TECHNISCHE UNIVERSITÄT ILMENAU
Fakultät für Informatik und Automatisierung

Master Thesis Exposé

A Review on State-of-the-art Text-To-SQL Solutions

presented by

Shahriar Yazdipour
Matrikel 62366

Supervisor:

M. Sc. Martin Hofmann
Prof. Dr.-Ing. Patrick Mäder

Ilmenau, July 27, 2022

1 Introduction

Data retrieval in databases is typically done using SQL (Structured Query Language). Text-to-SQL machine learning models are a recent development in state-of-the-art research. The technique is an attractive alternative for many natural language problems, including complex queries and extraction tasks. The text is converted into a SQL query that can be executed on the database. This technique can save time and effort for both developers and end-users by enabling them to interact with databases through natural language queries. With the help of machine learning and knowledge-based resources, text language to SQL conversion is facilitated.

Semantic parsing is a natural language processing that extracts the meaning from text. Text-to-SQL, a type of Semantic Parsing, is a task that converts natural language problems into SQL query statements. This is achieved using machine learning and natural language processing algorithms, and this research is conducted to study different solutions and practices which has been taken by researchers to tackle this problem.

Text-to-SQL allows the elaboration of structured data with information about the natural language text in several domains, such as healthcare, customer service, and search engines. It can be used by data analysts, data scientists, software engineers, and end users who want to explore and analyze their data without learning SQL. It can be used in a variety of ways:

1) Data analysts can use it to generate SQL queries for specific business questions, such as ”What are the top ten products sold this month?”

2) Data scientists can use it to generate SQL queries for machine learning experiments, such as ”How does the price of these products affect their sales?”

3) Businesses can use this technique to automate data extraction and improve efficiency.

4) End-users who want to explore and analyze their data without learning SQL can use it by clicking on a button on any table or chart in a user interface.

Although these models may not solve this problem entirely and perfectly, humans can still struggle with the task. For example, people involved in database migration projects often have to work on schema that they have never seen before.

This research study will review some of the most commonly used NLP technologies relevant to converting text language into Structured Query Language (SQL), and representative models and datasets in the recent solutions for this challenge and their technical implementation.

2 Reseach Question and Work Description

Representative datasets for Text-to-SQL include the WikiSQL[ZXS] dataset and the SPIDER[YZY+] dataset, which contains more complex SQL queries.

The former case consists of Single Table - Multiple Question and the latter case Multiple Table - Multiple Question. First, We will take a shallow look at older datasets and why they are no longer used in Text-to-SQL studies. We will take a look at the difference between these datasets, which made a significant difference in the performance of Text-to-SQL systems.

The top language models and techniques utilized for the Text-To-SQL solution will be studied. We will jump into details of the architecture and backbond of these studies. We will evaluate the success of these researches and how they are different from each other.

After these models' performance review and hardware requirements, it is planned to develop a web application with minimum essentials for average users to access their database with their natural language questions without any SQL knowledge. This application will be open source via Github and accessible for enthusiasts to try this technology on their own.

3 State of the Art

The text-to-SQL problem, or NL2SQL, is defined as the following: Given a Natural Language Query (NLQ) on a Relational Database (RDB), produce a SQL query equivalent to the NLQ. Several challenges include ambiguity, schema linking, vocabulary gaps, and user errors.

It has been a holy grail for the database community for over 30 years to translate user queries into SQL. During this section, we will provide a very brief overview of the earlier approaches, especially those that database communities have proposed.

3.1 Datasets

In this thesis, we will review the Text-to-SQL Challenges and datasets and structure of existing datasets and difference between them. Datasets to be covered are: ATIS, GeoQuery, IMDb, Advising, WikiSQL, Spider.

ATIS (Air Travel Information System) Dataset

A relational schema is used to organize data from the official airline guide in the ATIS corpus. There are 25 tables containing information about fares, airlines, flights, cities, airports, and ground services. All questions related to this dataset can be answered using a single relational query. The relational database uses shorter tables for this dataset to answer queries intuitively.

GeoQuery Dataset

United States geography is represented in the Geoquery dataset. About 800 facts are expressed in Prolog. State, city, river, and mountain information can be found in the database. Geo-

graphic and topographical attributes such as capitals and populations make up the majority of the attributes.

