Comparative Study of Neural Semantic Parsers

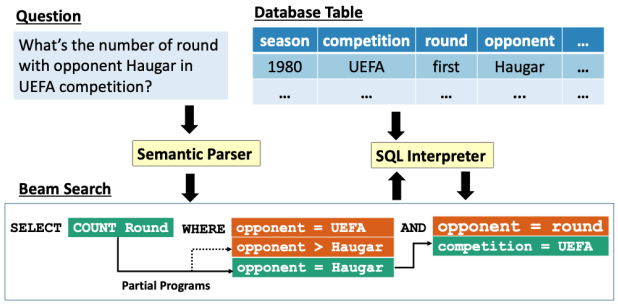
Ramraj Chandradevan
Emory University

Abstract

Neural semantic parsing is text to structured-text generation task, and to be more specific, it is also called as text to structured query language (SQL). Text-to-SQL is a recently advancing area of research for the database community, where researchers leverage the advances in natural language processing (NLP) and deep learning (DL) to enhance the performance, robustness, and scalability. We initiated the step to draft a very first survey on neural text-to-SQL literature to enlighten the research community. In addition, we conducted experiments to show the time profiling against different hardware settings to show the potential use of distributed implementation and multi-GPU. Also we experimented the two recent text-to-SQL frameworks, such as Seq2SQL[25] and TypeSQL[20], to show that performance trained and evaluated across different datasets to show why model complexity is important in a neural based models.

1 Introduction

Semantic parsing is an area of research where the goal is to semantically understand the natural language to predict a grammar-based target language. Even though there has been few traditional frameworks that model semantic parsing, with the emergence of deep learning, neural semantic parsing became an interesting topic. Figure 1 shows an example of a text-to-SQL generation task, where an input question and a table is given. The algorithm needs to predict the correct sequence of SQL grammar terms, table variable names, and constant values.



The problem of text-to-SQL can be modelled in multiple-ways, for example as sequence-to-sequence generative task, slot-filling task, and sketch-based task. We are defining the problem as slot-filling task, where the output SQL query has a fixed template with commands like *SELECT*, *WHERE*, *JOIN*, and *GROUPBY*. However, there can be number of conditional clauses placed together with conditional operators, such as *AND*, *OR*, multiple table join operations using *JOIN* and *GROUPBY* operators, and nested sub-queries in the place of conditional clause, column names, or constant values. A simple SQL query would typically have *SELECT*, *WHERE*, and *aggregation operators* (such as *SUM* and *COUNT*) with variable number of conditional clauses. The task for a generative model is to predict the slot tokens. Those tokens can be aggregation operators, table column names, constant values, and conditional operators.

As mentioned in the problem definition, we can imply the complexity of the task in hand. Moreover, the challenges can rise from the database and question components. Question can bring challenges, such as names not found in database, variation of constant value data type (integer, float, string, data, and time), and typos. Database schema itself has challenges, for example table header names delimited with special characters, large number of records where finding the constant value in the column bring a retrieval and indexing challenge, invisible and non-accessible database header and content

to the user or the querying system. There has been many frameworks designed to handle each of these aforementioned challenges uniquely or together, and we will discuss them in the next section. Moreover, text-to-SQL can be used in many real-world use cases, for example conversational systems and dialogue systems often need a time sensitive high qualitative database retrieval system queried using natural language question. Therefore, in this work, we have carefully investigated and provided a survey on text-to-SQL to support the future research on neural semantic parsing. Moreover, we implemented and analyzed a distributed training and inference of the state-of-the-models (Seq-to-SQL[25] and TypeSQL[20]) in CPU and GPU devices to show the comparison of time profiling. This comparison can help to decide which system can be deployed in real world dialogue systems. Moreover, we have used Seq-to-SQL[25] as a baseline model and evaluated the performance across different recently published datasets. This comparison will demonstrate the generalizability and applicability of a model trained on a specific dataset. We have also summarized our three-fold contributions as follows.

1. We have provided a detailed literature survey on text-to-SQL generation with proper category of frameworks and datasets.
2. We have compared the time profiling of two recent generative models against distributed CPU and GPU training.
3. We have compared the performance of Seq-to-SQL model on wide range of recent datasets to show the model generalizability.

