X-SQL: reinforce schema representation with context

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

Abstract

In this work, we present X-SQL, a new network architecture for the problem of parsing natural language to SQL query. X-SQL proposes to enhance the structural schema representation with the contextual output from BERT-style pre-training model, and together with type information to learn a new schema representation for down-stream tasks. We evaluated X-SQL on the WikiSQL dataset and show its new state-of-the-art performance.

1 Introduction

Question Answering (QA) is among the most active research areas in natural language processing recently. In this paper, we are interested in QA over structured databases. This is usually done by mapping natural language question to a SQL query representing its meaning, a problem known as semantic parsing, followed by executing the SQL query against databases to obtain the answer.

The largest annotated dataset for this problem is WikiSQL (Zhong et al., 2017). Early work adopts a neural sequence-to-sequence approach with attention and copy mechanism, while recent focus has been on incorporating the SQL syntax into neural models. Xu et al. (2017) and Yu et al. (2018a) capture the syntax via dependency between different prediction modules. Dong and Lapata (2018) and Finegan-Dollak et al. (2018) use a slot filling approach where syntactic correctness is ensured by predefined sketches.

Recent advances in language representation modeling (Radford et al., 2018; Devlin et al., 2018) demonstrate the value of transfer learning from large external data source. For WikiSQL, the work of Hwang et al. (2019) has shown significant improvement with such pre-training techniques. In view of this trend, we propose X-SQL, an improved pre-training based neural model with contributions from the following three perspectives.

There are two types of textual information to be considered for this problem: one is the unstructured natural language query, and the other is the structured data schema. Previous work either models them independently or builds cross-attention between them (Xu et al., 2017; Shi et al., 2018). While the structured information such as table column is relatively stable, natural language queries are highly variable. We leverage an existing pre-trained model named MT-DNN (Liu et al., 2019) to capture this variation and summarize the unstructured query into a global context representation, which is then being used to enhance the structured schema representation for downstream tasks. To the best of our knowledge, this is the first attempt to incorporate BERT-style contextual information into a problem-dependent structure, and build a new representation to better characterize the structure information.

Second, part of SQL syntax is bounded to the type of structured data schema. For example, aggregator MIN only appears with numerical column, and operator $>$ doesn't pair with string typed column. We incorporate schema type information in two places. Section 2.1 describes type embedding with a modification of pretrained language representation, and Section 2.3 shows how to further improve certain sub-tasks with a separately learned type embedding.

Lastly, we observe previous approach using multiple binary classifiers for where clause prediction cannot effectively model the relationship between columns, since each classifier is optimized independently, and their outputs are not directly comparable. To tackle this issue, X-SQL takes a list-wise global ranking approach by using the Kullback-Leibler divergence as its objective to bring all columns into a comparable space (Section 2.4).

2 Neural Architecture

The overall architecture consists of three layers: sequence encoder, context enhancing schema encoder and output layer.

2.1 Sequence Encoder

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

- A special empty column `[EMPTY]` is appended to every table schema. Its usage will become clear in Section 2.4.
- Segment embeddings are replaced by 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-stream NLP tasks.

Note, we rename `[CLS]` output of BERT to `[CTX]` in the following sections and Figure 1. This is to emphasize that context information is being captured there, rather than a representation for down-stream tasks.

In addition to these changes, our encoder differs from SQLova with NL2SQL layer (Hwang et al., 2019) in the following important way: while SQLova still runs bi-LSTM/column attention on top of the encoder, our architecture enjoys a much simpler yet powerful design for consequent layers, which we believe is largely attributed to a better alignment of BERT with the problem.

2.2 Context Enhanced Schema Encoder

Let $h_{\texttt{[CTX]}}, h_{q_1}, \cdots, h_{q_n}, h_{\texttt{[SEP]}}, h_{C_{11}}, \cdots, h_{\texttt{[SEP]}}, h_{C_{21}}, \cdots h_{\texttt{[SEP]}}, \cdots, h_{\texttt{[EMPTY]}}, h_{\texttt{[SEP]}}$ denote the output from the encoder, each of dimension d. Each question token is encoded as h_{q_i}, followed by $h_{C_{ij}}$ which encodes the j-th token from column i since each column name may contain multiple tokens. Our context enhanced schema encoder (Figure 1(a)) tries to learn a new representation h_{C_i} for each column i by strengthening the original encoder output with the global context information captured in $h_{\texttt{[CTX]}}$.

Denoting the number of tokens in column i as n_i, the schema encoder 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

$$s_{it} = f\Big(U h_{\texttt{[CTX]}}/\sqrt{d}, V h_{C_{it}}/\sqrt{d}\Big).$$

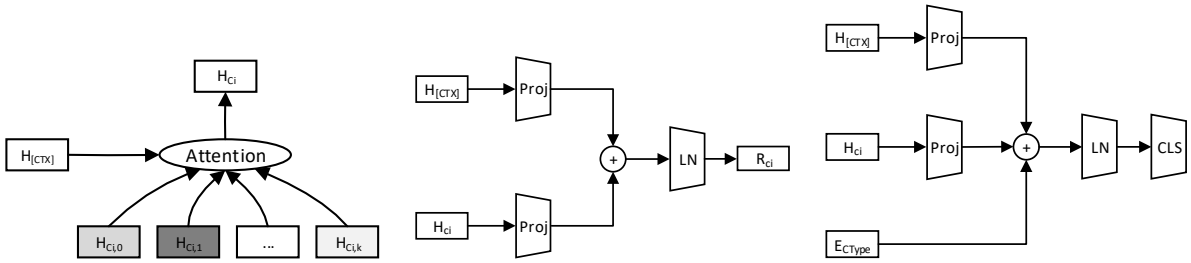
Both $U, V \in \mathbf{R}^{m \times d}$, and we use simple dot product for f.

