Received February 20, 2019, accepted March 6, 2019, date of publication March 13, 2019, date of current version April 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904720

NADAQ: Natural Language Database Querying Based on Deep Learning

BOYAN XU[1], RUICHU CAI[1], ZHENJIE ZHANG[2], XIAOYAN YANG[2], ZHIFENG HAO[1,3], ZIJIAN LI[1], AND ZHIHAO LIANG[1]
[1]Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, China
[2]Singapore R&D, Yitu Technology Pte Ltd., Singapore 117372
[3]School of Mathematics and Big Data, Foshan University, Foshan 528000, China

Corresponding author: Ruichu Cai (cairuichu@gmail.com)

This work was supported in part by the NSFC-Guangdong Joint Found under Grant U1501254, in part by the Natural Science Foundation of China under Grant 61876043 and 61472089, in part by the Natural Science Foundation of Guangdong under Grant 2014A030306004 and Grant 2014A030308008, in part by the Science and Technology Planning Project of Guangdong under Grant 2015B010108006 and Grant 2015B010131015, in part by the Guangdong High-level Personnel of Special Support Program under Grant 2015TQ01X140, in part by the Pearl River S&T Nova Program of Guangzhou under Grant 201610010101, and in part by the Science and Technology Planning Project of Guangzhou under Grant 201604016075.

ABSTRACT The high complexity behind SQL language and database schemas has made database querying a challenging task to human programmers. In this paper, we present our new natural language database querying (NADAQ) system as an alternative solution, by designing new translation models smoothly fusing deep learning and traditional database parsing techniques. On top of the popular encoder-decoder model for machine translation, NADAQ injects new dimensions of schema-aware bits associated with the input words into encoder phase and adds new hidden memory neurons controlled by the finite state machine for grammatical state tracking into the decoder phase. We further develop new techniques to enable the augmented neural network to reject queries irrelevant to the contents of the target database and recommend candidate queries reversely transformed into natural language. NADAQ performs well on real-world database systems over human labeled workload, returning query results at 90% accuracy.

INDEX TERMS Databases, natural language processing, recurrent neural networks.

I. INTRODUCTION

Structured Query Language (SQL) has been the standard querying interface of traditional relational database systems for the last few decades. A good relational database programmer is expected to master SQL programming language and get familiar with the schemas of the database, before writing quality queries for database applications. Both of the tasks of SQL mastering and schema learning are extremely challenging, even for experts with a strong computer science background. The database research community has been devoting huge efforts to the enhancement of database usability, by easing the hardness of SQL programming and complex schema comprehension [20], [21].

The explosive development of machine learning and artificial intelligence techniques, especially boosted by deep learning [6], has revealed a new possibility of interaction mechanisms between human users and complex machine systems. Particularly, the huge success of performance improvement on machine translation tasks [17], [18] is inspiring the adoption of natural language as the inputs of queries and commands instead of SQL, therefore minimizing the technical demands to the system administrator on both programming skills as well as knowledge to the database schema.

Different from the existing techniques in the database community, which requires human efforts on error correction [11], deep learning approach attempts to construct the transformation from input natural language sequence to the output SQL language sequence purely based on the translation samples only. As a mainstream deep learning based approach, semantic parsing convert natural language into formal and executable logical form [3], [5], [19]. Besides of semantic parsing, researchers are also attempting to generate

The associate editor coordinating the review of this manuscript and approving it for publication was Vlad Diaconita.

executable logical form by directly linking the semantic interpretation of the input natural language and the records in the database [14], [15], [22].

However, the straightforward solution by using deep learning models does not render the effectiveness and efficiency as expected. Firstly, most of the computation in deep learning model training is wasted on learning the grammar structure of SQL, which is actually well studied and modeled by mature techniques in the database community. Secondly, machine learning approaches linking query results with individual cells in the database tables [14], [15], [22] do not scale well with the table size, even not reusable when updates are applied on the tables. Thirdly, machine learning approaches always output the result SQL queries even when the input question does not match the schema/contents of the database.