2 Background and Related Work

Text-to-SQL generative task has been famous for the past five years since the release of large scale publicly available dataset, called WikiSQL [25]. It was a work from Salesforce, and the authors have developed a framework based on reinforcement learning, called Seq2SQL. The identify potential challenges and addresses few solutions in their model architecture to improve the performance. After that there has been many public datasets released, for example ATIS [5], GeoQuery, SParC [23], CoSQL [22], WikiTableQuestions, and Spider [12]. With the release of diverse, large-scale, cross-domain dataset release, there has been many research questions rise in the SQL and database community to tackle. Researchers has use many diverse topic of technologies in the domain of deep learning to improve the performance, robustness, simplicity, and generalizability of the proposed architectures, for example zero-shot learning, interaction and human-in-the-loop based interaction feedback enhanced learning, reinforcement learning, and BERT based pre-trained language model based learning.

For the problem of text-to-SQL generation, the input is a question and the database meta-data (often database content as well, but typically restricted because of the privacy concern). The output is either a serial sequence of tokens formulating a SQL query or the corresponding placeholder predictions for $SELECT$, $GROUPBY$, $column_name$, $aggregation_operator$, and $table_name$. We have categorized

To describe the survey on text-to-SQL generative models, we have categorized the existing research works based on the research objectives, such as cross-domain applicability, model generalizability, and robust to complex nature of database schema. We also have categorized the research works based on the which component that the model focus on either question, database schema/ content, or both combined. Moreover, all the existing frameworks can be divided into the following model-based approaches: reinforcement learning based models, supervised (sequence-to-sequence) models, zero-shot learning based models, pre-trained language-model (LM) based models, and human interaction (feedback) based models. We are going to have concise description of the related works in the below content.

2.1 Research Objective Based

2.1.1 Cross-domain Applicability (Generalizability)

Typically, the models trained using deep learning techniques on a certain dataset might fail to attain good performance in a different dataset that posses different data distribution or different characteristic and nature of data. The possible reasons might be the model was over-fitted for the pre-trained dataset, test data has unseen samples from a different complex dataset, and model lacks generalizing ability. To tackle this problem, different techniques have been used in DL domain, such as data augmentation, cross-domain training, zero-shot or few-shot learning, and regularizing the model training. In our text-to-SQL research community, the datasets Spider [12] and SParC [23], and CoSQL [22] are used to evaluate any models that were developed to handle cross-domain adoptation. Most of the recent works always attempt to accommodate cross-domain applicability in some form in their proposals.

SynataxSQLNet [21] is the first work to identify the challenge, and it uses syntax tree-based decoder with column-aware attention encoders and SQL generation path history. Additionally, the authors used cross-domain data augmentation to enhance the results further.

IRNet [7] is another work that adopts cross-domain challenge. However, instead of building an end-to-end

framework, the authors decoupled the framework into three phases: schema linking over question and database schema, synthesizing SemQL query based on grammar-based neural model, and finally predicting SQL query using IRNet using domain knowledge.

Another work based on **edit mechanism** [24] leverages context information from the question to improve the performance. Based on the fact that adjacent questions are linguistically dependent and also shares SQL overlap, the model leverages user interaction history to improve the cross-domain applicability.

RYANSQL [4] uses two input manipulation methods to improve the cross-domain generalizing ability.

2.1.2 Complex Database Incorporation

Any database in enterprise database servers are usually in large-scale, diverse, and links content using foreign-keys. Typically, the number of records is enormously larger than the table column in most of the database schemes. SQL queries can range from simple selection command of column name or with an aggregation function to complex queries with table join operations and multiple conditions along with nested queries at any of the query components. At the early stages of text-to-SQL generation, models that evaluated on WikiSQL[25] only infer simple SQL queries without any *join* and *groupby* operations, or nested queries. With the release of Spider[12], above operations with linked database schema with foreign-keys have become inevitable. Thus, the models try to solve complexity in SQL query generation has to address the inherent issues in Spider dataset. Usually, the datasets used for cross-domain adaptability also contain complex database schema, for example Spider[12] and SParC[23]. Moreover, the models address the cross-domain incorporation also propose solution for database complexity because both of the challenges are complementary in large-scale practical databases. SyntaxSQLNet[21], IRNet[7], and RYANSQL[4] are few of the works that solve database complexity and cross-domain adaptability at the same time.

