A Review of Cross-Domain Text-to-SQL Models

Yujian Gan **Matthew Purver** **John R. Woodward**
School of Electronic Engineering and Computer Science
Queen Mary University of London
Mile End Road, London E1 4NS, UK
{y.gan,m.purver,j.woodward}@qmul.ac.uk

Abstract

WikiSQL and Spider, the large-scale cross-domain text-to-SQL datasets, have attracted much attention from the research community. The leaderboards of WikiSQL and Spider show that many researchers propose their models trying to solve the text-to-SQL problem. This paper first divides the top models in these two leaderboards into two paradigms. We then present details not mentioned in their original paper by evaluating the key components, including schema linking, pretrained word embeddings, and reasoning assistance modules. Based on the analysis of these models, we want to promote understanding of the text-to-SQL field and find out some interesting future works, for example, it is worth studying the text-to-SQL problem in an environment where it is more challenging to build schema linking and also worth studying combing the advantage of each model toward text-to-SQL.

1 Introduction

Text-to-SQL is a task to translate the natural language query (input) written by users into the SQL query (output) automatically. For example, in Table 3, we want to input the question in the table into the model to get the SQL output. Early work on text-to-SQL focused on small-scale domain-specific databases such as Restaurants, GeoQuery, ATIS, IMDB, and Yelp (Yaghmazadeh et al., 2017; Li and Jagadish, 2014; Iyer et al., 2017; Zelle and Mooney, 1996; Tang and Mooney, 2000; Popescu et al., 2003; Giordani and Moschitti, 2012). More recently, Zhong et al. (2017) proposed the first large-scale cross-domain text-to-SQL dataset, WikiSQL, which attracted much attention from the research community (Xu et al., 2017; Yu et al.; Dong and Lapata, 2018). Now, some models (He et al., 2019; Lyu et al., 2020; Anonymous, 2020) for WikiSQL have achieved over 90% execution accuracy, leading to the impression that the text-to-SQL problem has been solved. However, WikiSQL's complexity is limited: its SQL queries only cover a single *SELECT* column and aggregation, together with relatively simple selection predicates in the *WHERE* clauses, thus lacking in terms of complex SQL queries. To facilitate the study of complex SQL generation, Yu et al. (2018b) introduced Spider, a large-scale cross-domain text-to-SQL benchmark with complex SQL queries. Experiments on Spider show previous models designed for WikiSQL suffer a significant performance drop.

In this paper, we discuss the top models for the WikiSQL and Spider benchmarks. Since relatively high generation accuracy has already been achieved for the WikiSQL benchmark, and the SQL structures in Spider cover all SQL structures in WikiSQL, we focus more on models designed for Spider. This paper starts from the comparison of the overall paradigms of the models and then discusses the key modules used by most models. Overall, our contributions are as follows:

- We divide existing text-to-SQL models into two paradigms:
  1) Generate SQL structure ⇒ Fill schema
  2) Label the question ⇒ Generate SQL.
- We study that pretrained embeddings improve performance by improving schema linking and SQL structure generation.
- We evaluate the applicability and advantages of the reasoning assistance modules of previous work.
- We suggest three directions for the future.
  1) How to generate SQL if it is more challenging to build the schema linking.
  2) How to combine the different paradigms (in section 3) toward text-to-SQL.
  3) How to use graph neural networks to improve SQL structure generation.