Inspired by recent work on integrating grammar structure into sequence-to-sequence model [7], [16], [23], we design our new natural-language database querying (NADAQ) system to tackle all these problems, by smoothly fusing deep learning and traditional database query parsing techniques. The core technical contributions include: 1) instead of forcing deep learning models to learn SQL grammar, we use finite state machines to track the grammatical states of the output SQL sequence. These states are also fed into the neural network, in order to help the model to better select the subsequent words; 2) by tracking the selection of tables, columns and predicate expressions, we build models to evaluate the relevance of the original questions to the result SQL queries, and thus rejecting irrelevant questions; 3) we adopt customized beam search technique to identify candidate queries and present the natural language explanations of the queries to the user's query refinement.

In the rest of the paper, section II overviews the system architecture of NADAQ. Section III introduces the models adopted by NADAQ. Section IV presents the demonstration workflow. Section V evaluates the performance of the models experimentally, and Section VI finally concludes the work.

II. SYSTEM OVERVIEW

As is shown in Figure 1, NADAQ consists of three major components, including data storage module, model management module and user interface module. Data storage module includes MySQL as the underlying database engine, which extracts meta-data from the tables for translation model training and executes the SQL queries to return search results to human users. Model management module is the core of NADAQ, which manages various models for bidirectional translation between natural language and SQL, as well as

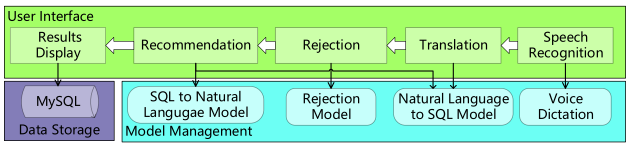


FIGURE 1. NADAQ system overview.

rejection models for irrelevant question screening. User interface module contains the interfaces to human users to support all important functionalities. The key technical components include:

Speech Recognition converts the audio input into natural language text. The module uses the voice dictation interface provided by the iflytek open platform.[1] It also supports manual correction, such that the user is allowed to revise the text output based on his/her own understanding.

Translation adopts a customized machine learning model based on encoder-decoder [2], [17], which is state-of-the-art solution of machine translation. The key technical innovation in this part is the addition of additional hidden states into the model to track the SQL-aware grammatical states based on a finite state machine. These hidden states are helpful to filter out invalid output words at decoding phase and provide useful hints for better neural network training.

Rejection enables NADAQ to reject meaningless inputs from the users. In order to evaluate the relevance between user inputs and contents of the database, NADAQ builds a rejection model on top of the translation model. The rejection decisions are made based on the uncertainty of the translation model when choosing tables, columns and predicate expressions for the SQL queries.

Recommendation enhances the effectiveness of NADAQ by providing candidate queries to the users for refinement and selection. To help users without any SQL background knowledge, NADAQ translates the candidate queries into natural language, so that human users could easily understand the physical meanings of the candidate queries and improve the efficiency of human-computer interaction.

Result Display executes the SQL query on the database and consequently displays the query outcomes to the user.

III. MODELS IN NADAQ

A. QUERY TRANSLATION

We develop a SQL-aware recurrent neural network structure on top of the encoder-decoder framework, for the task of translation from natural language to SQL, as is shown in Figure 2. Basically, the neural network accepts natural language input as a sequence of words in the encoder phase. After finishing the processing of all input words, the neural network transits to decoder phase, starting to output words

[1]http://www.xfyun.cn

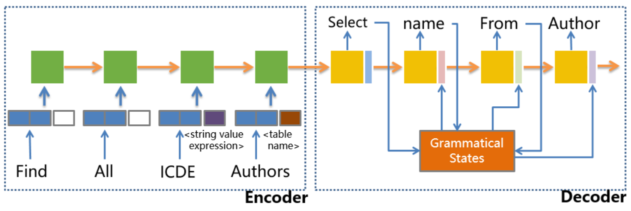


FIGURE 2. A running example of customized encode-decode neural network for our SQL translation task.

as the translation outcomes. The motivations of the neural network customization for SQL translation are introduced below.