2.2 Input Attention Based

The only inputs to the text-to-SQL models are the natural language question and table header names. In research settings, often the table content (the index of database table contents) is also used to improve the performance by ignoring the privacy concerns. However, it is not possible in real system deployments. The natural language question is a free-form text, where typos are inevitable. Most of the time, since the database content is not visible (accessible) to the user or the system that sends out query, the requested table column name or the conditioning constant values can present in varying format (they might not exactly match the substring/string in table headers or content often with numbers and strings). This challenge motivates the researchers to invest more time on encoding the natural language question effectively and address the inherent challenges. At the same time, the database columns usually written in natural language variable names with special characters delimited. This nature raise the challenge of out-of-vocabulary encoding of the tokens (with special character delimiters) from table headers. The columns in a table has heterogeneous values across, such as string, integers, floating point values, dates, times, and etc. The requested natural language text might not bring the constant values in the favorable data type to be matched with column value. These aforementioned challenges motivate the practitioners to either address the concerns at the prepossessing stage or inference stage (on-the-fly). **TypeSQL**[20] utilizes the data type information to better understand and encode rare entities and numbers. The content-sensitive nature of the model improves the performance significantly. Another work[14] that designs an **execution-guided decoder** to leverage SQL semantics for robust performance. **SyntaxSQLNet**[21] introduces column-aware attention mechanism to better understand the table schema. Another work[2] better represents the parsing language, design a better encoder-decoder architecture, and uses Graph Neural Network (GNN) for better database schema encoding. A relation-aware self-attention based model[11] was proposed to better encoder database schema. This design better suits for unseen database schema in test time. **Rule-SQL**[8] uses rule-based exploration methods on table columns and question string and then apply weak-supervised learning to train a model. Several rules were used to narrow down the SQL exploration space. **TREQS**[15] model was used in the domain of electronic medical record application to handle wide use of abbreviation, terminologies, and typos. A global reasoning model[1] was proposed to reason about the query structure to make more contextually informed selection of database constants. This framework uses GNN and message-passing to select database constants conditioned on question. Also for better alignment, it trains a neural ranking model. An edit based method[24] uses utterance-table encoder and table-aware decoder to incorporate context of user utterance and table schema. A Byte-Pair Encoding (BPE)[**?**] method builds Abstract Syntax Tree (AST) of SQL to guide BPE merges to leverage BPE in better question encoding. This work also introduces a novel stopping criterion to avoid over-fitting. **RAT-SQL**[13] proposes a unified framework with relation-aware self-attention mechanism to address better schema encod-

ing, schema linking, and feature representation. **RYANSQL**[4] uses Statement Position Code (SPC) to transform nested SQL queries into non-nested SELECT commands before feeding into the generative model to improve the performance. **TABERT**[19] proposed to encode semi-structured tables with BERT pre-trained language model for better representation learning.

2.3 Model Based

To solve the problem of text-to-SQL, there has been wide range of models applied in the literature, ranging from different learning paradigms to user-interaction based frameworks. These works can be categorized and described in the following content. Almost all of the methods use deep learning techniques and learning approach in their core algorithm.

2.3.1 Reinforcement Learning

In the context of deep learning, most of the time, supervised-learning is used to train any neural models. However, reinforcement learning (RL) is applied on unique applications where it outperforms supervised learning. RL requires to have a certain properties in the problem statement. There has to be an agent, which performs actions in an action-space, functions in a environment, called state-space. Based on the success of failure of the agent's action, the environment gives an incentive as reward or penalty. By modelling and learning an agent to complete the task successfully, we mimic a human's performance to an agent.

Seq2SQL[25] is the very first attempt to model text-to-SQL as RL mechanism. The main reason to use RL in Seq2SQL is that the order of the conditional statements in a SQL query do not matter, and the successful execution of an SQL query does matter. Only after predicting every components in an SQL query, the success or failure of prediction can be inferred. However, later years, the researchers shifted their research focus on addressing different challenges, and they found comfortable using supervised-learning. The improve from RL is limited in those frameworks. Later, a interactive semantic parsing framework[17] was proposed to enhance the semantic parsing accuracy and user confidence. The goal was to design a model-based agent that cleverly mimics a human performance in semantic parsing. In addition to question and table headers, subsequent user feed-backs also provided as input. Most recently, a human-in-the-loop based model[18] was proposed to directly learning semantic parser from users. An agent imitates the user behaviour and continues improving the performance automatically. A novel *annotation-efficient imitation learning algorithm* was used to combat the sparsity of annotations.

