Service-oriented Text-to-SQL Parsing

Wangsu Hu, Jilei Tian
BMW Technology Corporation / 540 W Madison St 2400, Chicago, IL 60661
wangsu.hu@bmwna.com, jilei.tian@bmwna.com

Abstract

The information retrieval from relational database requires professionals who has an understanding of structural query language such as SQL. TEXT2SQL models apply natural language inference to enable user interacting the database via natural language utterance. Current TEXT2SQL models normally focus on generating complex SQL query in a precise and complete fashion while certain features of real-world application in the production environment is not fully addressed. This paper is aimed to develop a service-oriented Text-to-SQL parser that translates natural language utterance to structural and executable SQL query. We introduce a algorithmic framework named Semantic-Enriched SQL generator (SE-SQL) that enables flexibly access database than rigid API in the application while keeping the performance quality for the most commonly used cases. The qualitative result shows that the proposed model achieves 88.3% execution accuracy on WikiSQL task, outperforming baseline by 13% error reduction. Moreover, the framework considers several service-oriented needs including low-complexity inference, out-of-table rejection, and text normalization.

1 Introduction

The relational database stores a vast of information then support applications in various areas. API and query language normally enforce access to this data. To help retrieving information from database based on utterance, one conventional solution is applying Natural Language Understanding (NLU) model firstly to extract entities as attributes to call the downstream APIs. The extracted entities fulfill the required slot in the pre-defined query templates to retrieve information from the database. Such a rule-based mechanism ensures the input value at runtime while limiting the information retrieval in two major aspects. First, the entity extraction might be constrained under closed domains. It is challenging to apply one trained NLU model when the database is modified, e.g., the new schema or new table. Second, creating or modifying the query template requires numerous human labor and limits the requested query by fixed knowledge of API's designer. For example, as query the weekly average temperature via date range constraint and aggregation operation, users won't get the result if there is no such pre-defined query template in the scheme. To address such issues, TEXT2SQL models (Liang and Potts, 2015; Zhong et al., 2017; Xu et al., 2017; Dong and Lapata, 2018; Yu et al., 2019; Dong et al., 2019; Bogin et al., 2019; Lee, 2019; Hwang et al., 2019; Guo and Gao, 2019) are aimed to map natural language utterance to executable SQL query with or without the known database. One example of TEXT2SQL task can be found in Table 1.

This paper aims to develop a service-oriented Text-to-SQL parser translating natural language utterance to structural and executable SQL query without the limitation above. We introduce an algorithmic framework named Semantic-Enriched SQL generator (SE-SQL) with the following key contributions:

Enable flexibly to access the database while keeping the performance quality for the most commonly used cases, such as SELECT, FROM, and WHERE, including aggregators and operators. Therefore, the coverage of the primary query is supported.

Consider service-oriented needs, including low-complexity inference and confidence measure for quality when dealing with real-life scenarios.

Better user experience is obtained by improved algorithm performance from both questions by text normalization and schema by semantic enrichment. Documents and codes can be found at: github.com/nicholasadam/SESQL.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2218–2222

Sports			
Player	Number	Nationality	City
AL	21	US	Chicago
GL	32	US	New York
BM	99	US	Chicago

Question: Who is the player wearing 32 and what country is he from?
SQL: SELECT Player, Nationality FROM Sports WHERE No.=32
Answer: GL from US

Table 1: Example of TEXT2SQL task.

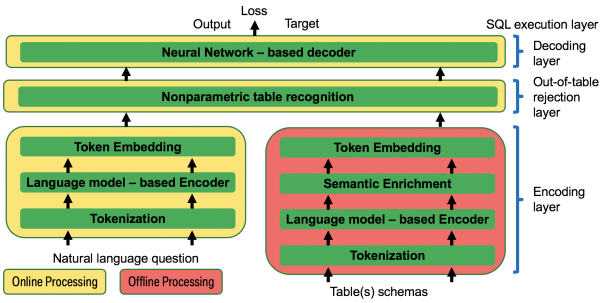

Figure 1: Algorithm Framework - High level

2 Semantic-enriched SQL generator

For the natural language questions collected across users' utterances and the table schema collected from the database, the proposed algorithmic framework processes such two types of data into SQL query, organized by context-aware Question Encoding Layer, semantic-enriched Table Schema Encoding Layer, out-of-table prediction layer, TEXT2SQL decoding layer, and SQL execution layer as shown in Figure 1.