1) ENCONDER PROCESSING

The key of the encoder phase is to digest the original natural language input and put the most important information in the memory before proceeding to the decoder phase. We propose to extract additional semantic features that link the original words to the semantics of the grammatical structure of the target language. We generate a group of labels based on the Backus-Naur Form (BNF) of SQL. Specifically, each label corresponds to a terminal symbol in the BNF Grammar for SQL. Given a small group of samples, we manually label the words and employ conditional random fields (CRFs) [10] to build effective classifiers.

2) DECODER PROCESSING

We incorporate SQL parsing techniques into the neural network in two different ways, including the embedding of grammar state in the hidden layer and the masking of word outputs. In SQL parsers, based on the precedented word outputs, the parser selects the candidate expression for following words based on the structure of BNF. Motivated by this, we use a binary vector structure to represent all possible grammatical states, each of which corresponds to a candidate expression in BNF. We also employ a stack structure, which is used to track the grammar states when sub-queries are generated recursively. Let g_t denote the grammar state in top entry in the stack at time t. To incorporate g_t into the model, the memory of the neural network is updated as follows:

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + V_f g_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + V_i g_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + V_o g_{t-1} + b_o) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes \sigma_c(W_f x_t + U_f h_{t-1} + V_f g_{t-1} + b_f) \\
h_t &= o_t \otimes \sigma(c_t)
\end{aligned}$$

where $\otimes$ indicates element-wise multiplication operation.

TABLE 1. Partial rules of ***short-term dependencies.***

ID	State	Current Word/Symbol	Next Word/Symbol
S1	⟨query⟩	SELECT	⟨derived column⟩
S2	⟨select list⟩	⟨derived column⟩	⟨comma⟩,FROM
S3	⟨from clause⟩	FROM	⟨table name⟩

In the decoder, there are two types of word masks used to filter out invalid words for outputting, which are mainly based on *short-term dependencies* and *long-term dependencies* respectively. At each step, the decoder chooses one rule from candidate short-term dependencies, e.g., rules in Table 1, and possibly multiple rules from candidate long-term dependencies, e.g., rules in Table 2. The short-term dependency rule is updated according to the current grammar state as well as the last output word from the decoder. Long-term dependencies are updated based on the active symbols chosen by the SQL parser, maintained in the grammar state vector.

TABLE 2. Partial rules of ***long-term dependencies.***

ID	Symbol	Current Word	Long Term Word Mask
L1	⟨derived column⟩	name	movie_table, movie_column, name
L2	⟨table name⟩	actor	movie_table, movie_column

We use a binary vector s to indicate the masks generated by the single rule of short-term dependency, and l_i for the i-th mask generated by the rule of long-term dependencies. Given these masks, the word selection process in the decoder is modified as:

$$y_t = \sigma_y(W_y h_t + b_y) \otimes s \otimes l_1 \ldots \otimes l_L, \qquad (1)$$

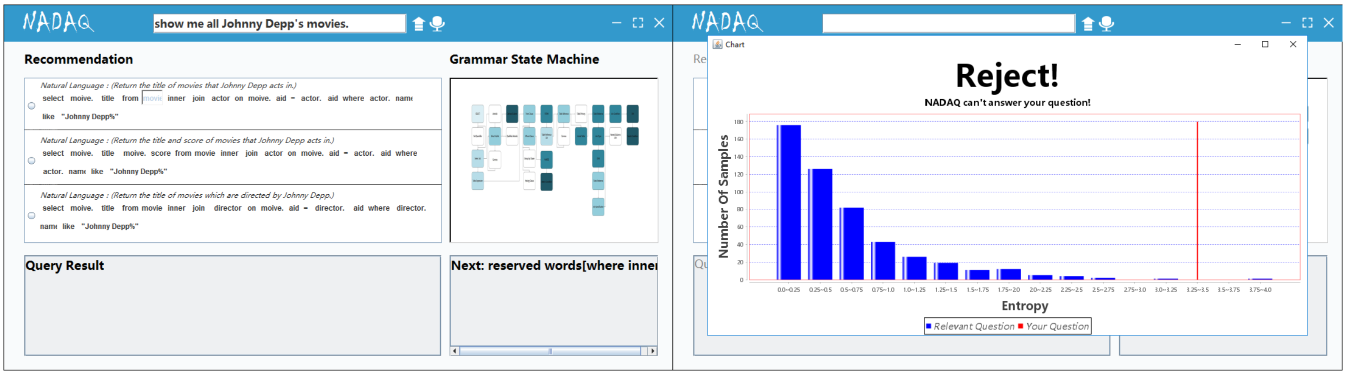
where L is the number of active rules of long-term dependencies.

