X-SQL: REINFORCE CONTEXT INTO SCHEMA REPRESENTATION

Pengcheng He, Yi Mao, Kaushik Chakrabarti, Weizhu Chen
Microsoft Dynamics 365 AI
{penhe,maoyi,kaushik,wzchen}@microsoft.com

ABSTRACT

In this work, we present a new network architecture called X-SQL for the problem of parsing natural language to SQL program. X-SQL proposes to reinforce the contextual output from BERT-style pre-training model into the structural schema representation, and learn a new schema representation for downstream tasks. We evaluated X-SQL on the WikiSQL dataset. Experimental results show that X-SQL achieves new state-of-the-art results on WikiSQL, with substantial improvement on all metrics over both dev and test set.

1 INTRODUCTION

The task of semantic parsing is to map natural language to a logical form representing its meaning. It has been studied extensively by the natural language processing community, with applications ranging from question answering (Zettlemoyer & Collins, 2005; Wong & Mooney, 2007; Zettlemoyer & Collins, 2007) to robot navigation (Chen & Mooney, 2011; Tellex et al., 2011). The choice of target meaning representation is highly application dependent. For problems related to structured data such as database or knowledge graph, both λ-calculus (Kwiatkowski et al., 2011) and SQL are popular picks.

In this work, we focus on the semantic parsing task of translating natural language queries into executable SQL programs (NL2SQL). We experiment with WikiSQL, a dataset introduced by Zhong et al. (2017). It is the first large-scale dataset with annotated pairs of natural language query and its corresponding SQL form. WikiSQL is highly diverse in its questions, table schemas and table contents, making it an attractive dataset for neural network modeling.

Early work on WikiSQL models it as a sequence generation problem, and leverages neural sequence-to-sequence models with attention and copy mechanisms (Sutskever et al., 2014; Bahdanau et al., 2015; Vinyals et al., 2015). Despite that there is no guarantee of generating syntactically-valid outputs, reasonably good results can already be achieved (Dong & Lapata, 2016; Zhong et al., 2017). Since then, various approaches have been proposed to incorporate the syntax of the SQL programming language into neural network models. Xu et al. (2017) and Yu et al. (2018) capture the syntax via dependency between different prediction modules. Dong & Lapata (2018) and Finegan-Dollak et al. (2018) use a slot filling approach where syntax is enforced with a predefined set of sketches. Wang et al. (2017) and Shi et al. (2018) take a sequence-to-action approach, and encode syntax in feasible actions. The work of Wang et al. (2018b) observes that partially generated SQL queries can already be executed, and tries to ensure syntactic correctness at various stages during decoding.

Despite the great success of applying the neural network to NL2SQL problem, the advantage of the neural network modeling is often constrained by the availability of the large-scale high-quality training data. Although WikiSQL is the largest public dataset so far, its size is still far away from the sufficiency to capture various natural language variation. Fortunately, recent advance of pretraining technique using large-scale external data has been showing appealing results in related tasks. Work such as Devlin et al. (2018), Radford et al. (2018) and Liu et al. (2019) have demonstrated the value of transfer learning from external data source in various natural language understanding tasks with significant improvement. In view of this trend, Hwang et al. (2019) replaced the glove encoding layer in Xu et al. (2017) with pre-trained BERT model (Devlin et al., 2018) and established the new state-of-the-art on WikiSQL. In X-SQL, we adapt the similar pre-training technique but leverage a

recent work called MT-DNN. Besides the unlabeled data utilized by BERT, MT-DNN combines it with labeled data via multi-task learning. This makes the WikiSQL task benefit from related labeled dataset captured in MT-DNN.

However, previous works use the output of pre-training models either to capture the sequential information (Xu et al., 2017; Hwang et al., 2019) or to predict the final output directly with a subsequent single layer network (Devlin et al., 2018; Liu et al., 2019). Moreover, this does not effectively capture the structure property in the semantic parsing problem such as NL2SQL. In contrast to this simple approach, we propose an additional layer to reinforce the contextual information from pre-training models to the highly structural schema information, such as each column in the table, and then build a new contextualized schema representation. Then, all downstream tasks are built on top of this new structural schema representation. We treat this as the first attempt to reinforce the BERT-style contextual information into a problem-dependent structure, and then build a new representation to better characterize its structural information for downstream tasks.