IMDb Dataset

The IMDb dataset contains 50K reviews from IMDb. There is a limit of 30 reviews per movie[MDP+11]. Positive and negative reviews are equally represented in the dataset. The dataset creators considered a negative review with a score of 4 out of 10 and a positive review with a score of 7 out of 10. When creating the dataset, neural reviews are not taken into account. Furthermore, Training and testing datasets are equally divided.

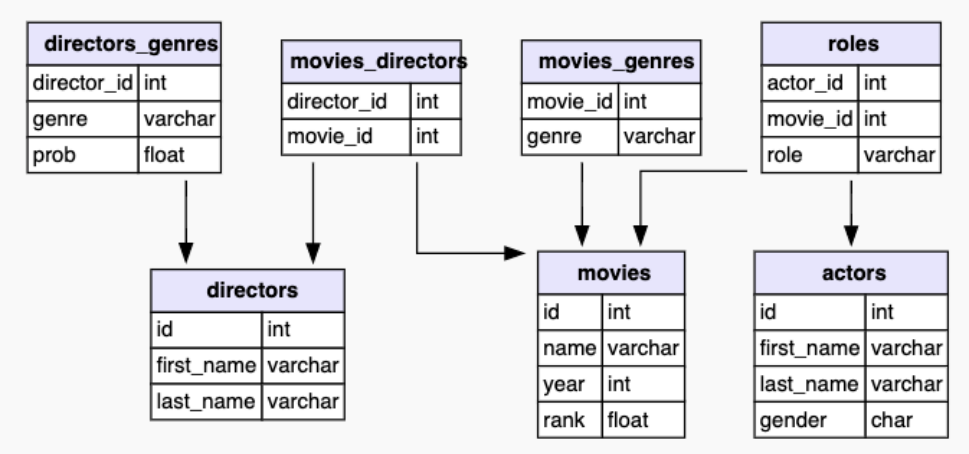

Figure 1: Database Structure of IMDb dataset

Advising Dataset

The Advising dataset was created in order to propose improvements in text2SQL systems. The creators of the dataset compare human-generated and automatically generated questions, citing properties of queries that relate to real-world applications. Dataset consists of questions from university students about courses that lead to particularly complex queries. The database contains fictional student records. The dataset includes student profile information, such as recommended courses, grades, and previous courses. In an academic advising meeting, students were asked to formulate questions they would ask if they knew the database. Many of the queries in this dataset were the same as those in ATIS, GeoQuery, and Scholar.

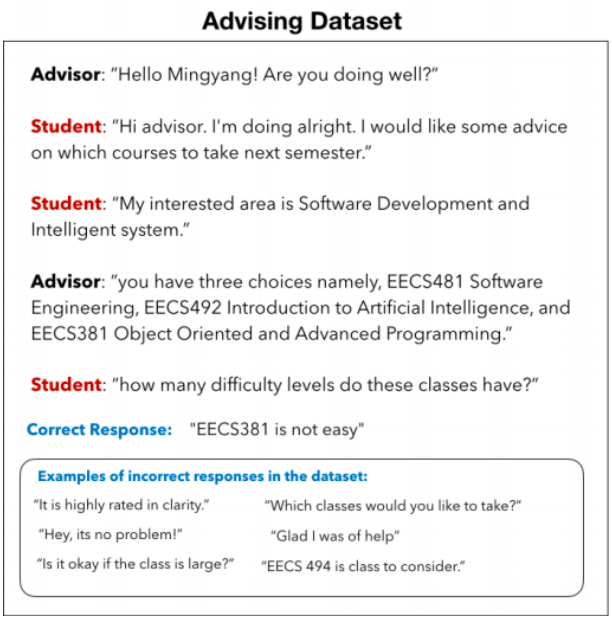

Figure 2: Example from Advising dataset [VR19]

WikiSQL Dataset

WikiSQL consists of 80K+ natural language questions and corresponding SQL queries on 24K+ tables extracted from Wikipedia. Neither the train nor development sets contain the database in the test set. Databases and SQL queries have simplified the dataset's creators' assumptions. This dataset consists only of SQL labels covering a single SELECT column and aggregation and WHERE conditions. Furthermore, all the databases contain only one table.

The database does not include complex queries involving advanced operations like JOIN, GROUP BY, ORDER BY, etc. Prior to the release of SPIDER, this dataset was considered to be a benchmark dataset. Using WikiSQL has been the subject of a great deal of research. WikiSQL's "WHERE" clause has been recognized as one of the most challenging clauses to parse semantically, and SQLNet and SyntaxSQL were previous state-of-the-art models.