While there is already some degree of context being captured in the output from sequence encoder, such influence is limited as self-attention tends to focus on only certain regions. On the other hand, the global contextual information captured in `[CTX]` is diverse enough, thus is used to complement the schema representation from sequence encoder.

Although context enhanced schema encoder 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 overall architecture. Column attention changes h_{q_i} by signifying which query words are most relevant to each column. It does so for every column in the table, and columns are processed independent of each other. Context enhanced schema encoder, on the other hand, believes BERT style sequence encoder already performs well on the natural language side, and tries to come up with a better representation for schema. It uses only contextual information captured in `[CTX]` to update the schema part. Since `[CTX]` also contains information from other parts of the schema, columns are no longer updated independently.

2.3 Output Layer

The output layer composes the SQL program from both sequence encoder outputs $h_{\texttt{[CTX]}}, h_{q_1}, \cdots, h_{q_n}$ and context enhancing schema encoder outputs $h_{C_1}, h_{C_2}, \cdots, h_{\texttt{[EMPTY]}}$. Similar to 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.



(a) Context enhancing schema encoder (Equation 1).

(b) Sub-network that modulates schema representation with context (Equation 2).

(c) Neural network for select column aggregator (Task S-AGG, Equation 3).

Figure 1: Components of X-SQL neural network model architecture.

Unlike their models, X-SQL enjoys a much simplified structure due to context enhancement.

We first introduce a task dependent sub-network that modulates the schema representation h_{C_i} using context $h_{\texttt{CTX}}$. Specifically,

$$r_{C_i} = \text{LayerNorm}\big(U' h_{\texttt{[CTX]}} + V' h_{C_i}\big). \quad (2)$$

Different from Equation 1, this computation is done separately for each sub-task, to better align the schema representation with the particular part of natural language question that each sub-task should focus on.

The first task, S-COL, predicts the column for the SELECT clause. 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 $W^{\text{S-COL}} \in \mathbf{R}^{1\times d}$. Note, S-COL depends on r_{C_i} only, as opposed to both query and schema in previous work.

The second task, S-AGG, predicts the aggregator for the column selected by SELECT. To enhance the intuition that aggregator depends on the selected column type, e.g. MIN aggregator doesn't go with a string typed column, we explicitly add column type embedding to the model. The probability of the aggregator is computed as

$$p^{\text{S-AGG}}(A_j|C_i) = \text{SOFTMAX}\Big(W^{\text{S-AGG}}[j,:] \times \text{LAYERNORM}\left(U'' h_{\texttt{[CTX]}} + V'' h_{C_i} + E^T_{C_i}\right)\Big) \quad (3)$$

where $W^{\text{S-AGG}} \in \mathbf{R}^{6\times d}$ with 6 being the number of aggregators. Different from other sub-tasks we use h_{C_i} here rather than r_{C_i}, as we incorporate column type in a similar way to Equation 2. Type embedding $E^T_{C_i}$ is learned separately from the one used by the sequence encoder.

The remaining 4 tasks W-NUM, W-COL, W-OP and W-VAL together determine the 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, which is delegated to W-COL through the Kullback-Leibler divergence as explained in Section 2.4. Task W-COL outputs a distribution over columns using

$$p^{\text{W-COL}}(C_i) = \text{SOFTMAX}\big(W^{\text{W-COL}} r_{C_i}\big) \quad (4)$$

and based on the number from W-NUM, top scoring columns are selected for the where clauses. Task W-OP chooses the most likely operator for the given where column using $p^{\text{W-OP}}(O_j|C_i) = \text{SOFTMAX}\big(W^{\text{W-OP}}[j,:] r_{C_i}\big)$. Intuitively operator also depends on the column type, and could be modeled in the same way as Task S-AGG in Equation 3. However, we didn't observe improvement during experiments, therefore we prefer to keep the original simple model. Model 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.

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 $g(x) := Wx + b$. 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.

Table 2: Dev/test results for each sub-module. $*$ means results obtained by running SQLova code from github.

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

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 4. 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 a 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 (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.

Table 1 includes results both with and without execution guidance (EG) applied during inference (Wang et al., 2018b). We compare our results with the most recent work of 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-SQL is still 2.4% better in logical form accuracy, and 2.2% 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). X-SQL is the first model better than human performance without the help of execution guidance.

Table 1: Logical form (*lf*) and execution (*ex*) accuracy.

Model	Dev		Test	
	Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
SQLNet	63.2	69.8	61.3	68.0
SQLova	81.6	87.2	80.7	86.2
X-SQL	**83.8**	**89.5**	**83.3**	**88.7**
SQLova + EG	84.2	90.2	83.6	89.6
X-SQL + EG	**86.2**	**92.3**	**86.0**	**91.8**

Table 2 reports the accuracy for each sub-task, and 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 as a list-wise ranking problem using KL divergence. Another significant improvement is the W-VAL task, with an absolute 1.2% gain without EG and 2.0% with EG. This partly results from the column set prediction (i.e. W-COL) improvement as well, since the value generation depends highly on the predicted column set for the where clause.

4 Conclusion

We propose a new model X-SQL, demonstrate its exceptional performance on the WikiSQL task, and achieve new state-of-the-art across all metrics. While the contribution around loss objective may be bounded by the specific SQL syntax that WikiSQL uses, how contextual information is leveraged and how schema type is used can be immediately applied to other tasks that involve pre-trained language model for structured data. Future work includes experimenting with more complex dataset such as Spider (Yu et al., 2018b).

References

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

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

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

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. Technical report.

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

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

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. 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*, pages 353–355.

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

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

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. 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)*, pages 588–594.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

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