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

Ben Bogin[1] **Matt Gardner**[2] **Jonathan Berant**[1,2]

[1]School of Computer Science, Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
ben.bogin@cs.tau.ac.il, mattg@allenai.org, joberant@cs.tau.ac.il

Abstract

State-of-the-art semantic parsers rely on auto-regressive decoding, emitting one symbol at a time. When tested against complex databases that are unobserved at training time (zero-shot), the parser often struggles to select the correct set of database constants in the new database, due to the local nature of decoding. In this work, we propose a semantic parser that globally reasons about the structure of the output query to make a more contextually-informed selection of database constants. We use message-passing through a graph neural network to softly select a subset of database constants for the output query, conditioned on the question. Moreover, we train a model to rank queries based on the global alignment of database constants to question words. We apply our techniques to the current state-of-the-art model for SPIDER, a zero-shot semantic parsing dataset with complex databases, increasing accuracy from 39.4% to 47.4%.

1 Introduction

The goal of zero-shot semantic parsing (Krishnamurthy et al., 2017; Xu et al., 2017; Yu et al., 2018b,a; Herzig and Berant, 2018) is to map language utterances into executable programs in a new environment, or database (DB). The key difficulty in this setup is that the parser must map new lexical items to DB constants that weren't observed at training time.

Existing semantic parsers handle this mostly through a *local* similarity function between words and DB constants, which considers each word and DB constant in isolation. This function is combined with an auto-regressive decoder, where the decoder chooses the DB constant that is most similar to the words it is currently attending to. Thus, selecting DB constants is done one at a time rather than as a set, and informative global considerations are ignored.

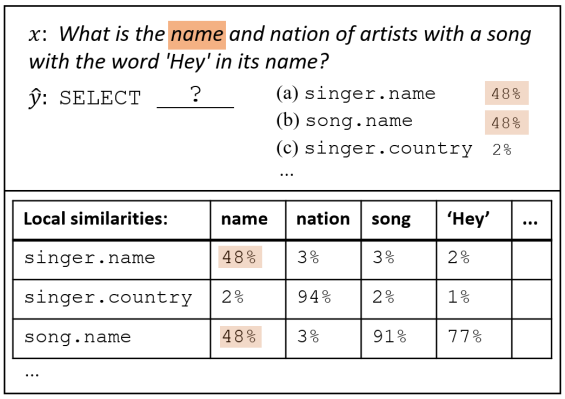

Figure 1: An example where choosing a DB constant based on local similarities is difficult, but the ambiguity can be resolved through global reasoning (see text).

Consider the example in Figure 1, where a question is mapped to a SQL query over a complex DB. After decoding `SELECT`, the decoder must now choose a DB constant. Assuming its attention is focused on the word *'name'* (highlighted), and given local similarities only, the choice between the lexically-related DB constants (`singer.name` and `song.name`) is ambiguous. However, if we globally reason over the DB constants and question, we can combine additional cues. First, a subsequent word *'nation'* is similar to the DB column `country` which belongs to the table `singer`, thus selecting the column `singer.name` from the same table is more likely. Second, the next appearance of the word *'name'* is next to the phrase *'Hey'*, which appears as the value in one of the cells of the column `song.name`. Assuming a one-to-one mapping between words and DB constants, again `singer.name` is preferred.

In this paper, we propose a semantic parser that reasons over the DB structure and question to make a *global* decision about which DB constants should be used in a query. We extend the parser of Bogin et al. (2019), which learns a representation for the

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3659–3664, Hong Kong, China, November 3–7, 2019.

DB schema at parsing time. First, we perform message-passing through a graph neural network representation of the DB schema, to softly select the set of DB constants that are likely to appear in the output query. Second, we train a model that takes the top-K queries output by the auto-regressive model and re-ranks them based on a global match between the DB and the question. Both of these technical contributions can be applied to any zero-shot semantic parser.

We test our parser on SPIDER, a zero-shot semantic parsing dataset with complex DBs. We show that both our contributions improve performance, leading to an accuracy of 47.4%, well beyond the current state-of-the-art of 39.4%.

Our code is available at https://github.com/benbogin/spider-schema-gnn-global.

2 Schema-augmented Semantic Parser

Problem Setup We are given a training set $\{(x^{(k)}, y^{(k)}, S^{(k)})\}_{k=1}^{N}$, where $x^{(k)}$ is a question, $y^{(k)}$ is its translation to a SQL query, and $S^{(k)}$ is the schema of the corresponding DB. We train a model to map question-schema pairs (x, S) to the correct SQL query. Importantly, the schema S was not seen at training time.