Table:

Player	Country	Points	Winnings ($)
Steve Stricker	United States	9000	1260000
K.J. Choi	South Korea	5400	756000
Rory Sabbatini	South Africa	3400	4760000
Mark Calcavecchia	United States	2067	289333
Ernie Els	South Africa	2067	289333

Question: What is the points of South Korea player?

SQL: `SELECT Points WHERE Country = South Korea`

Answer: 5400

Figure 3: Example from WikiSQL dataset[**?**]

Spider Dataset

Yale University students created this dataset. The SPIDER database contains 10K questions and 5K+ complex SQL queries covering 138 different domains across 200 databases. As opposed to previous datasets (most of which used only one database), this one incorporates multiple datasets. Creating this corpus was primarily motivated by the desire to tackle complex queries and generalize across databases without requiring multiple interactions.

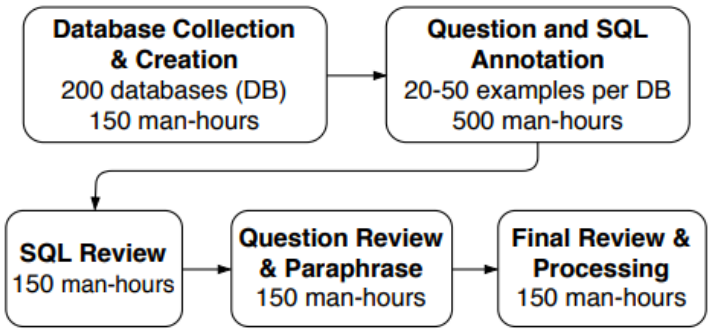

Figure 4: Example from Spider dataset[YZY+]

Creating a dataset involves three main aspects: SQL pattern coverage, SQL consistency, and question clarity. Several databases from WikiSQL are included in the dataset. The table is complex as it links several tables with foreign keys. In SPIDER, SQL queries include: SELECT with multiple columns and aggregations, WHERE, GROUP BY, HAVING, ORDER BY, LIMIT, JOIN, INTERSECT, EXCEPT, UNION, NOT IN, OR, AND, EXISTS, LIKE.

SPIDER's exact matching accuracy was 12.4% compared to existing state-of-the-art models. As a result of its low accuracy, SPIDER presents a strong research challenge. Current SPIDER accuracy is around 75.5% with an exact set match without values (refers to values in the WHERE clause) and around 72.6% with values.

Complex question

What are the name and budget of the departments with average instructor salary greater than the overall average?

Complex SQL

```
SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as
T2 ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) >
    (SELECT avg(salary) FROM instructor)
```

Figure 5: Example of Question-Query set from SPIDER[YZY+]

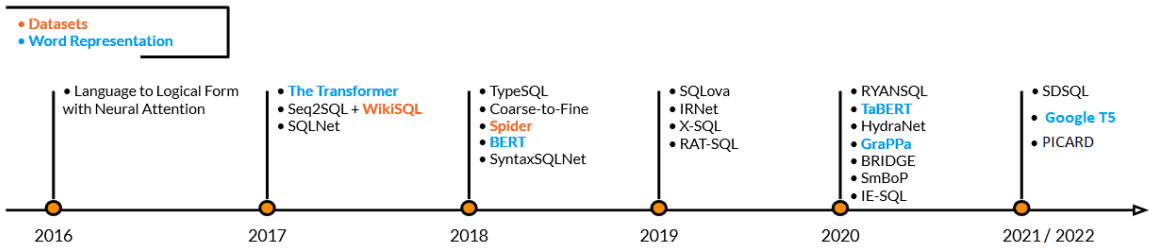

Figure 6: An overview of the deep learning process for Text-to-SQL.

3.2 Models

An efficient text-to-SQL solution requires state-of-the-art natural language processing techniques. As a result of the neural network's ability to handle only numerical inputs and not raw text, word embedding has been used to represent numerical words.

Aside from that, in the past few years, language models have become increasingly popular as a solution for increasing performance in natural language processing tasks.