Meanwhile, we observed the challenge in the WikiSQL problem mainly lies in the where clause prediction, in which we need to predict a column set in the where clause, including zero column when there is no where clause in the SQL. Previous works often frame this problem as multiple binary classification problems, with each for a column. However, such approach can not effectively account for the relationship between columns, since their prediction are optimized independently. Furthermore, the outputs from multiple binary classifications are not comparable, since they are basically from different independent models. To tackle this issue, X-SQL proposes a list-wise global ranking approach using the KL divergence (Kullback, 1987) as its new objective. To cope with the situation that there is no where clause in some SQL, we intentionally introduce an Empty column into the encoder and consider it as a candidate in the global ranking prediction. This marriage between the Empty column and the global ranking objective brings us a novel approach to better solve the column set prediction in all NL2SQL problems.

We apply X-SQL into the WikiSQL dataset and compare it with recent state-of-the-art models in the WikiSQL leaderboard. X-SQL obtains new state-of-the-art results on all the metrics over both dev and test sets. The improvement over previous state-of-the-art SQLova is about absolute 2%, pushing the new execution accuracy to 91.8%. Based on the analysis in Hwang et al. (2019), even our score without execution guided (Wang et al., 2018a) has already surpassed human performance. All of these clearly demonstrate the exceptional performance of X-SQL on the WikiSQL dataset.

2 NEURAL ARCHITECTURE

Figure 1 shows the overall architecture consisting of three layers: encoder, context reinforcing layer and output layer.

2.1 ENCODER

For encoder, we use a model similar to the one in Devlin et al. (2018) with the following changes:

- A special empty column is introduced, with token `[EMPTY]` being appended to every table schema. Its usage will become clear in Section 2.4.
- Segment embeddings for question and schema segment are extended to type embeddings, where we learn embeddings for four different types: question, categorial column, numerical column and the special empty column.
- Instead of initializing with BERT-Large, we initialize our encoder with MT-DNN (Liu et al., 2019), which has the same architecture as BERT, but trained on multiple GLUE tasks (Wang et al., 2018a). MT-DNN has been shown to be a better representation for down-streaming NLP tasks.

Note, we replace `[CLS]` with `[CXT]` in Figure 1 to emphasize that context information is being captured there, rather than a representation for down-streaming tasks.

In addition to these three, our encoder differs from SQLova with NL2SQL layer (Hwang et al., 2019) in an important way: while they run bi-LSTM/column attention on top of encoder, consequent