B. QUERY REJECTION

The query rejection model is used to block meaningless or irrelevant questions that cannot be processed by SQL queries based on the database schema. Our query rejection model is motivated by the observation on the uncertainty of the neural network when outputting keywords in SQL output, especially on the table name, column name and predicate expressions. Therefore, NADAQ collects the statistics of entropies of output word selection based on the grammatical states maintained by the translation model. The threshold method and the MLP method are two basic strategies employed by the system. The first strategy simply rejects a question if the total entropy is above a specified threshold, while the second strategy builds a 3-layer fully connect neural network model by utilizing valid and invalid questions to the database.

C. QUERY RECOMMENDATION

There are always errors in the translations from natural language to SQL, regardless of the training performance of the machine learning model. To further enhance the effectiveness of the system, we design a recommendation mechanism, which returns multiple candidate SQL answers to the user. The key technical challenge is to cover meaningful and representative SQL queries which are more likely to be answers to the original question in natural language. To facilitate effective query recommendation, we design search strategies to generate multiple answers, ranking strategy for candidate selection and presentation strategy for better human-computer interaction.



Natural Language : (Return the title of movies that Johnny Depp acts in.)
select moive. title from movie inner join actor on moive. aid = actor. aid where actor. name like "Johnny Depp%"

Natural Language : (Return the title and score of movies that Johnny Depp acts in.)
select moive. title moive. score from movie inner join actor on moive. aid = actor. aid where actor. name like "Johnny Depp%"

Natural Language : (Return the title of movies which are directed by Johnny Depp.)
select moive. title from movie inner join director on moive. aid = director. aid where director. name like "Johnny Depp%"

(a)

(b)

FIGURE 3. User interface snapshots in NADAQ. (a) Interface of query inputs and recommendation. (b) Interface of query rejection.

1) SEARCH STRATEGY

In sequence-to-sequence learning and its variants, beam search is commonly used in the inference phase, e.g., [9], in order to maintain multiple promising candidates at the same time. We apply guided beam search, by expanding the group of candidates at crucial grammatical states, when the output sequence is attempting to choose tables or columns.

2) RANKING STRATEGY

Given the candidate queries for selection, we rank the queries based on certain score function over the SQL queries. The score is calculated based on the probability of the output words at pivot time steps, which are the time steps involving expansion of the beam. The score is then normalized based on the number of pivot time steps encountered by the individual SQL query, to avoid the unfair preference to long queries.

IV. DEMONSTRATION WORKFLOW

In the demonstration of NADAQ, following the general workflow of the system, users are allowed to ask queries in speech. As is shown in Figure 3(a), the recommendation candidates are presented on the screen. By clicking the words in the SQL query candidate, NADAQ displays the grammatical status on the top right corner and the word selection options in the bottom right corner. Once the user selects one of the recommendations, NADAQ executes the query and present the results in the box on the bottom of the interface. If NADAQ decides to reject the query due to the limited relevance of the original question to the contents of the database, as is shown in Figure 3(b), it displays the histogram of valid questions' entropy, as well as the confidence threshold (the red line in the figure) of the rejection decision. Here, the confidence threshold can be determined by the threshold method and the MLP method.