2.3.2 Sequence-to-Sequence Learning

Any language generative models has been considered as sequence-to-sequence learning frameworks. SQL is also a language that the database systems can only understand. Given an input sequence of question tokens, a generative model is expected to predict most likely sequence of SQL commands, key-words, variable names, and constants by following the SQL constraints and grammar. Most of the text-to-SQL modelling has been framed as slot-filling sketch based models. SQLNet[16] is one of the first work that use supervised sequence-to-sequence learning to improve the performance compared to RL based Seq2SQL[25]. A column attention was added to improve the accuracy. TypeSQL[20] is another work that used content-sensitive information to improve the seq-to-seq performance. *Wang et al.*[14] proposed a execution-guided decoder to improve the seq-to-seq performance. There has been more recent works that has the core learning method as seq-to-seq learning[21, 2, 7, 11, 15, 1, 24, 4].

2.3.3 Zero-shot Learning

Zero-shot learning is a recently introduced topic in the context of deep learning. It has been applied on wide range of research topics, such as computer vision, natural language processing, data mining, and etc. In the scenario, where the test data contains the unseen examples during the train phase, the embedding and encoding nature of the test input is difficult. However, zero-shot learning suggests a pre-trained model on one domain, can directly be applied to a new domain (with unseen data) by training the earlier model with a slightly different objective function. The nature of zero-shot learning is most welcomed in text-to-SQL because we often see unseen table and header information in test data.

Chang et al.[3] raise the question that does the impressive performance means the model is well generalized and performs well in test data. The authors first diagnosed the bottlenecks in text-to-SQL model, and designed a simple, effective auxiliary task to guide the training objective. This auxiliary task acts as a supportive model and a regularization term to better generalize the model for test data. The authors used a zero-shot subset from WikiSQL to demonstrate the improvement.

2.3.4 Pretrained Language Model (BERT)

Pretrained language model (LM) was recently introduced in the domain of language based research. Since then it has been widely used in all areas, such as NLP, IR, conversational system, and etc. BERT[6] is a commonly

used pre-trained LM in a versatile of downstream tasks, such as machine translation, question answering, document ranking, and etc. As expected, BERT-based models has superior results compared to neural models based on deep learning architectures. Therefore, researchers always introduce and demonstrate their novel techniques in separated set of benchmarks, called LM-based models and neural-based models.

BERT was initially used in Rule-SQL[8] work to better encoder the representation space. Thereafter, a content-enhanced BERT-based model[9] use simple methods to leverage table contents as additional two feature vectors to support BERT-based generative model. In RAT-SQL[13], the authors showed that the BERT-based augmented model achieves better accuracy. First time, Bertrand-DR[10] proposes a discriminative re-ranker to improve the performance. It first generate list of SQL prediction candidates using beam-search. Then a schema-agnostic BERT re-ranker was fine-tuned for the classification task to select best candidate. Finally, TABERT[19] proposes to jointly learn natural language sentence representation and semi-structured tables.

2.3.5 Human-in-the-loop

Although the neural models performs automatically in generative, classification, regression, or ranking tasks, a little human-intervention can support the performance increase drastically. Following this hypothesis, many research works has shown the superior performance of human-interactive feedback based (human-in-the-loop: HITL) models. *Yao et al*[17] proposed an interactive semantic parsing to improve the SQL generation task. An another imitation learning based model[18] also used user interaction to train an agent to mimic the performance. Moreover, a new dataset, called CoSQL[22] was proposed for the use-case of text-to-SQL in the dialogue systems. It is a cross-domain dataset, collected from general-purpose database querying dialogue system.

3 Methods

Most parts of the work are a literature survey and comparison study of different experiments. Therefore, besides the implementation of existing two state-of-the-art text-to-SQL models and their distributed implementation, we do not have unique methods to talk about.

4 Experiments

Based on the survey that we conducted, we have decided to move forward with two recent neural text-to-SQL models for our comparative study: Seq2SQL[25] and TypeSQL[20]. In terms of datasets, we have used WikiSQL[25], SParC[23], ATIS[5], and Spider[12]. Our experiments are three-fold, and here are the summary of our experiments.