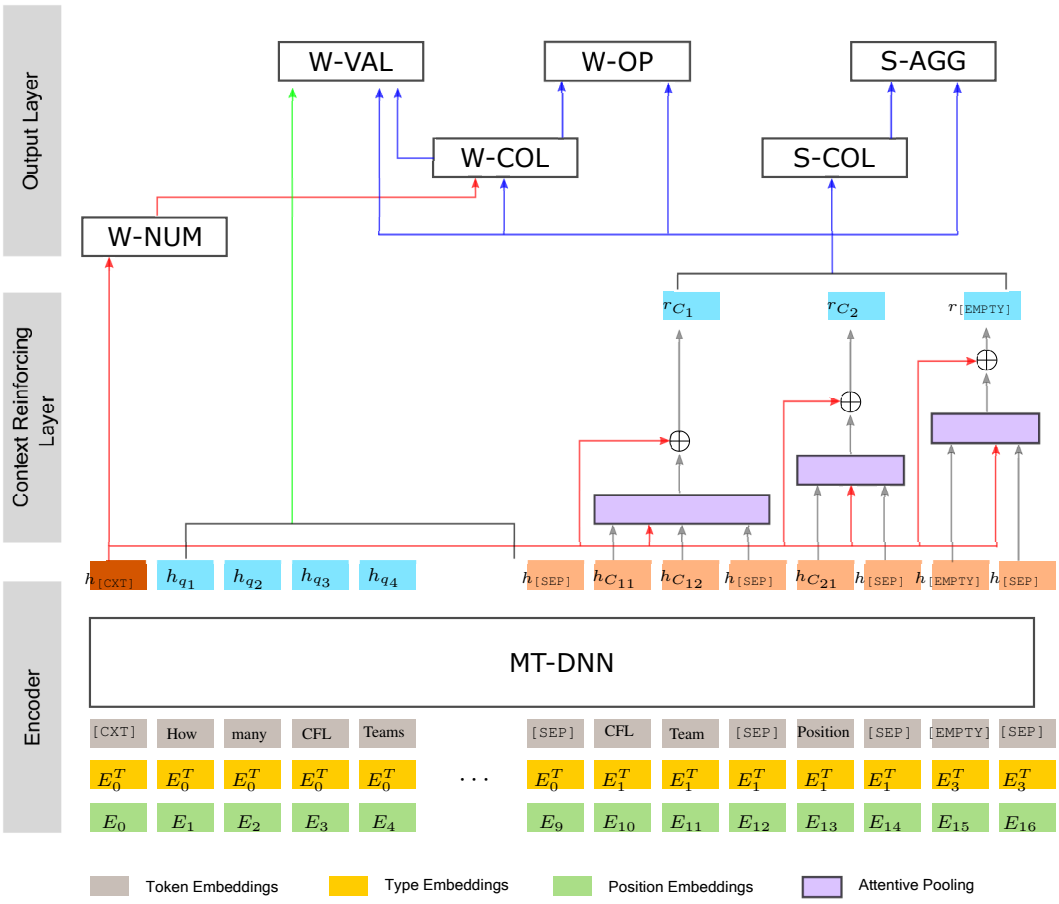

Figure 1: Our neural network model architecture.

layers in our architecture consist of simple yet powerful computations, which we believe is largely attributed to a better alignment of BERT for the problem.

2.2 Context Reinforcing Layer

Let $h_{\texttt{[CXT]}}, h_{q_1}, \cdots, h_{q_n}, h_{\texttt{[SEP]}}, h_{C_{11}}, \cdots, h_{\texttt{[SEP]}}, h_{C_{21}}, \cdots h_{\texttt{[SEP]}}, h_{\texttt{[EMPTY]}}, h_{\texttt{[SEP]}}$ denote the output from the encoder, each of dimension d. Note, each column name may contain multiple tokens, with $h_{C_{ij}}$ being the output for the j-th token of column i. Our context reinforcing layer tries to learn a new representation r_{C_i} for each column by strengthening the original encoder output with the global context information captured in $h_{\texttt{[CXT]}}$.

Denoting the number of tokens in column i as n_i, context reinforcing layer first summarizes each column by computing

$$h_{C_i} = \sum_{t=1}^{n_i} \alpha_{it} h_{C_{it}} \tag{1}$$

where $\alpha_{it} := \textsc{softmax}(s_{it})$. The alignment model s_{it} tells how well t-th token of column i matches the global context, and is defined as

$$s_{it} = f\big(U h_{\texttt{[CTX]}}, V h_{C_{it}}\big). \tag{2}$$

Both $U, V \in \mathbf{R}^{m \times d}$, and we use simple dot product for f.

The final representation r_{C_i} is obtained by simply adding $h_{\texttt{[CTX]}}$ and h_{C_i} from Equation 1. Although computation in Equation 1 already encourages context by weighting different tokens according to

the match, adding $h_{\texttt{[CTX]}}$ gives the schema representation a better alignment with the particular natural language question being asked.

Context reinforcing layer is not the only place that introduces global context to schema. For example, there is already some degree of context being captured in the encoder. We believe its influence in this manner is not strong enough, therefore should be reinforced as a separate step.

While context reinforcing layer and column attention introduced in Xu et al. (2017) share a similar goal of better aligning natural language question and table schema, they differ significantly in both technical solution and the role played in the entire architecture. Column attention changes h_{q_i} by signifying which query words are most relevant to each column. It does this for every column in the table, and is tied closely with output layer rather than as a separate module. Context reinforcing layer, on the other hand, believes BERT style encoder already performs well on natural language side, and tries to come up with a better representation for schema. It uses only context information captured in `[CTX]` to update the schema part. As a result, the modeling of each sub-task in our output layer has been greatly simplified, as opposed to SQLNet.

2.3 Output Layer

The output layer composes the SQL program from both encoder outputs $h_{\texttt{[CXT]}}, h_{q_1}, \cdots, h_{q_n}$ and outputs of context reinforcing layer $r_{C_1}, \cdots, r_{\texttt{[EMPTY]}}$. Like Xu et al. (2017) and Hwang et al. (2019), the task is decomposed into 6 sub-tasks, each predicting a part of the final SQL program, with task dependency shown in Figure 1. Unlike their models, X-SQL enjoys a much simplified structure thanks to context reinforcing layer.

Task S-COL predicts the column for the SELECT clause and task S-AGG predicts the aggregator for the column. The probability of column C_i being chosen for the SELECT statement is modeled as

$$p^{\text{S-COL}}(C_i) = \text{SOFTMAX}\big(W^{\text{S-COL}} r_{C_i}\big)$$

with $W^{\text{S-COL}} \in \mathbf{R}^{1\times d}$, and the probability of aggregator is conditioned on the selected column and computed as

$$p^{\text{S-AGG}}(A_j|C_i) = \text{SOFTMAX}\big(W^{\text{S-AGG}}[j,:] r_{C_i}\big)$$

where $W^{\text{S-AGG}} \in \mathbf{R}^{6\times d}$ with 6 being the number of aggregators. Note, S-COL and S-AGG depend on r_{C_i} only, as opposed to both query and schema in previous work.