V. EXPERIMENTS

We run our experiments on three databases. Academic (MAS) database has 17 tables, containing records of 3,543,360 publications and 1,592,014 researchers, collected by Microsoft Academic Search. The *IMDB* database has 3 tables containing records of 3,654 movies, 4,370 actors and 1,659 directors, respectively. We also employ the logical form benchmark *GEO*, by converting the logical queries into equivalent SQL queries as done in [4] and [8]. The *GEO* database has 7 tables, which contains records of 368 cities in 51 states in the USA. The statistics of the datasets are listed in Table 3. Note that SQL vocabulary size of GEO is much larger than that of the other two datasets, and nearly 40% of the SQL statements containing sub-queries on the GEO dataset. On *GEO*, we use original workloads in the dataset. On the other two databases, we ask volunteers to label over randomly generated queries, by describing the queries in English. Two examples of the generated SQL queries with the aggregator and join operation are given as follows:

```
SELECT <column_array>
FROM <table> WHERE <column> =/> <value>
```

```
SELECT AGG(<table_1.column_array>)
FROM <table_1> INNER JOIN <table_2>
ON <table_1.key> = <table_2.key>
WHERE <table_2.column> =/> <value>
```

All these samples are used in the training of the SQL translation model. The translation model is implemented on top of Tensorflow 1.2.0. We employ 1 hidden layer with 256 neurons for both encoder and decoder, and softmax neuron for output word selection. In order to augment Long Short

TABLE 3. Workload statistics on three databases.

	MAS	*IMDB*	*GEO*
Average NL length	13.88	16.48	7.59
NL vocab size	179	233	151
Average SQL length	15.18	14.83	16.06
SQL vocab size	33	39	89
Having subqueries (%)	0	0	39.8
Having joins (%)	74.18	48.55	0.45
Workload generation	Add-on	Add-on	Original [8]

Term Memory [6] with the grammatical states, we manually revise the original LSTM model in Tensorflow to add new hidden states and manipulation logic.

A. BASELINE APPROACHES

We employ three baseline approaches in performance evaluation, including the convolutional neural network machine translation (conv-seq in short) model [4], the attention-based sequence-to-sequence machine translation (attn-seq in short) model [1], and semantic parsing model with feedback (SPF in short) [7]. Note that SPF does not work on the IMDB and MAS dataset. Because SPF uses templates to enlarge training data, and there is no template for join operations on these two datasets.

B. PERFORMANCE METRICS

We measure F1 accuracy of the results based on the recall and precision of the query results. To get the recall and precision, we executing these queries in the relational database and compare the results against that of the ground-truth queries.

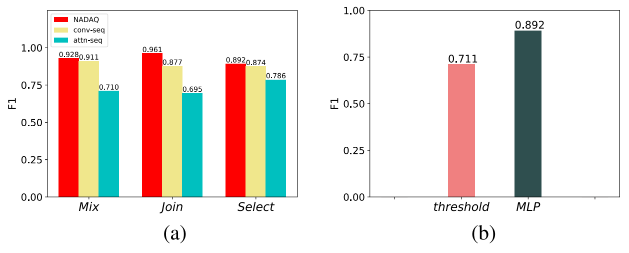


FIGURE 4. **Testing results on *IMDB*. (a) Query result accuracy. (b) Query rejection accuracy.**

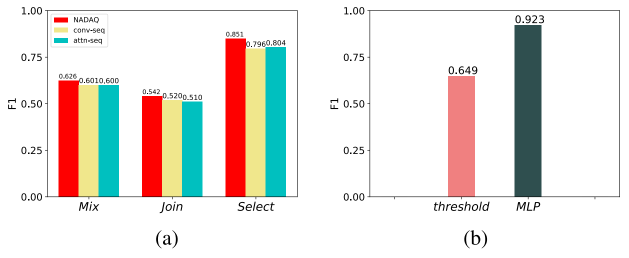


FIGURE 5. **Testing results on *MAS*. (a) Query result accuracy. (b) Query rejection accuracy.**

C. EXPERIMENTAL RESULTS

We report the experimental results of translation and rejection on three databases in Figures 4, 5 and 6 respectively. On *IMDB* database (Figure 4(a)) and *MAS* database (Figure 5(a)), we present the testing results with three types of workloads, *Mixed* workload with all join and select queries, as well as *Join* and *Select* workloads with individual type of queries only. On *GEO* database, we present the results with the original workloads(Figure 6a).