A DB schema S includes : (a) A set of DB tables, (b) a set of columns for each table, and (c) a set of foreign key-primary key column pairs where each pair is a relation from a foreign-key in one table to a primary-key in another. Schema tables and columns are termed *DB constants*, denoted by $\mathcal{V}$.

We now describe a recent semantic parser from Bogin et al. (2019), focusing on the components relevant for selecting DB constants.

Base Model The base parser is a standard top-down semantic parser with grammar-based decoding (Xiao et al., 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017; Rabinovich et al., 2017; Lin et al., 2019). The input question $(x_1, \ldots, x_{|x|})$ is encoded with a BiLSTM, where the hidden states $\boldsymbol{e}_i$ of the BiLSTM are used as contextualized representations for the word x_i. The output query y is decoded top-down with another LSTM using a SQL grammar, where at each time step a grammar rule is decoded. Our main focus is decoding of DB constants, and we will elaborate on this part.

The parser decodes a DB constant whenever the previous step decoded the non-terminals `Table` or `Column`. To select the DB constant, it first computes an attention distribution over the question words $\{\alpha_i\}_{i=1}^{|x|}$ in the standard manner (Bahdanau et al., 2015). Then the score for a DB constant v is $s_v = \sum_i \alpha_i s_{\text{link}}(v, x_i)$, where s_{link} is a local similarity score, computed from learned embeddings of the word and DB constant, and a few manually-crafted features, such as the edit distance between the two inputs and the fraction of string overlap between them. The output distribution of the decoder is simply $\text{softmax}(\{s_v\}_{v \in \mathcal{V}})$. Importantly, the dependence between decoding decisions for DB constants is weak – the similarity function is independent for each constant and question word, and decisions are far apart in the decoding sequence, especially in a top-down parser.

DB schema encoding In the zero-shot setting, the schema structure of a new DB can affect the output query. To capture DB structure, Bogin et al. (2019) learned a representation $\boldsymbol{h}_v$ for every DB constant, which the parser later used at decoding time. This was done by converting the DB schema into a graph, where nodes are DB constants, and edges connect tables and their columns, as well as primary and foreign keys (Figure 2, left). A graph convolutional network (GCN) then learned representations $\boldsymbol{h_v}$ for nodes end-to-end (De Cao et al., 2019; Sorokin and Gurevych, 2018).

To focus the GCN's capacity on important nodes, a *relevance probability* ρ_v was computed for every node, and used to "gate" the input to the GCN, conditioned on the question. Specifically, given a learned embedding $\boldsymbol{r}_v$ for every database constant, the GCN input is $\boldsymbol{h}_v^{(0)} = \rho_v \cdot \boldsymbol{r}_v$. Then, the GCN recurrence is applied for L steps. At each step, nodes re-compute their representation based on the representation of their neighbors, where different edge types are associated with different learned parameters (Li et al., 2016). The final representation of each DB constant is $\boldsymbol{h}_v = \boldsymbol{h}_v^{(L)}$.

Importantly, the relevance probability ρ_v, which can be viewed as a soft selection for whether the DB constant should appear in the output, was computed based on local information only: First, a distribution $p_{\text{link}}(v \mid x_i) \propto \exp(s_{\text{link}}(v, x_i))$ was defined, and then $\rho_v = \max_i p_{\text{link}}(v \mid x_i)$ was computed deterministically. Thus, ρ_v doesn't consider the full question or DB structure. We address this next.

