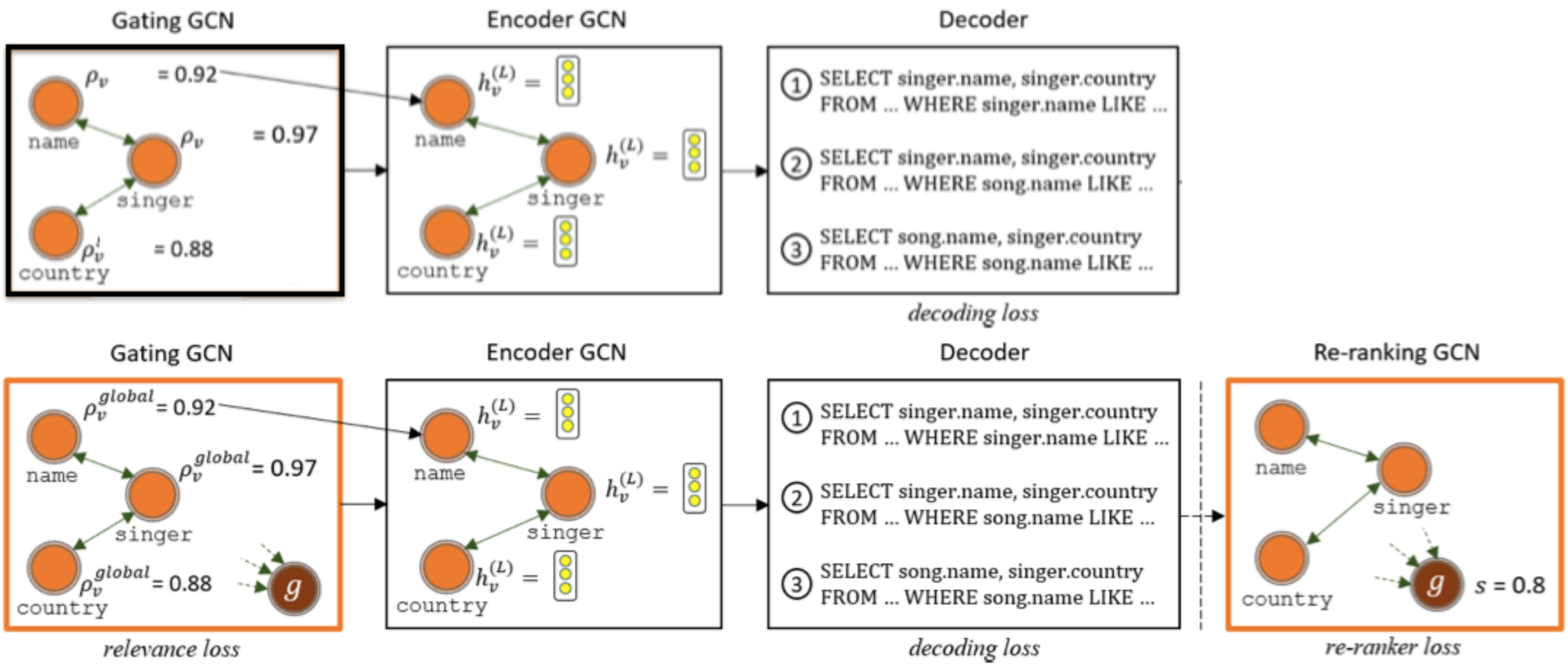
Contributions of Authors

- Learned gating GCN to estimate relevance probability for each node
  - Softly selects DB constants most likely to appear in output query
- Re-ranking GCN to discriminatively re-evaluate top K queries output by decoder based on global match between question and DB schema
  - Ensures that query covers all aspects of question

Base model vs. Global reasoning

Base model:

Global reasoning:



Global Gating

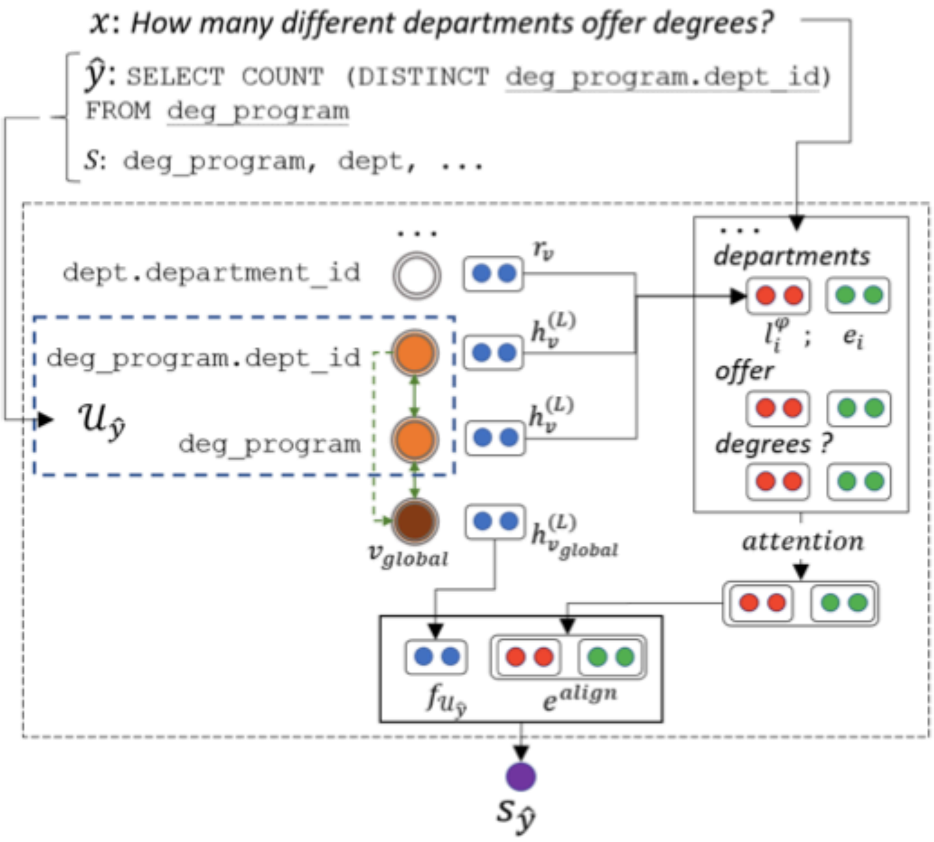
- Same input to gating GCN, but add new node v_{global} to shorten paths between other nodes
  - Initial embedding of v_{global} randomly initialized
- Input to GCN at node v is $\mathbf{g_v}^{(0)} = FF([\mathbf{r_v}; \bar{\mathbf{h}}_\mathbf{v}; p_v)]$, a representation for DB constant and question
  - ; represents concatenation
  - $FF(\cdot)$: feed-forward network
  - $\bar{\mathbf{h}}_\mathbf{v} = \sum_i p_{\text{link}}(x_i \mid v) \cdot e_i$): weighted average of contextual representation of question words
- Calculate new relevance probability $p_v^{\text{global}} = \sigma(FF(\mathbf{g_v}^{(L)}))$, which replaces original p_v as new input to encoder GCN
- In addition to usual decoding loss, add relevance loss: $-\sum_{v \in \mathcal{U}_y} \log p_v^{\text{global}} - \sum_{v \notin \mathcal{U}_y} \log(1 - p_v^{\text{global}})$
  - $\mathcal{U}_y$: subset of DB constants that appear in gold query y