Assuming that words have numerical representations that differ from those of other words, word embeddings aim to map each word to a multidimensional vector, incorporating valuable information about the word. In addition to the brute-force creation of one-hot embeddings, researchers have developed highly efficient methods for creating representations that convey a word's meaning and relationships with other words. In most, if not all, Text-to-SQL systems, word embedding techniques such as Word2Vec[Ron14], GloVe, and WordPiece embeddings[WSC+16] are used.

Recently Language models have been shown to excel at NL tasks as a new type of pre-trained neural network. It is important to note that language models are not a replacement for word embeddings since they are neural networks and need a way to transform words into vectors.

Depending on the specific problem they want to solve, researchers can adapt the pre-trained model's inputs and outputs and train it for an additional number of epochs on their dataset. Thus, we can achieve state-of-the-art performance without complex architectures [DCLT18]. Recent neural network architectures, like the Transformer[VSP+17], have been used to achieve such

performance by these models, which excel at handling NL and sequences of NL that are characterized by connections between words. Several language models have been used to handle the text-to-SQL task, including BERT [DCLT18] and MT-DNN [LHCG19], while new models pre-trained specifically for structured data tasks are emerging, such as TaBERT[YNYR20] and GraPPa [YWL+20].

The research section will assess the best state-of-the-art research in this field, starting from Seq2SQL[ZXS] study in 2017 with the hype in WikiSQL challange and we will continue with SQLova[**?**] SQLNet[XLS] and indtroduction of transformers and BERT with focus on RAT-SQL[WSL+] (2019), BRIDGE[LSX] with BERT, HydraNet[LCH+20] (2020), and the most recent solution with Google T5[RSR+], PICARD[SSB] in SPIDER (2021).

After reviewing the research papers on these models, we will study the implementation steps of these models. Moreover, evaluation methods and approaches to compare these models in accuracy for different datasets and if they are usable and reliable enough for our usage.

Most of these studies have excellent documentation regarding their implantation. Execution of these studies will be documented and published on Github. Nonetheless, In case of old and impractical implementation instructions, we will skip the implementation and continue with the top models available.

Seq2SQL

An output of a sequence-to-sequence approach is a sequence of SQL tokens and schema elements, with that sequence being used to predict SQL queries or at least a significant portion of them. An NLQ sequence is transformed into a SQL sequence by these programs. There is no doubt that this approach is the simplest, but it is also the most error-prone. Seq2SQL[ZXS], one of the first deep-learning systems, used this approach, but later, systems avoided it. sequence-to-sequence architectures have the major disadvantage of not taking the strict grammar rules of SQL into account when generating queries.

As part of this model, its authors released the WikiSQL dataset, which ushered in a new era of text-to-SQL deep learning research. GloVe embeddings represent the inputs in the network architecture, which combines LSTM and linear layers. With a seq-to-seq network, the system predicts the aggregation function and the column for the SELECT clause. Its major drawback is that it generates parts of the query that can lead to syntactic errors.

SQLNet

The model was designed to demonstrate that reinforcement learning should be limited in Text2SQL tasks. Until SQLNet[XLS], all previous models used reinforcement learning to improve the decoder results when it generated appropriate serializations.

In cases where order is irrelevant, SQLNet avoids the seq2seq structure. For making predictions, the model uses a sketch-based approach consisting of a dependency graph that allows previous predictions to be taken into account. To improve the results, the model also incorporates column attention (weights assigned to significant words and phrases in sentences). According to the flowchart below, SQLNet employs three phases to generate SQL queries for WikiSQL

tasks.

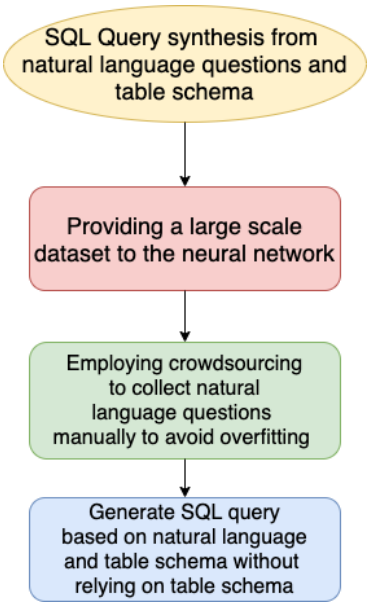

Figure 7: SQLNet

Sketch-based query synthesis

The token with the $ sign represents an empty slot, and the token name represents the type of prediction. Tokens in bold represent SQL keywords such as SELECT, WHERE, etc. $AGG can be filled with either an empty token or one of the aggregation operators, such as SUM or MAX. Fill in the $COLUMN and $VALUE slots with the column name and substring of the question, respectively. The $OP slot can be a value between {=, }. The notion ...* uses a regular expression to indicate zero or more AND clauses.