1. time profiling between single versus multiple GPUs
2. cross domain adaptability of public datasets
3. analysis of rare terms and unmatched terms against database columns

4.1 Time Profiling

Database querying is a time critical task in the applications like dialogue systems (bots). Any dialogue systems has a component as database information extraction, where it also has other components, for example natural language understanding, knowledge reasoning, user intent classification, answer recommendation, and natural language generation. A real world user expects a bot to respond in fraction of seconds. In recent years, all of the above mentioned sub-tasks use neural networks, including database querying. Therefore, by making individual component faster, response quality accuracy and user reliability is achieved.

There can be many number of requests made from each bots to the database server corresponding to the bot. Therefore, we have attempted to make GPU computation parallel so that the text-to-SQL prediction can be speed up. Table 1 shows the time taken by Seq2SQL and TypeSQL for one-batch (64) of execution time taken at inference stage of parallel GPUs, compared against single CPU and single GPU.

Figure 2 shows the time against input natural language question length (question scale). Correspondingly, Figure 3, 4 shows the time taken against number of database records (database scale) and number of clauses (SQL scale) generated in the predicted SQL query. All these three figure results were generated for both Seq2SQL and TypeSQL a multi-GPU setting.

4.2 Dataset Adaptability

We took another step to analyze the dataset adaptability across different test settings. Even though there are many public datasets are available, WikiSQL is the very first dataset. Spider is one of the very largest, complex, cross-domain dataset. Therefore, we compared the performance of Seq2SQL and TypeSQL across four datasets: WikiSQL[25], SParC[23], ATIS[5], and Spider[12]. We followed cross-validation (leave-one-fold-out) across datasets for training and evaluation, and the results are shown in Table 2. Each section belongs to the performance of either Seq2SQL or TypeSQL model. First two sections belongs to the training and testing on

Model	CPU (sec)	GPU (sec)	multi-GPU (sec)
Seq2SQL	0.00	0.00	0.00
TypeSQL	0.00	0.00	0.00

Table 1: Inference time profiling compared on different hardware settings

Model	Train Split	Test Split	Accuracy(qm)	Accuracy(ex)
Seq2SQL	WikiSQL	WikiSQL	0.00	0.00
Seq2SQL	SParC	SParC	0.00	0.00
Seq2SQL	ATIS	ATIS	0.00	0.00
Seq2SQL	Spider	Spider	0.00	0.00
TypeSQL	WikiSQL	WikiSQL	0.00	0.00
TypeSQL	SParC	SParC	0.00	0.00
TypeSQL	ATIS	ATIS	0.00	0.00
TypeSQL	Spider	Spider	0.00	0.00
Seq2SQL	Spider	WikiSQL	0.00	0.00
Seq2SQL	Spider	SParC	0.00	0.00
Seq2SQL	Spider	ATIS	0.00	0.00
Seq2SQL	Spider	Spider	0.00	0.00
TypeSQL	Spider	WikiSQL	0.00	0.00
TypeSQL	Spider	SParC	0.00	0.00
TypeSQL	Spider	ATIS	0.00	0.00
TypeSQL	Spider	Spider	0.00	0.00

Table 2: Performance accuracy comparison across dataste adaptability

the corresponding data splits. However, last two sections belongs to the splits of training in Spider - the large scale cross-domain dataset, and evaluation carried on four different datasets. We have reported query-match accuracy and execution accuracy. Query-match accuracy means the predicted SQL query matches with the ground-truth query by exact string matches. Execution accuracy means the predicted SQL query matches with ground-truth query by SQL execution operation.

5 Discussion

I am still working on generating results so that I can write the discussion confidently.

6 Conclusion

Neural semantic parsing is a recently emerging field of research inter-disciplined between deep learning and database (systems) community. We have provided a first detailed survey on the neural semantic parsing researches conducted until 2020. This survey will be a first step among the beginners in the research on text-to-SQL generation. Moreover, we have compared the time profiling of existing text-to-SQL model to show how multi-GPU environment can speed up the inference in a real-world deployment setting. In addition, we have also conducted a cross-validation based performance evaluation of models trained and tested across different datasets to show how complex nature of a dataset plays important role in the model adaptability and generalizability.

References

[1] BOGIN, B., GARDNER, M., AND BERANT, J. Global reasoning over database structures for text-to-sql parsing, 2019.

[2] BOGIN, B., GARDNER, M., AND BERANT, J. Representing schema structure with graph neural networks for text-to-sql parsing, 2019.

[3] CHANG, S., LIU, P., TANG, Y., HUANG, J., HE, X., AND ZHOU, B. Zero-shot text-to-sql learning with auxiliary task, 2019.