Generally speaking, NADAQ outperforms *conv-seq* and *attn-seq* in all settings by a significant margin, because the addition of grammar states brings benefits to save the

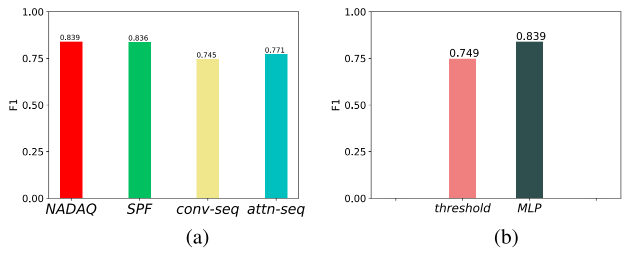


FIGURE 6. **Testing results on *GEO*. (a) Query result accuracy. (b) Query rejection accuracy.**

unnecessary efforts on grammar understandings. Though NADAQ and SPF both performed well on the GEO dataset, the F1 score of NADAQ(0.839) is still higher than that of SPF(0.836) on *GEO* database. Moreover, NADAQ involves fewer manual operations. The F1 scores of NADAQ on *IMDB* database and *GEO* are above 80%, close to the standard of commercial usage.

We also report the F1 score of the query rejection mechanism with two alternative models on all the three datasets. For the threshold method, the candidate is rejected if its entropy is larger than the mean of the valid questions' average entropy and the invalid questions' average entropy. For the MLP method, a multi-layer neural network method is trained to determine the threshold. As shown in the figures, the MLP method outperforms the threshold method over all the three datasets. In detail, the F1 score of the threshold approach is around 0.75 on IMDB and GEO, and 0.65 on MAS, while that of the MLP method is above 0.9 on IMDB and MAS, and 0.84 on GEO.

VI. CONCLUSION

In this paper, we demonstrate our new natural language database querying (NADAQ) system. The design of NADAQ combines the state-of-the-art deep learning techniques and the mature SQL parsing techniques. It enables the system to significantly improve the quality of query translation and inspire new solutions to meaningless question rejection and query candidate recommendation. The results show NADAQ provides nearly commercial standard service as a querying interface to human users, even when they neither understand SQL nor comprehend the database schema.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. (2015). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473

[2] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[3] L. Dong and M. Lapata, "Language to logical form with neural attention," in *Proc. ACL*, 2016, pp. 33–43.

[4] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. ICML*, 2017, pp. 1243–1252.

[5] K. Guu, P. Pasupat, E. Z. Liu, and P. Liang, "From language to programs: Bridging reinforcement learning and maximum marginal likelihood," in *Proc. ACL*, 2017, pp. 1051–1062.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a neural semantic parser from user feedback," in *Proc. ACL*, 2017, pp. 963–973.

[8] R. Jia and P. Liang, "Data recombination for neural semantic parsing," in *Proc. ACL*, 2016, pp. 12–22.

[9] A. Kannan *et al.*, "Smart reply: Automated response suggestion for Email," in *Proc. KDD*, 2016, pp. 955–964.

[10] J. Lafferty *et al.*, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn. (ICML)*, vol. 1, 2001, pp. 282–289.

[11] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Proc. VLDB Endowment*, vol. 8, no. 1, pp. 73–84, 2014.

[12] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision," in *Proc. ACL*, 2017, pp. 23–33.