The remaining 4 tasks W-NUM, W-COL, W-OP and W-VAL together determine WHERE part. Task W-NUM finds the number of where clauses using $W^{\text{W-NUM}} h_{\texttt{[CTX]}}$, and is modeled as a classification over four possible labels each representing 1 to 4 where clauses in the final SQL. It doesn't predict the empty where clause case, but instead delegates it to W-COL through the Kullback-Leibler divergence explained in Section 2.4. Task W-COL outputs a distribution over columns using

$$p^{\text{W-COL}}(C_i) = \text{SOFTMAX}\big(W^{\text{W-COL}} r_{C_i}\big) \tag{3}$$

and based on the number from W-NUM, top scoring columns are selected for the where clauses. Task W-OP is modeled similar as S-AGG, i.e. $p^{\text{W-OP}}(O_j|C_i) = \text{SOFTMAX}\big(W^{\text{W-OP}}[j,:] r_{C_i}\big)$ to choose the most likely operator for the given where column. Predicting value for where clause (task W-VAL) is formulated as predicting a span of text from query, which simply becomes predicting the beginning and the end position of the span using

$$p^{\text{W-VAL}}_{\text{start}}(q_j|C_i) = \text{SOFTMAX}\ g\big(U^{\text{start}} h_{q_j} + V^{\text{start}} r_{C_i}\big)$$

and

$$p^{\text{W-VAL}}_{\text{end}}(q_j|C_i) = \text{SOFTMAX}\ g\big(U^{\text{end}} h_{q_j} + V^{\text{end}} r_{C_i}\big).$$

where $g(x) = Wx + b$. The parameters $W^{\text{W-NUM}}$, $W^{\text{W-COL}}$, $W^{\text{W-OP}}$ are in $\mathbf{R}^{4\times d}$, $\mathbf{R}^{1\times d}$ and $\mathbf{R}^{3\times d}$ respectively, with number of possible operators being 3. Parameters $U^{\text{start}}, V^{\text{start}}, U^{\text{end}}, V^{\text{end}} \in \mathbf{R}^{m\times d}$ and different g functions are learned for predicting start and end.

2.4 TRAINING AND INFERENCE

During training, we optimize the objective which is a summation over individual sub-task losses. We use cross entropy loss for task S-COL, S-AGG, W-NUM, W-OP and W-VAL. The loss for W-COL is defined as the Kullback-Leibler (KL) divergence between $D(Q||P^{\text{W-COL}})$, where $P^{\text{W-COL}}$ is modeled by Equation 3. Distribution Q from ground truth is computed as follows:

- If there is no where clause, $Q_{\texttt{[EMPTY]}}$ receives probability mass 1 for special column `[EMPTY]`,
- For $n \geq 1$ where clauses, each where column receives probability mass of $\frac{1}{n}$.

Inference is relatively straightforward except for the W-COL. If the highest scoring column is the special column `[EMPTY]`, we ignore the output from W-NUM and return empty where clause. Otherwise, we choose top n non-`[EMPTY]` columns as indicated by W-NUM and W-COL.

3 EXPERIMENTS

We use the default train/dev/test split of the WikiSQL dataset. Both logical form accuracy (the exact match of SQL queries) and execution accuracy (ratio of predicted SQL queries that lead to correct answer) are reported. The logical form accuracy is the metric we optimize during training.

Our X-SQL implementation is built on top of MT-DNN, which extends BERT implementation in PyTorch. We initilaize encoder with MT-DNN and then continue to learn all the parameters in Section 2. Specifically, we set batch size as 32, and use Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The global learning rate is $2e-5$ except we use a learning rate warm-up schedule for the first 1000 steps. Dropout rate is set to be 0.2 for all attention and fully connected layers. We train 10 epochs and use the dev set to pick the best model based on its logical form accuracy.