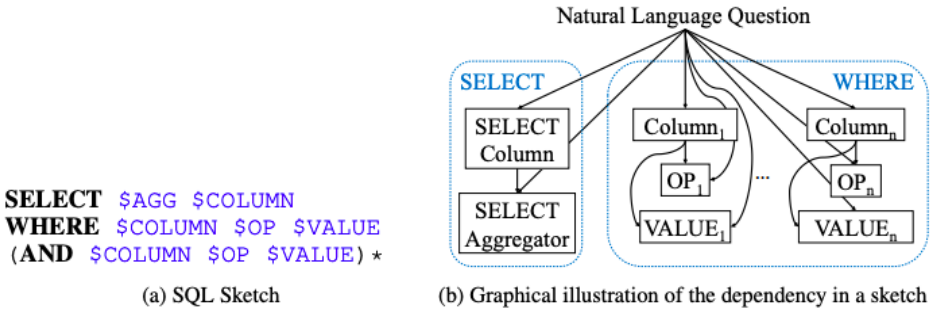

Figure 8: Sketch-based query synthesis

Column attention for sequence-to-set prediction

Instead of producing a sequence of column names, sequence-to-set prediction predicts the names of the columns of interest. Based on column names, column attention is part of the generic attention mechanism for computing the feature attention map on a question.

Predicting WHERE and SELECT clause

One of the most challenging tasks in Text2SQL is predicting the WHERE clause. According to SQL sketch, SQLNet finds the columns that appear in the WHERE clause and predicts the OP slots and value for each column.

It is predicted that the OP slot will be filled with one of the three classes {¡,¿,=}, and the VALUE slot will be filled with the substring from the natural language question. In SELECT clauses, columns are named, and aggregator functions are specified, such as count, sum, max, etc. There is only one difference between SELECT and WHERE: the column name. There is only one column selected in SELECT.

In the WikiSQL test set, SQLNet accuracy is 64.4%, and in the SPIDER test set, it is around 12.4%.

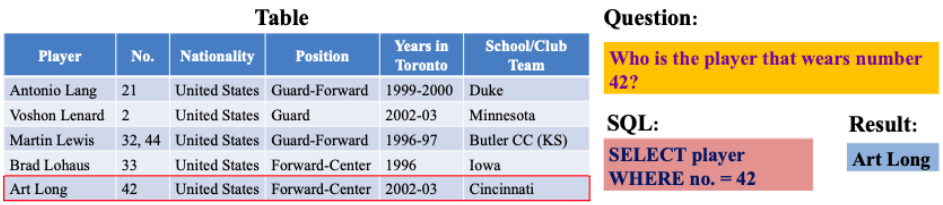

Figure 9: An example of a query executed by SQLNet on WikiSQL

SyntaxSQLNet

- The main goal of developing the SyntaxSQLNet model was to generate complex SQL queries with multiple clauses and generalize them to new databases.
- The model is based on a syntax tree network to address complex and cross-domain queries. The encoders are table-aware, and the decoders have a history of the SQL generation path.
- With a massive 7.3% improvement in accuracy, SyntaxSQLNet outperformed previous models, such as SQLNet, on the SPIDER dataset.

- A cross-domain data augmentation technique further improves accuracy by generating more variance during training.
- Below is a chart showing the various modules and their functions.

SQL Grammar and Attention Mechanism

- In order to enable the decoder to handle complex queries, SQL grammar is used. At each step of recursive decoding, it determines which module to invoke.
- Predicting the next SQL token is also based on the history of SQL path generation and current SQL tokens.