Question:	What airports don't have departing or arriving flights?
SQL:	**SELECT** AirportName **FROM** Airports **WHERE** AirportCode **NOT IN** (**SELECT** SourceAirport **FROM** Flights **UNION** **SELECT** DestAirport **FROM** Flights)

Table 1: A complex nested SQL with set operator

2 WikiSQL and Spider

Differences: The most significant difference between WikiSQL and Spider is that SQL queries in Spider are more complex than in WikiSQL. Table 1 presents a complex SQL example from Spider, in which the question seems conceptually simple but involves several different pieces of database structure and SQL clause. Besides this, the Spider database contains several tables while there is only one table in the WikiSQL database. The presence of multiple tables introduces column and table name disambiguation problems to Spider, where none exist in WikiSQL. For example, suppose that the table 'student', 'course', and 'studentship' all contain a 'student id' column in a database. You would need to choose one 'student id' column from these tables when the question is 'Show the student id who choose math'. Multiple tables in Spider also cause the number of columns to be dozens of times more massive than WikiSQL, which increases the difficulty of choosing the correct column.

Similarities: Although WikiSQL and Spider are cross-domain settings, most SQL queries do not need domain knowledge during generation. The domain knowledge is a consensus that only exists in a specific field and will not be clearly stated in the question. For example, in a scenario where domain knowledge is needed, ask for 'good restaurant' can correspond to a *WHERE* condition 'star $>$ 3.5' since this domain rates the restaurants ranging from 0 to 5 stars.

Some different domain examples only replace the words related to schema item names, keeping the same sentence structure. Besides, most sentences directly use words related to schema item names instead of synonyms, which allows the model to locate the schema items by word matching. For example, in Table 2, the question rarely uses 'income' or other synonyms to replace 'salary' since the schema table word is 'salary'.

3 The Paradigms of Text-to-SQL

We only discuss two paradigms achieving relatively high performance in the text-to-SQL task, shown in Figure 1, ignoring the paradigms such as directly using a seq2seq to generate SQL. Most models follow the Paradigm One but are based on different neural network architecture, while IE-SQL (Anonymous, 2020) brings a new paradigm achieving the SOTA result in WikiSQL benchmark.

3.1 Paradigm One (Generate SQL structure $\Rightarrow$ Fill schema)

The most common text-to-SQL paradigm is to generate the SQL structure first and then fill the schema items (schema columns and tables). In WikiSQL, because the dataset only contains simple SQL, most models decompose the SQL synthesis into several independent classification sub-tasks. Each sub-task employs an independent classifier taking the entire sentence as input. For example, one classifier would be used to determine which column is the column in *SELECT* clause, and another separate classifier to determine which aggregation function is correct. These models include: SQLNet (Xu et al., 2017), TypeSQL (Yu et al.), SQLova (Hwang et al., 2019), HydraNet (Lyu et al., 2020), X-SQL (He et al., 2019), Coarse2Fine (Dong and Lapata, 2018) and others.

Among them, the SQLNet and TypeSQL models designed for WikiSQL have been transferred to Spider; however, their performance drops significantly. SyntaxSQLNet (Yu et al., 2018a) is the first model designed for Spider and, based on a similar idea, uses independent modules to predict different clauses. However its performance is less effective than some later models based on one unified grammar-based decoder modules (Guo et al., 2019; Bogin et al., 2019a).

Although these later models are based on one unified module, they also treat SQL structure generation and filling the schema items as separate processes. SQL structure generation depends on analysis of the sentence, while filling the schema items depends on the similarity between schema items and sentence tokens. For example, in Table 2, we test the top models (RAT-SQL (Wang et al., 2020), IRNET (Guo et al., 2019), and GNN (Bogin et al., 2019a)) in the Spider leaderboard, and all these models tend to generate wrong predictions of the type shown. This type of example can be found in the Spider development set where the database id is 'concert_singer'. The example shows that although based on a unified module, there is no strong interaction between generating SQL struc-