Table 1: Main Results

Model	Dev		Test	
	Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
Seq2SQL (Zhong et al., 2017)	49.5	60.8	48.3	59.4
SQLNet (Xu et al., 2017)	63.2	69.8	61.3	68.0
Coarse2Fine (Dong & Lapata, 2018)	72.5	79.0	71.7	78.5
IncSQL (Shi et al., 2018)	49.9	84.0	49.9	83.7
SQLova (Hwang et al., 2019)	81.6	87.2	80.7	86.2
X-SQL	**83.8**	**89.5**	**83.3**	**88.7**
IncSQL + EG	51.3	87.2	51.1	87.1
SQLova + EG	84.2	90.2	83.6	89.6
X-SQL + EG	**86.2**	**92.3**	**86.0**	**91.8**

Table 1 includes results both with and without execution guidance applied during inference (Wang et al., 2017). We compare our results with the most recent work in WikiSQL leaderboard, including the previous state-of-the-art SQLova model. X-SQL is shown to be consistently and significantly better across all metrics and achieves the new state-of-the-art on both dev and test set. Without EG, X-SQL delivers an absolute 2.6% (83.3 vs. 80.7) improvement in logical form accuracy and 2.5% improvement in execution accuracy on test set. Even with EG, X-SQ is still 2.4% (86.0 vs. 83.6) better in logical form accuracy, and 2.2% (91.8 vs. 89.6) better in execution accuracy. It is worth noting that X-SQL+EG is the first model that surpasses the 90% accuracy on test set. On the other hand, for dev set human performance is estimated to be 88.2% according to Hwang et al. (2019) based on random sampling. X-SQL is the first model better than human performance without the help of execution guidance.

Table 2 reports the accuracy for each sub task, which demonstrates consistent improvement. In particular, task W-COL shows an absolute 1.1% gain without EG and 1.7% with EG. We attribute

this to our new approach of formalizing the where column prediction problem as a list-wise ranking problem using KL divergenceAnother significant improvement is the W-VAL task, with an absolute 1.2% gain without EG and 2.0% with EG. This can be partially attributed to the column set prediction (i.e. W-COL) improvement, since the value generation depends highly on the predicted column set for the where clause.

Table 2: Sub-module Results. Models marked with $*$ are obtained by running SQLova code at https://github.com/naver/sqlova.

Model	S-COL	S-AGG	W-NUM	W-COL	W-OP	W-VAL
Dev						
SQLova	97.3	90.5	98.7	94.7	97.5	95.9
X-SQL	97.5	90.9	99.0	96.1	98.0	97.0
SQLova + EG*	97.3	90.7	97.7	96.0	96.4	96.6
X-SQL + EG	97.5	90.9	99.0	97.7	98.0	98.4
Test						
SQLova	96.8	90.6	98.5	94.3	97.3	95.4
X-SQL	97.2	91.1	98.6	95.4	97.6	96.6
SQLova + EG*	96.5	90.4	97.0	95.5	95.8	95.9
X-SQL + EG	97.2	91.1	98.6	97.2	97.5	97.9

4 CONCLUSION

We propose a new model called X-SQL and demonstrate its exceptional performance on the WikiSQL task. The novelty of X-SQL mainly lies in two aspects. First, X-SQL reinforces the contextual information on the column schema to build a new column-wise representation for downstream tasks. Second, to tackle the optional column set prediction problem in NL2SQ, X-SQL models it as a list-wise ranking problem with the introduction of the `[EMPTY]` column. All of these help X-SQL to obtain a new state-of-the-art performance in WikiSQL across all metrics.

How to better combine the pretrained representation with the syntactic structure of a general semantic parsing problem would be a future problem to be explored. Meanwhile, we plan to apply X-SQL to other NL2SQL datasets to study its generalization capability.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, August 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Li Dong and Mirella Lapata. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33–43, 2016.

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 731–742, 2018.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 351–360, 2018.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on WikiSQL with table-aware word contextualization. Technical report, 2019. URL https://ssl.pstatic.net/static/clova/service/clova_ai/research/publications/SQLova.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Solomon Kullback. Letter to the editor: The kullback-leibler distance. 1987.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1512–1523, 2011.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 3104–3112, 2014.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashish G. Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 1507–1514, August 2011.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pp. 2692–2700, 2015.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018a.

Chenglong Wang, Marc Brockschmidt, and Rishabh Singh. Pointing out SQL queries from text. Technical Report MSR-TR-2017-45, November 2017. URL https://www.microsoft.com/en-us/research/publication/pointing-sql-queries-text/.

Chenglong Wang, Po-Sen Huang, Oleksandr Polozov, Marc Brockschmidt, and Rishabh Singh. Execution-guided neural program decoding. In *ICML workshop on Neural Abstract Machines & Program Induction v2 (NAMPI)*, 2018b.

Yuk Wah Wong and Raymond J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, June 2007.

Xiaojun Xu, Chang Liu, and Dawn Song. SQLNet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 588–594, 2018.

Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pp. 658–666, 2005.

Luke S. Zettlemoyer and Michael Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 678–687, 2007.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.