2.1 Question Encoding Layer

We leverage BERT as the language model (Devlin et al., 2018) for encoding the natural language question. Each question input is encoded as:

$$Q = [CLS], q_1, ..., q_L, [SEP]$$

$$E_Q = BERT(Q) = E_{[CLS]}, E_{q_1}, ..., E_{q_L}, E_{[SEP]}$$

where q_i is the i-th token of question Q, L is the total number of question tokens. $[CLS]$, $[SEP]$ are special token used by BERT to indicate the start and end index of input. E_Q is the generated tokens' embedding with the same length as Q. Given feature size S of embedding vector, the generated question embedding is a (L, S)-size matrix. The output from the final two layers of BERT are concatenated and fed into TEXT2SQL layer.

2.2 Table Schema Encoding Layer

Similar to question encoding, each table schema input is encoded as below.

$$H = [CLS], h_1, [SEP], ..., h_L, [SEP]$$

$$E_H = BERT(H) = E_{[CLS]}, E_{h_1}, E_{[SEP]}, ..., E_{h_L}, E_{[SEP]}$$

where h_j is the j-th token of table schema, L is the total number of schema tokens. We use $[SEP]$ to separate tokens in the schema.

Low-complexity implementation: Unlike conventional TEXT2SQL models (Hwang et al., 2019; Guo and Gao, 2019), question and table schema are separately encoded for two reasons: First, as BERT features contextual word representation, we generate question tokens only with its contextual information. Second, as the question part is online fed into the encoder, separate encoding of the schema part can be done offline to reduce the inference time online greatly. In reality, we have just one question to be encoded per request, while taking all encoded table schema into account. Otherwise, we have to encode pairwise instance as many as the number of tables that makes the complexity higher.
Semantic enrichment: Another key challenge is the alignment between user expression and schema expression towards the target object. A user might ask a question flexibly while headers are designed based on a DBMS-neutral guide for naming common objects. Here we enriched schema to reduce the mismatch condition via crawling alias and table contents for headers. Then the language model and transformation network are applied to produce embedding sharing the same length as headers. The final semantic representation of table schema is calculated as below.

$$E_{H_{SE}} = E_H + E_{H_{synonyms}} + E_{H_{content}}$$

where $E_{H_{SE}}$ is header's embedding, $E_{H_{synonyms}}$ is header synonyms embedding, $E_{H_{content}}$ is selected table content embedding. Compared to existing content-enhance method (Guo and Gao, 2019), the leverage of contents is constrained as it cannot afford the copy of whole contents during inference for real-world solutions. Moreover, semantic enrichment work can be pre-processing offline.

2.3 TEXT2SQL

TEXT2SQL layer is designed on top of question and schema encoding layers. Xu et al.(Xu et al., 2017) applied sketch-based syntactic constraints in

query generation as below:

$$SELECT : [(agg1, scol1), ...] FROM : [(table1)]$$

$$WHERE : [(wcol1, op1, val1), (wcol2, op2, val2), ...]$$

where agg, scol represent aggregator and header in SELECT clause, wcol/op/val represent header, operator, and value in WHERE clause respectively.

<FROM> clause: Towards a real-world application, out-of-table rejection is required to recognize the table in backend database for the question input. One nonparametric model is introduced on top of question embedding E_Q and schema embedding E_H via cosine distance measurement. We calculate the similarity between question E_{q_i} and schema E_H. Then we conduct minimum pooling along dimension j and take average value as the overall similarity from question to schema.

$$Distance_{i->H} = minpooling_j \sum_S E_{q_i} E_{h_j} / \sqrt{E_{q_i}^2} / \sqrt{E_{h_j}^2}$$

$$Distance_{Q->H} = \sum_{L_Q} Distance_{i->H} / L_Q$$

where $minpooling_j$ samples the minimum value along dimension j. The similarity from schema to question is calculated using the same method to get $Distance_{H->Q}$.