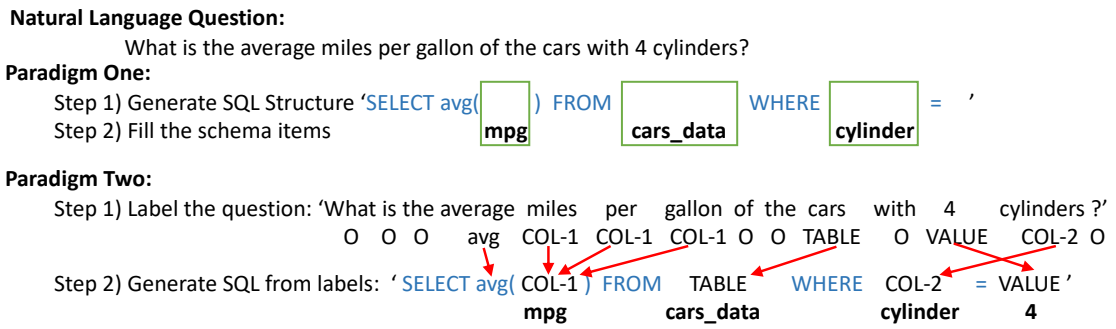

Figure 1: An example of different paradigms

Question:	What is the average salary.
Gold SQL:	**SELECT** average **FROM** salary
Wrong prediction:	**SELECT** avg(average) **FROM** salary

Table 2: A common prediction error

Question:	What are the names of houses properties?
SQL:	**SELECT** name **FROM** Properties **WHERE** type_code = 'House'

Table 3: An example of requirements for DB content

ture (generating the error 'avg' function) and filling the schema item (filling 'average' column) in the models.

3.2 Paradigm Two (Label the question ⇒ Generate SQL)

The WikiSQL SOTA model, IE-SQL (Anonymous, 2020), brings a new paradigm for text-to-SQL. At the time of writing this paper, the paper of IE-SQL is still anonymous. IE-SQL is an information extraction-based text-to-SQL method that tackles the task via sequence labeling, relation extraction, and text matching. IE-SQL first automatically labels the questions by analyzing its corresponding SQL, then trains a neural model to learn how to label a question without a SQL. Finally, IE-SQL can synthesize a SQL from the sequence labeling results in a deterministic way.

Although this approach seems to avoid the problem in Table 2, generating the correct annotation and synthesizing a SQL from the sequence labeling for Spider requires much more work than for WikiSQL because the sentences and SQL queries in the Spider are much more complicated than in the WikiSQL. The same name column that has not appeared in WikiSQL also restricts applying this method on Spider. While it is difficult to use this method on Spider directly, this method brings a new idea to solve the text-to-SQL problem. A method that combines Paradigm One and Paradigm Two is therefore worth thinking about in future.

4 Schema Linking

Schema Linking is a key module used by all the models in these two paradigms. It helps fill the schema items in Paradigm One and generate the 'COL(-∗)' and 'TABLE' label in Paradigm Two.

4.1 Schema Linking Definition

Schema linking is to establish a link between the question token and schema items. There must be a value or weight that guide a model to choose one schema item but not others. We name this value or weight as schema linking value. Any text-to-SQL model with decent performance needs a schema linking value. In Paradigm One approaches, only the schema items strongly related to the question tokens (with high schema linking value) will be filled into the SQL structure. In Paradigm Two, schema linking helps to generate the schema related labels.

4.2 Schema Linking Construction

There are different ways to construct a schema linking. The most common method is to train a neural network model that gives a higher similarity score to the link between a word token in a question and a schema item when they have the same meaning (Iyer et al., 2017). This method is widely used but may have different implementation details.

Some works implement extra schema linking by recognizing the columns and the tables mentioned in a question before training the model (Guo et al., 2019; Bogin et al., 2019a; Wang et al., 2020). It should be noted that Guo et al. (2019) and Wang

et al. (2020) name the extra schema linking as schema linking in the paper while Bogin et al. (2019a) do not mention this extra schema linking but implement it in the code. Extra schema linking is essential in these models because in the ablation study of IRNet and RAT-SQL based on Spider, removing the extra schema linking causes the biggest performance decline compared to removing other removable modules. Since the GNN paper did not present the ablation study on the extra schema linking, we conduct it in Spider and present the results in Table 4. When removing the extra schema linking from the GNN approach we observe a bigger performance decline than when removing the graph neural network component highlighted in the paper. It should be noted that some papers use the abbreviations 'GNN' or 'GNNs' refer to the graph neural networks, but in this paper, 'GNN' only represents the GNN model presented by Bogin et al. (2019a).