[13] P. Liang, "Learning executable semantic parsers for natural language understanding," *Commun. ACM*, vol. 59, no. 9, pp. 68–76, 2016.

[14] L. Mou, Z. Lu, H. Li, and Z. Jin, "Coupling distributed and symbolic execution for natural language queries," in *Proc. ICML*, 2017, pp. 2518–2526.

[15] A. Neelakantan, Q. V. Le, M. Abadi, A. McCallum, and D. Amodei. (2016). "Learning a natural language interface with neural programmer." [Online]. Available: https://arxiv.org/abs/1611.08945

[16] M. Rabinovich, M. Stern, and D. Klein, "Abstract syntax networks for code generation and semantic parsing," in *Proc. ACL*, 2017, pp. 1139–1149.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[18] Y. Wu *et al.* (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation." [Online]. Available: https://arxiv.org/abs/1609.08144

[19] C. Xiao, M. Dymetman, and C. Gardent, "Sequence-based structured prediction for semantic parsing," in *Proc. ACL*, 2016, pp. 1341–1350.

[20] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summarizing relational databases," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 634–645, 2009.

[21] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summary graphs for relational database schemas," *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 899–910, 2011.

[22] P. Yin, Z. Lu, H. Li, and B. Kao, "Neural enquirer: Learning to query tables with natural language," in *Proc. IJCAI*, 2016, pp. 2308–2314.

[23] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," in *Proc. ACL*, 2017, pp. 440–450.

BOYAN XU received the B.S. degree in computer science and technology from the Guangdong University of Technology, in 2014, and the master's and Ph.D. programs in computer application engineering for Ph.D. degree. His research interests include a variety of different topics including machine learning, natural language processing, and their applications.

RUICHU CAI received the B.S. degree in applied mathematics and the Ph.D. degree in computer science from the South China University of Technology, in 2005 and 2010, respectively. He was a Visiting Student with the National University of Singapore, from 2007 to 2009, and a Research Fellow with the Advanced Digital Sciences Center, Illinois at Singapore Pte, from 2013 to 2014. He is currently a Professor with the School of Computer, Guangdong University of Technology. His research interests include a variety of different topics including causality, machine learning, and their applications.

ZHENJIE ZHANG received the B.S. degree from the Department of Computer Science and Engineering, Fudan University, in 2004, and the Ph.D. degree in computer science from the School of Computing, National University of Singapore, in 2010. He was a Senior Research Scientist with the Advanced Digital Sciences Center, Illinois at Singapore Pte. He is currently the R&D Director with Singapore R&D, Yitu Technology Pte Ltd. His research interests include a variety of different topics, including causality, database query processing, high-dimensional indexing, and data privacy.

XIAOYAN YANG received the B.S. degree from the Department of Computer Science and Engineering, Fudan University, and the Ph.D. degree in computer science from the School of Computing, National University of Singapore. She was a Postdoctoral Fellow with the Advanced Digital Sciences Center, Illinois at Singapore Pte. She is currently a Senior Data Scientist with Singapore R&D, Yitu Technology Pte Ltd.

ZHIFENG HAO received the B.Sc. degree in mathematics from Sun Yat-sen University, in 1990, and the Ph.D. degree in mathematics from Nanjing University, in 1995. He is currently a Professor with the School of Computer, Guangdong University of Technology, and also with the School of Mathematics and Big Date, Foshan University. His research interests include various aspects of algebra, machine learning, data mining, and evolutionary algorithms.

ZIJIAN LI received the B.S. degree in software engineering from the Guangdong University of Technology, in 2017, where he is currently pursuing the M.S. degree in computer science and technology. He has been a Visiting Student with the Advanced Digital Sciences Center, Singapore, since 2017, and has also been with Nanyang Technological University, since 2018. His research interests include a variety of different topic including natural language processing, transfer learning, and domain adaptation.

ZHIHAO LIANG received the B.E. degree in computer science and technology from the Guangdong University of Technology, in 2018, where he is currently pursuing the master's degree. His research interest includes natural language processing.

• • •