Discriminative Re-Ranking

- Purpose: score each candidate tuple $(x, S, \hat{y})$ and globally reason over each candidate query $\hat{y}$
- Re-ranker trained to minimize re-ranker loss, i.e. the negative log probability of gold query y
- For each $\hat{y}$, compute logit $s_{\hat{y}} = \mathbf{w}^{\mathrm{T}} FF(\mathbf{f}_{\mathcal{U}_{\hat{y}}}, e^{\text{align}})$
  - $\mathbf{w}$: learned parameter vector
  - $\mathbf{f}_{\mathcal{U}_{\hat{y}}} = (FF(\mathbf{r}_{\mathbf{v}}; \bar{\mathbf{h}}_{\mathbf{v}}))^{(L)_{v_{global}}}$
    - Representation for sub-graph induced by the set $\mathcal{U}_{\hat{y}}$
    - v_{global} representation used to score question-conditioned subgraph
  - $e^{\text{align}} = [e_i; \sum_{v \in V} p_{\text{link}}(v \,|\, x_i) \cdot \phi_{\mathbf{v}}]$, where $\phi_{\mathbf{v}} = \mathbf{f}_{\mathbf{v}}^{(L)}$ if $v \in \mathcal{U}_{\hat{y}}$ and $\mathbf{r}_{\mathbf{v}}$ otherwise
    - Representation for global alignment between question words and DB constants
    - Goal: allow model to recognize attended words that are aligned with DB constants but were not selected for $\mathcal{U}_{\hat{y}}$

Penn Engineering

Re-ranking GCN architecture



Experimental Setup

- Train and evaluate on SPIDER, a zero-shot semantic parsing dataset with complex DBs
  - 7,000/1,034/2,147 train/development/test examples
- For training the re-ranker, take K = 40 candidates from beam output of the decoder
  - At each training step, if beam contains gold query, calculate the loss on the gold query and 10 random negative candidates
- At test time, re-rank top K = 10 candidates in the beam
- Remove either Global Gating or Re-Ranking functionalities and observe how results change

Results

Test set accuracy

Model	Accuracy
SyntaxSQLNet	19.7%
GNN	39.4%
Global-GNN	47.4%

Development set accuracy

Model	Accuracy	Beam	Single	Multi
GNN	40.7%	-	52.2%	26.8%
Global-GNN	**52.1%**	**65.9%**	**61.6%**	**40.3%**
- No Global Gating	48.8%	62.2%	60.9%	33.8%
- No Re-Ranking	48.3%	**65.9%**	58.1%	36.8%
- No Relevance Loss	50.1%	64.8%	60.9%	36.6%
No Align Rep.	50.8%	**65.9%**	60.7%	38.3%
Query Re-Ranker	47.8%	**65.9%**	55.3%	38.3%
Oracle Relevance	56.4%	73.5%	-	-

Qualitative Analysis

- Coverage: query covers all relevant question words
- Precision: query only joins tables relevant to question

Category	Question	Schema	Predicted Queries
Coverage	Show the name of the teacher for the math course.	**course**: course_id, staring_date, course **teacher**: teacher_id, name, age, hometown **course_arrange**: course_id, teacher_id, grade	**Baseline:** `SELECT teacher.name FROM teacher WHERE teacher.name = 'math'` **Our Model:** `SELECT teacher.name FROM teacher JOIN course_arrange ON teacher.teacher_id = course_arrange.teacher_id JOIN course ON course_arrange.course_id = course.course_id WHERE course.course = 'math'`
Precision	What are the makers and models?	**car_makers**: id, maker, fullname, country **model_list**: modelid, maker, model ...	**Baseline:** `SELECT car_makers.maker, model_list.model FROM car_makers JOIN model_list ON car_makers.id = model_list.maker` **Our Model:** `SELECT model_list.maker, model_list.model FROM model_list`

Shortcomings/Limitations

- This approach only deals with factual information and doesn't attempt to provide "reasons" for why one query works and another doesn't
- Some questions can be interpreted in multiple ways even in global setting - how to deal with these ambiguous cases?
- Doesn't consider missing data (only takes into account DB schema rather than contents)

Conclusions and Further Work

- Paying attention to the context of a token within a question improves English-to-SQL translation
- Re-ranker is best at choosing DB constants, while decoder can determine overall query structure
- A global model that selects both DB constants and SQL tokens might further improve performance
- Would be interesting to explore reverse translation, i.e. SQL to English