Besides, we can improve the extra schema linking through the database (DB) contents where the IRNet, RAT-SQL, GNN models all improve their performance by using the DB contents. For example, in Table 3, without inspecting the content of the database, it is hard to construct a linking between the word `houses` and the column '`property type code`', even by manual, since the word `houses` maybe a redundant word that often appears in questions.

However, most models in WikiSQL do not implement extra schema linking but achieve good performance. We conjecture that this is because the schema items in WikiSQL are much less than in Spider and top models use BERT (Devlin et al., 2019) to build a better schema linking than using other embedding methods. To test this conjecture, we conducted an ablation study on the GNN model and present the results in Table 4. We can see that the performance gap between GNN+BERT and GNN+BERT-ESL is smaller than between GNN and GNN-ESL. We believe BERT can compensate for part of the functionality of extra schema linking. EditSQL (Zhang et al., 2019) is another example that does not use the extra schema linking and achieves a similar improvement (around 20%) as GNN-ESL by extending BERT. It is also the highest improvement done by BERT for the models in the Spider leaderboard.

Model	Exact Match:
GNN	47.6%
GNN - ESL	24.9%
GNN - Graph Neural Networks	41.7%
GNN + BERT	49.3%
GNN + BERT - ESL	47.1%

Table 4: Ablation study results. ESL means extra schema linking.

Question:	What are the names of French singer?
SQL:	**SELECT** name **FROM** singer **WHERE** country = 'France'

Table 5: An example of requirements for pretrained word embeddings

5 Pretrained Word Embeddings

Pretrained word embeddings are also a key module widely used by most models in the two paradigms and two benchmarks. The SOTA models of the two benchmarks, belonging to different paradigms separately, are all based on BERT (Anonymous, 2020; Wang et al., 2020).

It is not surprising that BERT can improve the text-to-SQL models (Wang et al., 2020; Guo et al., 2019). As discussed in the last section, BERT provides a better embedding for schema linking than the original embeddings in the GNN model. Table 5 presents an example to illustrate why we need BERT to construct schema linking. The word '`French`' in the question cannot be constructed a schema linking through DB content since there is only '`France`' in the database. A proper pretrained word embeddings can make the distance between '`French`' and '`country`' shorter than non-pretrained embeddings.

To better understand the contribution of BERT, we list the component F1 score of RAT-SQL with and without BERT in Table 6. Spider metrics define these breakdown components according to SQL clause keywords. Among them, the '`select(no AGG)`' component represent the *SELECT* clause without aggregation function, which only include schema columns. So the F1 score of '`select(no AGG)`' depends on the accuracy of schema items appearing in *SELECT* clause. The score improvement in '`select(no AGG)`' is one more evidence that BERT can improve the schema linking. The improvement on '`keywords`' illustrate that BERT also improves the accuracy of SQL structure generation since the '`keywords`' represent

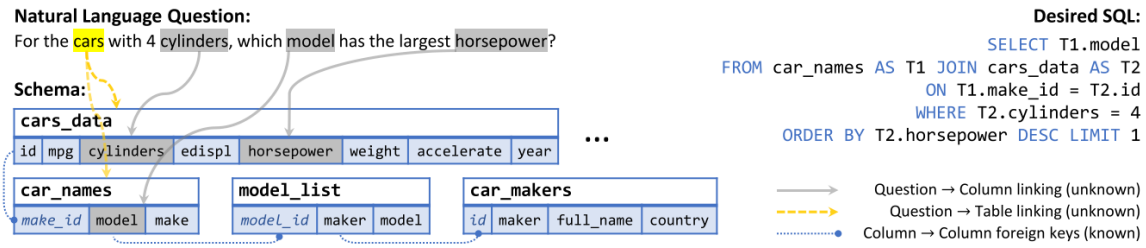

Figure 2: A challenging text-to-SQL task from the Spider dataset. (Wang et al., 2020)

the SQL structure without schema items.