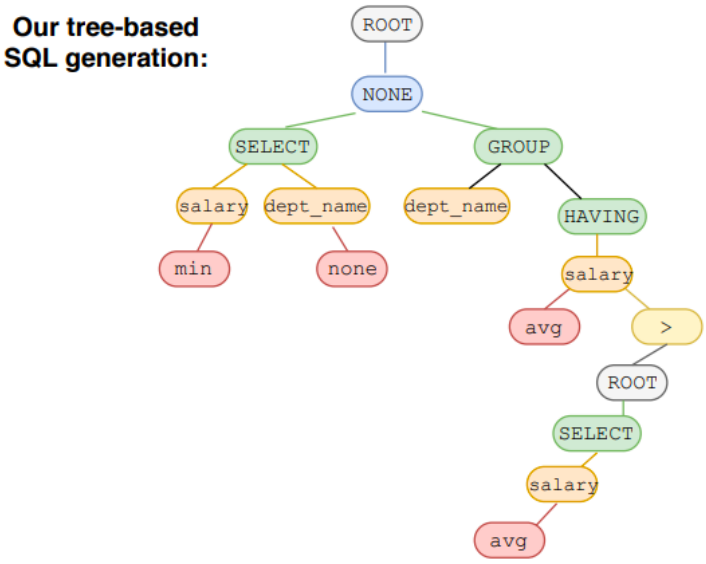

Figure 10: Tree-based SQL generator in SyntaxSQLNet

- The attention mechanism is also used to encode the question representation. Attention also applies to SQL path history encoding.

Data Augmentation

- Despite SPIDER's large dataset, it lacks complex queries.
- For proper generalization, cross-domain datasets are used for data augmentation.
- Various training databases of the SPIDER dataset are used to prepare a list of patterns for natural language questions and corresponding SQL queries.

The SPIDER model using syntaxSQLNet decoding history reaches 27.2% accuracy.

Compared to previous models, such as SQLNet, the accuracy increased by 15%.

GrammarSQL

Sequence-to-sequence models for neural text-to-SQL typically perform token-level decoding and do not consider generating SQL hierarchically.

- [XLS] proposes a grammar-based model for reducing the complexity of text2SQL tasks involving hierarchical grammars.
- The authors introduce schema-dependent grammar with minimal over-generation.
- The grammar developed in [XLS] covers 98% of the instances in ATIS and SPIDER datasets.

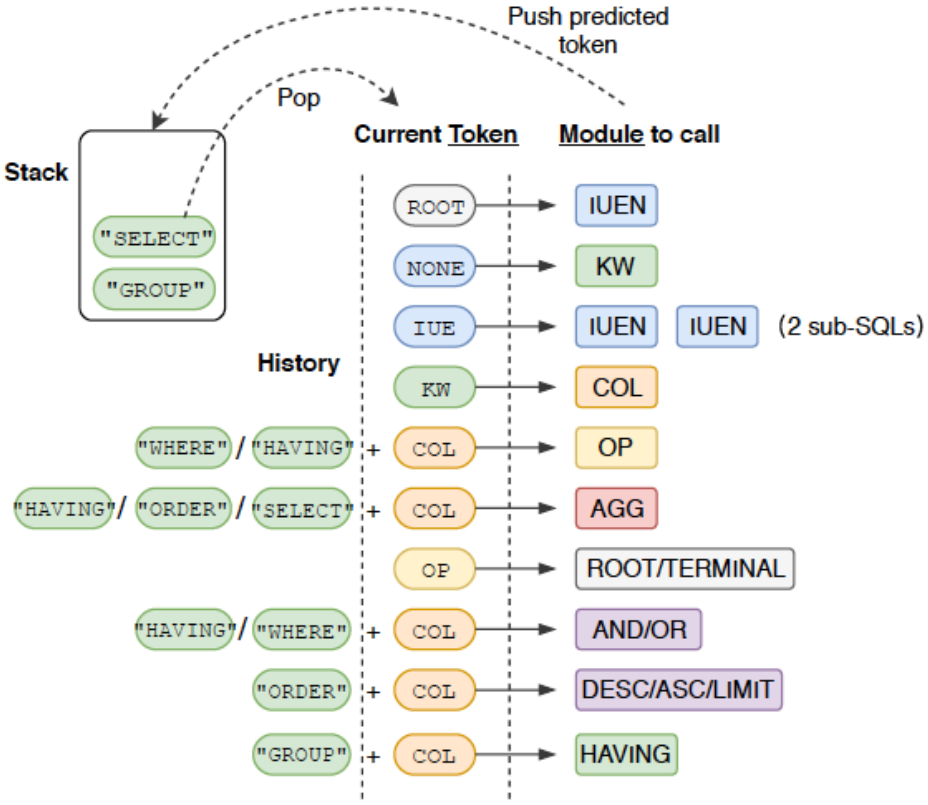

Figure 11: Modules defined in SyntaxSQLNet model

SQL Grammar

- The shallow parsing expression grammar aims to capture as little SQL as possible to cover most instances in the dataset.
- In order to ensure consistency of table, column, and value references in SQL, the authors added non-terminals to context-free grammar.
- They use runtime constraints during decoding to ensure that only valid programs can be used to join different tables in DB together using a foreign key.