By adding two normalized distances to represent the pairwise similarity score, we reject the prediction if the score is lower than the threshold $R_{threshold}$.

$$Distance_{Q<->H} = Distance_{H->Q} + Distance_{Q->H}$$

$$Reject = Distance_{Q<->H} > R_{threshold}$$

<SELECT> and <WHERE> clause: Tasks are to predict column(s) from schema headers and corresponding aggregator(s)/operator(s)/value(s) tied to column(s).

Column prediction predicts column(s) via attention and Learn-To-Rank method.

$$s(q|h) = E_h^T W E_q; p(q|h) = softmax(s(q|h))$$

$$H_h = \sum_i^N p(q|h) E_q$$

$$sc_h = maxpooling([E_h \cdot WH_h]); p_h = sigmoid(sc_h)$$

where W is the required transformation, H_h is context vector of header h, sc_h is column score, $[;]$ denotes concatenation operation, $maxpooling$ samples max value, and p_h is the probability of selecting header h. We choose the top k header(s) among candidates by predicting number of columns k.

$$k = argmax(softmax(E_{Q,CLS} W))$$

Agg/Op prediction predicts aggregator and operator tied to SELECT-column and WHERE-column, respectively.

$$p_z = Wtanh([E_q; W_{sc_h}); z = argmax(p_z)$$

where z is the z-th aggregator/operator.

Value parsing predicts start and end token index from question for the given header and operator op.

$$idx = argmax(softmax(WE_q sc_h V_{op}))$$

where V_{op} is the one-hot vector of operation choice.

Text Normalization After predicting the start and end index of parsed value, we conduct the named entity normalization to generate a regularized representation for objects such as time and range during the inference. As we found in the real application scenario, the WHERE-value parser contributes considerable amounts of "no result found" cases, although the other sub-modules have the correct prediction. Datetime range is one of the most happened entities in utterances. An un-normalized parsing might lead to a null result. Therefore, we leverage the named entity recognition enabler to parse normalized WHERE-value. For example, "till (date)" will be translated to "WHERE-column (date) WHERE-operator (<=) WHERE-value (YYYY-MM-DD)".

3 Experiments and Results

Experiment Setup As motivated by developing a service-oriented solution, we determine the benchmark dataset based on the following criteria. First, the benchmark task should meet the basic requirement of real-world TEXT2SQL application, e.g., Spider (Yu et al., 2018) and WikiSQL (Zhong et al., 2017). Second, the numbers of question-table-query instances should be large enough to ensure the generality of the model. Third, the generated SQL should cover the potential sessions in the production environment, while high complexity parser might introduce the processing latency in real-world inference procedures. Here we train and evaluate our models on WikiSQL ver. 1.1 that was firstly introduced by Zhong et al. (Zhong et al., 2017). WikiSQL contains a large corpus of question-table-SQL instances from Wikipedia, then divided into train (55k instances), dev (8k instances), and test sets (16k instances). We applied a negative sampling method to create the sub-dataset for out-of-table rejection. Two metrics are applied regarding models and WikiSQL dataset: 1, logical-form accuracy(LF): exact string match with the ground truth query. 2, execution accuracy(EX): the executed result match.

Before training, BERT-based encoding layers are used. During training, the TEXT2SQL layer is fine-

Model	LF (%)	EX (%)
WikiSQL dev set		
SQLNet	63.2	69.8
SQLova(baseline)	81.6	87.2
SE-SQL(ours)	82.1 (+0.5)	87.3 (+0.1)
+ Semantic enrich	83.3 (+1.7)	88.3 (+1.1)
Error reduction	13.0	8.6
WikiSQL test set		
SQLNet	61.3	68.0
SQLova(baseline)	80.7	86.2
SE-SQL(ours)	81.9 (+1.2)	87.2 (+1.0)
+ Semantic enrich	82.5 (+1.8)	87.8 (+1.6)
Error reduction	9.3	11.6

Table 2: Overall performance of WikiSQL task.