Although BERT can improve the schema linking and SQL structure generation, boosting the performance by extending BERT is computational resource consuming. For example, in Table 4, we simply add the BERT to the GNN model without searching the best hyperparameter, and the performance only improves a little. The SOTA model RAT-SQL in Spider benchmark train 100 times to search the best hyperparameter for extending the BERT, which needs running about 500 days in a single TITAN RTX 24G GPU.

6 Reasoning Assistance Module

Some SQL clauses in Spider need reasoning to generate, but WikiSQL has almost no such clauses. For example, we cannot make out the *JOIN ON* clause directly from the question in Figure 2. Existing models mainly implement reasoning through graph neural networks and intermediate representation.

6.1 Graph Neural Networks

Graph Neural Networks in Existing Models To our knowledge, there is no WikiSQL model using graph neural networks, but some Spider models use it. The reason is that there is only one table in the WikiSQL databases. Every node that came from columns is equivalent in a graph built by only one table. Graph neural networks can not give different information or values to the equivalent nodes, which restrict the usage of graph neural networks in WikiSQL. However, if we build a graph from the table and question tokens, it may work well in WikiSQL, such as the RAT-SQL.

The models in the Spider leaderboard using graph neural networks include GNN (Bogin et al., 2019a), Global-GNN (Bogin et al., 2019b), and RAT-SQL (Wang et al., 2020). The GNN represents a schema as a graph and uses graph neural networks to embed each schema item. There are only schema item embeddings in the graph node of GNN, but the graph node in Global-GNN and RAT-SQL contains embeddings and extra schema linking data. The difference between the graph in Global-GNN and RAT-SQL is that the graph nodes in RAT-SQL include the schema items and question tokens, while nodes in Global-GNN only contain the schema items.

	RAT-SQL	RAT-SQL(BERT)
select	85.3%	90.7%
select(no AGG)	86.4%	92.1%
where	71.7%	77.1%
where(no OP)	76.0%	82.2%
group(no Having)	77.8%	83.0%
group	73.0%	80.0%
order	75.9%	83.9%
and/or	97.9%	97.7%
IUEN	47.6%	53.8%
keywords	85.3%	89.8%

Table 6: F1 scores of component matching of RAT-SQL and RAT-SQL(BERT) on dev set.

Benefit of Using Graph Figure 2 copied from RAT-SQL (Wang et al., 2020) illustrates the challenge of ambiguity in schema linking while '*model*' in the question refers to `car_names.model` rather than `model_list.model`. Graph neural networks can give a bigger schema linking value to the `car_names.model` than `model_list.model` by the uniquely matched `horsepower` column propagating its weight through the schema relations (e.g. foreign keys).

The other benefit of using a graph is to give the schema items that are not mentioned in the question a more significant schema linking value. Examples are often seen in *JOIN ON* clause and subqueries. In Figure 2, it is hard to construct the schema linking from question to the column `cars_data.id` and `cars_names.make_id` that appear in the *JOIN ON* clause of the SQL. The graph neural networks can construct the schema linking value for these two columns from the propagation of other linked

columns.

6.2 Intermediate Representation

Yu et al. (2018a) introduces an intermediate representation (IR) SQL that dispense with *JOIN ON* and *FROM* clauses. This IR can generate full SQL containing *JOIN ON* in a deterministic way by analyzing the schema structure. However, this IR may not generate the correct *JOIN ON* clause when there are more than one available *JOIN ON* clauses or facing the self-join. The RAT-SQL uses both graph neural networks and IR, whose IR only dispense with *JOIN ON* clauses from SQL.