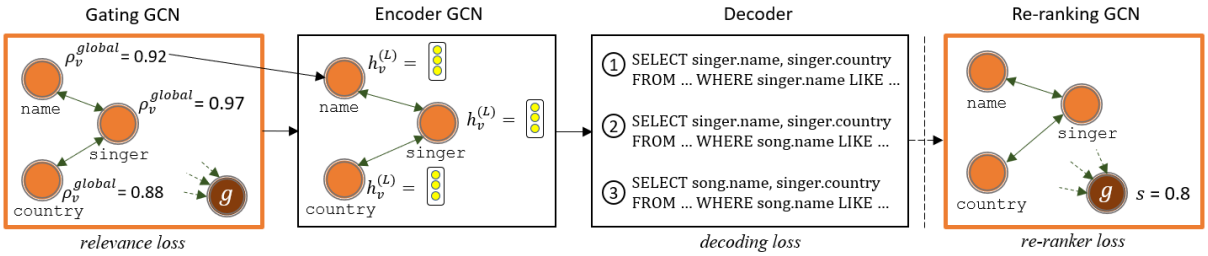

Figure 2: High-level overview, where our contributions are in thick orange boxes. First, a relevance score is predicted for each of the DB constants using the gating GCN. Then, a learned representation is computed for each DB constant using the encoder GCN, which is then used by the decoder to predict K candidates queries. Finally, the re-ranking GCN scores each one of these candidates, basing its score only on the selected DB constants. The dashed line and arrow indicate no gradients are propagated from the re-ranking GCN to the decoder, as the decoder outputs SQL queries. Names of loss terms are written below models that are trained with a loss on their output.

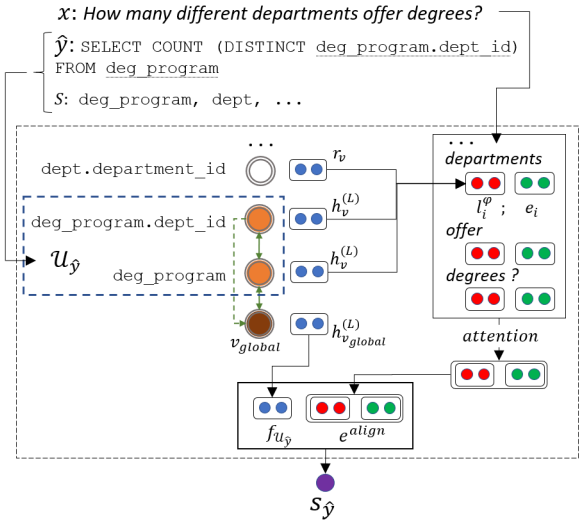

Figure 3: The re-ranking GCN architecture (see text).

3 Global Reasoning over DB Structures

Figure 2 gives a high-level view of our model, where the contributions of this paper are marked by thick orange boxes. First, the aforementioned relevance probabilities are estimated with a *learned* gating GCN, allowing global structure to be taken into account. Second, the model discriminatively re-ranks the top-K queries output by the generative decoder.

Global gating Bogin et al. (2019) showed that an oracle relevance probability can increase model performance, but computed ρ_v from local information only.

We propose to train a GCN to directly predict ρ_v from the global context of the question and DB.

The input to the gating GCN is the same graph described in §2, except we add a new node v_{global}, connected to all other nodes with a special edge type. To predict the question-conditioned relevance of a node, we need a representation for both the DB constant and the question. Thus, we define the input to the GCN at node v to be $\boldsymbol{g}_v^{(0)} = FF([\boldsymbol{r}_v; \bar{\boldsymbol{h}}_v; \rho_v])$, where ';' is concatenation, $FF(\cdot)$ is a feed-forward network, and $\bar{\boldsymbol{h}}_{\boldsymbol{v}} = \sum_i p_{\text{link}}(x_i \mid v) \cdot \boldsymbol{e}_i$ is a weighted average of contextual representations of question tokens. The initial embedding of v_{global} is randomly initialized. A relevance probability is computed per DB constant based on the final graph representation: $\rho_v^{\text{global}} = \sigma(FF(\boldsymbol{g}_v^{(L)}))$. This probability replaces ρ_v at the input to the encoder GCN (Figure 2).

Because we have the gold query y for each question, we can extract the gold subset of DB constants $\mathcal{U}_y$, i.e., all DB constants that appear in y. We can now add a *relevance loss* term $-\sum_{v \in \mathcal{U}_y} \log \rho_v^{\text{global}} - \sum_{v \notin \mathcal{U}_y} \log(1 - \rho_v^{\text{global}})$ to the objective. Thus, the parameters of the gating GCN are trained from the relevance loss and the usual *decoding loss*, a ML objective over the gold sequence of decisions that output the query y.

Discriminative re-ranking Global gating provides a more accurate model for softly predicting the correct subset of DB constants. However, parsing is still auto-regressive and performed with a local similarity function. To overcome this, we separately train a discriminative model (Collins and Koo, 2005; Ge and Mooney, 2006; Lu et al., 2008; Fried et al., 2017) to re-rank the top-K queries in the decoder's output beam. The re-ranker scores each candidate tuple $(x, S, \hat{y})$, and thus can globally reason over the entire candidate query $\hat{y}$.