[4] CHOI, D., SHIN, M. C., KIM, E., AND SHIN, D. R. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases, 2020.

[5] DEBORAH A. DAHL, MADELEINE BATES, M. B. W. F. K. H.-S. D. P. C. P. A. R., AND SHRIBER, E. Expanding the scope of the ATIS task: The ATIS-3 corpus. *Proceedings of the workshop on Human Language Technology* (1994), 43–48.

[6] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Figure 1: Time profiling plot against natural language question scale.

Figure 2: Time profiling plot against database-scale.

Figure 3: Time profiling plot against SQL-scale.

[7] GUO, J., ZHAN, Z., GAO, Y., XIAO, Y., LOU, J.-G., LIU, T., AND ZHANG, D. Towards complex text-to-sql in cross-domain database with intermediate representation, 2019.

[8] GUO, T., AND GAO, H. Using database rule for weak supervised text-to-sql generation, 2019.

[9] GUO, T., AND GAO, H. Content enhanced bert-based text-to-sql generation, 2020.

[10] KELKAR, A., RELAN, R., BHARDWAJ, V., VAICHAL, S., KHATRI, C., AND RELAN, P. Bertrand-dr: Improving text-to-sql using a discriminative re-ranker, 2020.

[11] SHIN, R. Encoding database schemas with relation-aware self-attention for text-to-sql parsers, 2019.

[12] TAO YU, RUI ZHANG, K. Y. M. Y. D. W. Z. L. J. M. I. L. Q. Y. S. R. Z. Z., AND RADEV, D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 3911–3921.

[13] WANG, B., SHIN, R., LIU, X., POLOZOV, O., AND RICHARDSON, M. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers, 2020.

[14] WANG, C., TATWAWADI, K., BROCKSCHMIDT, M., HUANG, P.-S., MAO, Y., POLOZOV, O., AND SINGH, R. Robust text-to-sql generation with execution-guided decoding, 2018.

[15] WANG, P., SHI, T., AND REDDY, C. K. Text-to-sql generation for question answering on electronic medical records, 2020.

[16] XU, X., LIU, C., AND SONG, D. Sqlnet: Generating structured queries from natural language without reinforcement learning, 2017.

[17] YAO, Z., SU, Y., SUN, H., AND YIH, W.-T. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. 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)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 5447–5458.

[18] YAO, Z., TANG, Y., TAU YIH, W., SUN, H., AND SU, Y. An imitation game for learning semantic parsers from user interaction, 2020.

[19] YIN, P., NEUBIG, G., TAU YIH, W., AND RIEDEL, S. Tabert: Pretraining for joint understanding of textual and tabular data, 2020.

[20] YU, T., LI, Z., ZHANG, Z., ZHANG, R., AND RADEV, D. Typesql: Knowledge-based type-aware neural text-to-sql generation, 2018.

[21] YU, T., YASUNAGA, M., YANG, K., ZHANG, R., WANG, D., LI, Z., AND RADEV, D. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task, 2018.

[22] YU, T., ZHANG, R., ER, H. Y., LI, S., XUE, E., PANG, B., LIN, X. V., TAN, Y. C., SHI, T., LI, Z., JIANG, Y., YASUNAGA, M., SHIM, S., CHEN, T., FABBRI, A., LI, Z., CHEN, L., ZHANG, Y., DIXIT, S., ZHANG, V., XIONG, C., SOCHER, R., LASECKI, W. S., AND RADEV, D. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases, 2019.

[23] YU, T., ZHANG, R., YASUNAGA, M., TAN, Y. C., LIN, X. V., LI, S., ER, H., LI, I., PANG, B., CHEN, T., JI, E., DIXIT, S., PROCTOR, D., SHIM, S., KRAFT, J., ZHANG, V., XIONG, C., SOCHER, R., AND RADEV, D. Sparc: Cross-domain semantic parsing in context, 2019.

[24] ZHANG, R., YU, T., ER, H. Y., SHIM, S., XUE, E., LIN, X. V., SHI, T., XIONG, C., SOCHER, R., AND RADEV, D. Editing-based sql query generation for cross-domain context-dependent questions, 2019.

[25] ZHONG, V., XIONG, C., AND SOCHER, R. Seq2sql: Generating structured queries from natural language using reinforcement learning, 2017.