Guo et al. (2019) further proposes an intermediate representation, named SemQL that removes the *GROUP BY* clause and merges the *HAVING* and *WHERE* clauses. SemQL reduces the reasoning work from SQL structure generation that does not significantly benefit from graph neural networks. However, about 20% of the generated *GROUP BY* clauses from SemQL are different from the original, which restricts its performance in Spider exact match metrics. Although most different *GROUP BY* clauses do not affect the accuracy of Spider execution match metrics, SemQL can not generate executable SQL in the current version. IR research still has room for improvement.

7 Conclusion

We discuss the existing cross-domain SOTA text-to-SQL models from the whole to the detailed modules to give a clear picture of the current text-to-SQL research progress. We illustrate that pre-trained embeddings improve the models by constructing a better schema linking and a more accurate SQL structure through experiments. This paper also provide many details that are not mentioned in the original papers, such as . However, due to space limitations, this paper cannot cover all the details of these SOTA models. We hope this paper will help you understand the key connections and differences between the previous models and have a comprehensive understanding of the text-to-SQL field.

8 Future Work

Most questions in Spider and WikiSQL directly use the words related to schema item names instead of synonyms, which means all existing models can build a schema linking by locating the same words. If you want to use these models to implement a natural language interface for database systems, you need to avoid synonyms. However, in some cases, synonyms cannot be avoided, so it is worth studying the text-to-SQL problem in an environment where it is more challenging to build schema linking.

Although only following the Paradigm Two step toward text-to-SQL in Spider needs a lot of works, a method of combining the advantages of two paradigms may boost the performance. For example, we can generate a label to every word token and then use a machine learning model to learn the word tokens with the label to generate SQL.

To improve the text-to-SQL reasoning ability, designing a new IR to simplify SQL structure generation is also a good research topic. Besides, the graph neural networks are all focused on improving the schema linking. How to use graph neural networks to improve SQL structure generation is also worth looking forward to.

Acknowledgements

We would like to thank Denis Newman-Griffis for his meticulous guidance in revising the camera-ready version and the anonymous reviewers for their helpful comments. Matthew Purver is partially supported by the EPSRC under grant EP/S033564/1, and by the European Union's Horizon 2020 programme under grant agreements 769661 (SAAM, Supporting Active Ageing through Multimodal coaching) and 825153 (EMBEDDIA, Cross-Lingual Embeddings for Less-Represented Languages in European News Media). The results of this publication reflect only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

References

Anonymous. 2020. IE-SQL: Text-to-SQL as Information Extraction. *https://drive.google.com/file/d/1t3xEltqKpYJGYekAhQ5vYFen1ocHJ3sY/view*.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy. Association for Computational Linguistics.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Global Reasoning over Database Structures for Text-to-SQL Parsing. 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 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 731–742, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alessandra Giordani and Alessandro Moschitti. 2012. Automatic Generation and Reranking of SQL-derived Answers to NL Questions. In *Proceedings of the Second International Conference on Trustworthy Eternal Systems via Evolving Software, Data and Knowledge*, pages 59–76.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: REINFORCE CONTEXT INTO SCHEMA REPRESENTATION. *https://www.microsoft.com/en-us/research/uploads/prod/2019/03/X_SQL-5c7db555d760f.pdf*.

Wonseok Hwang, Jinyeung Yim, and Minjoon Seo Seunghyun Park. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid Ranking Network for Text-to-SQL. *https://www.microsoft.com/en-us/research/uploads/prod/2020/03/HydraNet_20200311-5e69612887fcb.pdf*.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a Theory of Natural Language Interfaces to Databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 149–157.

Lappoon R Tang and Raymond J Mooney. 2000. Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. *Mathematics of Computation*, 22(103):651.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: Query Synthesis from Natural Language. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM*, pages 63:1—-63:26.

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

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, pages 1050–1055.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*, abs/1709.0.