We focus the re-ranker capacity on the main pain point of zero-shot parsing – the set of DB constants $\mathcal{U}_{\hat{y}}$ that appear in $\hat{y}$. At a high-level (Figure 3), for each candidate we compute a logit $s_{\hat{y}} = \boldsymbol{w}^\top FF(\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}, \boldsymbol{e}^{\text{align}})$, where $\boldsymbol{w}$ is a learned

parameter vector, $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}$ is a representation for the set $\mathcal{U}_{\hat{y}}$, and $\boldsymbol{e}^{\text{align}}$ is a representation for the global alignment between question words and DB constants. The re-ranker is trained to minimize the *re-ranker loss*, the negative log probability of the correct query y. We now describe the computation of $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}$ and $\boldsymbol{e}^{\text{align}}$, based on a re-ranking GCN.

Unlike the gating GCN, the re-ranking GCN takes as input only the sub-graph induced by the selected DB constants $\mathcal{U}_{\hat{y}}$, and the global node v_{global}. The input is represented by $\boldsymbol{f}_v^{(0)} = FF(\boldsymbol{r}_v; \bar{\boldsymbol{h}}_v)$, and after L propagation steps we obtain $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}} = \boldsymbol{f}^{(L)_{v_{\text{global}}}}$. Note that the global node representation is used to describe and score the question-conditioned sub-graph, unlike the gating GCN where the global node mostly created shorter paths between other graph nodes.

The representation $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}$ captures global properties of selected nodes but ignores nodes that were not selected and are possibly relevant. Thus, we compute a representation $\boldsymbol{e}^{\text{align}}$, which captures whether question words are aligned to selected DB constants. We define a representation for every node $v \in \mathcal{V}$:

$$\boldsymbol{\varphi}_v = \begin{cases} \boldsymbol{f}_v^{(L)} & \text{if } v \in \mathcal{U}_{\hat{y}} \\ \boldsymbol{r}_v & \text{otherwise} \end{cases}$$

Now, we compute for every question word x_i a representation of the DB constants it aligns to: $\boldsymbol{l}_i^{\varphi} = \sum_{v \in \mathcal{V}} p_{\text{link}}(v \mid x_i) \cdot \boldsymbol{\varphi}_v$. We concatenate this representation to every word $\boldsymbol{e}_i^{\text{align}} = [\boldsymbol{e}_i; \boldsymbol{l}_i^{\varphi}]$, and compute the vector $\boldsymbol{e}^{\text{align}}$ using attention over the question words, where the attention score for every word is $\boldsymbol{e}_i^{\text{align}\top} \boldsymbol{w}_{\text{att}}$ for a learned vector $\boldsymbol{w}_{\text{att}}$. The goal of this term is to allow the model to recognize whether there are any attended words that are aligned with DB constants, but these DB constants were not selected in $\mathcal{U}_{\hat{y}}$.

In sum, our model adds a gating GCN trained to softly select relevant nodes for the encoder, and a re-ranking GCN that globally reasons over the subset of selected DB constants, and captures whether the query properly covers question words.

4 Experiments and Results

Experimental setup We train and evaluate on SPIDER (Yu et al., 2018b), which contains 7,000/1,034/2,147 train/development/test examples, using the same pre-processing as Bogin et al. (2019). To train the re-ranker, we take $K = 40$ candidates from the beam output of the decoder. At each training step, if the gold query is in the beam, we calculate the loss on the gold query and 10 randomly selected negative candidates. At test time, we re-rank the best $K = 10$ candidates in the beam, and break re-ranking ties using the auto-regressive decoder scores (ties happen since the re-ranker considers the DB constants only and not the entire query). We use the official SPIDER script for evaluation, which tests for loose exact match of queries.

Model	Accuracy
SYNTAXSQLNET	19.7%
GNN	39.4%
GLOBAL-GNN	**47.4%**

Table 1: Test set accuracy of GLOBAL-GNN compared to prior work on SPIDER.

Model	Acc.	Beam	SINGLE	MULTI
SYNTAXSQLNET	18.9%		23.1%	7.0%
GNN	40.7%		52.2%	26.8%
+ RE-IMPLEMENTATION	44.1%	62.2%	58.3%	27.6%
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%		

Table 2: Development set accuracy for various experiments. The column 'Beam' indicates the fraction of examples where the gold query is in the beam ($K = 10$).