tuned with ADAM optimizer with the learning rate of 0.001. The batch size is set to 16. Both questions and table schema are tokenized into the sub-word level by WordPiece tokenizer (Devlin et al., 2018). NLTK library (Loper and Bird, 2002) is applied to generate synonyms of headers in the table schema. Set $R_{threshold}$ as 1.1 and negative sampling size as 11 to do out-of-table rejection. Meanwhile, the beam search method (Wang et al., 2018) considered an execution guide for the generated query, although widely applied for NLI evaluation, was not used due to maintaining the low communication and processing latency between application and database. Therefore, the reported performance can be treated as the lower-bound one. Meanwhile, after reviewing the open-sourced TEXT2SQL literature, we selected one of the state-of-the-art models named "SQLova" (Hwang et al., 2019) as the baseline. The author claims that SQLova is the first natural-language-to-SQL model to achieve human performance in the WikiSQL task.

Qualitative performance The result shows that SE-SQL outperforms the baseline by a promising margin, as shown in Table 2. Moreover, to understand the performance of SE-SQL in detail, the breakdown accuracy of each sub-module under the logical form metric was shown in Appendix. The result indicates that the proposed framework shows a promising potential of being a TEXT2SQL solution in terms of model performance.

Semantic enrichment As shown in Table 2, semantic enrichment contributed a nearly 1% performance improvement along with better user experience. Here, we let each header have 3 numbers of synonyms and 2 numbers of table contents.

Error analysis We randomly selected 50 samples of mismatches under the logical form metric from WikiSQL dev set between the ground-truth query and the generated query by SE-SQL. It found that 6 numbers of ground truth errors (the annotation failed), 10 numbers of answerable errors (the question cannot be answered based on the provided table) and 34 numbers of prediction error (the question can be answered but the prediction is wrong) exist.

Inference time SE-SQL applied separate-encoding of question and table schema while the conventional TEXT2SQL models applied joint-encoding of question-table pairwise instance. Regarding the inference time consumption, we reported both schemes' service response time under the same testing environment. The system configuration is attached in Appendix. The result shows that SE-SQL achieved 95 seconds in total for 15,870 requests (6ms per query) while the other spent 2,698 seconds (170ms per query).

Out-of-table rejection Regarding confidence measure in from-clause prediction, the proposed non-parametric prediction model achieved 85% true positive, 55% true negative, 6% false positive, and 6% false negative. When having rejection enabled, it rejected 9% cases. Among the 91% accepted cases, accuracy is improved from the original 85% to 87%. Moreover, the setting of the threshold offers the opportunity to adjust the performance.

4 Conclusion

In this paper, we proposed a novel algorithmic framework named SE-SQL that not only enables access database flexibly while keeping the performance quality for the most commonly used cases but more importantly offer the solution towards meeting product requirement from the lab research. We reconstructed the previous TEXT2SQL framework to introduce question-table separate contextualization in a low-complexity fashion. An improved algorithm performance obtains better user experience that the proposed model outperforms the baseline by a 13% error reduction in the WikiSQL task. Moreover, the properties, including out-of-table rejection, confidence measurement for quality, and fast online inference, are considered toward the production environment.

References

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-sql parsing. *arXiv preprint arXiv:1908.11214*.

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. *arXiv preprint arXiv:1805.04793*.

Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-anonymous encoding for text-to-sql generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5408–5417.

Tong Guo and Huilin Gao. 2019. Content enhanced bert-based text-to-sql generation. *arXiv preprint arXiv:1910.07179*.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Dongjun Lee. 2019. Clause-wise and recursive decoding for complex and cross-domain text-to-sql generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6047–6053.

Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annu. Rev. Linguist.*, 1(1):355–376.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

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, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Appendix

Sub-modular	Dev	Test
select-column	98.5	98.3
select-aggregator	91.5	91.1
where-column	94.9	95.1
where-operator	98.0	97.9
where-value	97.0	97.1

Table 3: Breakdown result of Logical Form accuracy under WikiSQL dev set

Category	Description
Machine instance	Azure DSVM NC6s
Core	6
RAM	112GB
GPU	V100 (single GPU)
Inference samples	15,870

Table 4: System configuration