Results As shown in Table 1, the accuracy of our proposed model (GLOBAL-GNN) on the hidden test set is 47.4%, 8% higher than current state-of-the art of 39.4%. Table 2 shows accuracy results on the development set for different experiments. We perform minor modifications to the implementation of Bogin et al. (2019), improving the accuracy from 40.7% to 44.1% (details in appendix A). We follow Bogin et al. (2019), measuring accuracy on easier examples where queries use a single table (SINGLE) and those using more than one table (MULTI).

GLOBAL-GNN obtains 52.1% accuracy on the development set, substantially higher than all previous scores. Importantly, the performance increase comes mostly from queries that require more than one table, which are usually more complex.

Removing any of our two main contributions (NO GLOBAL GATING, NO RE-RANKING) leads to a 4% drop in performance. Training without the relevance loss (NO RELEVANCE LOSS) results in a 2% accuracy degrade. Omitting the representation e^{align} from the re-ranker (NO ALIGN REP.)

reduces performance, showing the importance of identifying unaligned question words.

We also consider a model that ranks the entire query and not only the set of DB constants. We re-define $s_{\hat{y}} = \boldsymbol{w}^\top FF(\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}, \boldsymbol{h}^{\text{align}}, \boldsymbol{h}^{\text{query}})$, where $\boldsymbol{h}^{\text{query}}$ is a concatenation of the last and first hidden states of a BiLSTM run over the output SQL query (QUERY RE-RANKER). We see performance is lower, and most introduced errors are minor mistakes such as `min` instead of `max`. This shows that our re-ranker excels at choosing DB constants, while the decoder is better at determining the SQL query structure and the SQL logical constants.

Finally, we compute two oracle scores to estimate future headroom. Assuming a perfect global gating, which gives probability 1.0 iff the DB constant is in the gold query, increases accuracy to 63.2%. Adding to that a perfect re-ranker leads to an accuracy of 73.5%.

Qualitative analysis Analyzing the development set, we find two main re-occurring patterns, where the baseline model is wrong, but our parser is correct. (a) *coverage*: when relevant question words are not covered by the query, which results in a missing joining of tables or selection of columns (b) *precision*: when unrelated tables are joined to the query due to high lexical similarity. Selected examples are in Appendix B.

Error analysis In 44.4% of errors where the correct query was in the beam, the selection of $\mathcal{U}$ was correct but the query was wrong. Most of these errors are caused by minor local errors, e.g., `min`/`max` errors, while the rest are due to larger structural mistakes, indicating that a global model that jointly selects both DB constants and SQL tokens might further improve performance. Other types of errors include missing or extra columns and tables, especially in complex queries.

5 Conclusion

In this paper, we demonstrate the importance of global decision-making for zero-shot semantic parsing, where selecting the relevant set of DB constants is challenging. We present two main technical contributions. First, we use a gating GCN that globally attends the input question and the entire DB schema to softly-select the relevant DB constants. Second, we re-rank the output of a generative semantic parser by globally scoring the set of selected DB-constants. Importantly, these contributions can be applied to any zero-shot semantic parser with minimal modifications. Empirically, we observe a substantial improvement over the state-of-the-art on the SPIDER dataset, showing the effectiveness of both contributions.

Acknowledgments

This research was partially supported by The Yandex Initiative for Machine Learning. This work was completed in partial fulfillment for the Ph.D degree of the first author.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

B. Bogin, M. Gardner, and J. Berant. 2019. Representing schema structure with graph neural networks for text-to-sql parsing. In *Association for Computational Linguistics (ACL)*.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

N. De Cao, W. Aziz, and I. Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *North American Association for Computational Linguistics (NAACL)*.

D. Fried, M. Stern, and D. Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Association for Computational Linguistics (ACL)*.

R. Ge and R. J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 263–270.

J. Herzig and J. Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Krishnamurthy, P. Dasigi, and M. Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. 2016. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*.

K. Lin, B. Bogin, M. Neumann, J. Berant, and M. Gardner. 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

W. Lu, H. T. Ng, W. S. Lee, and L. S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii. Association for Computational Linguistics.

M. Rabinovich, M. Stern, and D. Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Association for Computational Linguistics (ACL)*.

D. Sorokin and I. Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3306–3317, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

C. Xiao, M. Dymetman, and C. Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Association for Computational Linguistics (ACL)*.

X. Xu, C. Liu, and D. Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

P. Yin and G. Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Association for Computational Linguistics (ACL)*, pages 440–450.

T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domaintext-to-SQL task. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. *arXiv preprint arXiv:1809.08887*.