Few details on the proposed model

- As an input, the proposed model takes an utterance of natural language, a database, and grammar about that utterance.
- String matching heuristics are applied after taking the input to link words in the input to identifiers or tokens in the database.
- Afterward, the bidirectional LSTM receives a concatenated string of the learned word and the link embeddings for each token.
- Using the attention mechanism, the decoder builds up the SQL query iteratively on the input sequence.

- Database identifiers in natural language questions and SQL queries are also anonymized.

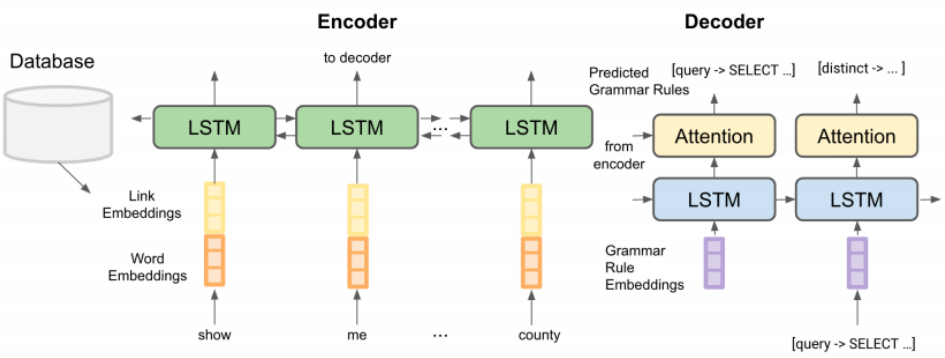

Figure 12: Structure of the proposed model

On ATIS and SPIDER datasets, the GrammarSQL model was evaluated. By 14%, it outperformed SyntaxSQLNet, the previous SOTA model.

IRNet

- In Text2SQL tasks, the Intermediate Representation Network (IRNet) addresses two main challenges.
- Among the challenges are mismatches between natural language intents and predicting columns resulting from a more significant number of out-of-domain words.
- Instead of synthesizing SQL queries end-to-end, IRNet decomposes natural language into three phases.
- Schema linking is performed over a database schema and a question during the first phase.
- IRNet uses SemQL to bridge the gap between SQL and natural language.
- It includes a Natural Language (NL) encoder, a Schema Encoder, and a Decoder.

- The model provides different functions to accomplish Text2SQL tasks.
- Natural language is encoded into an embedding vector by the NL encoder. By using a bi-directional LSTM, these embedding vectors are used to construct hidden states.
- A schema encoder takes a database schema as input and outputs representations for columns and tables.
- Using a context-free grammar, the decoder synthesizes SemQL queries.
- On the SPIDER dataset, IRNet performs 46.7% better than previous benchmark models by 19
- The accuracy of 54.7% is achieved by combining IRNet with BERT.

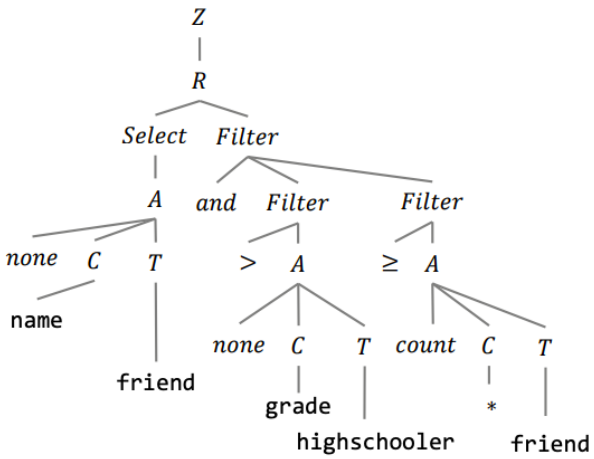

Figure 13: An illustrative example of SemSQL from [GZG+19]

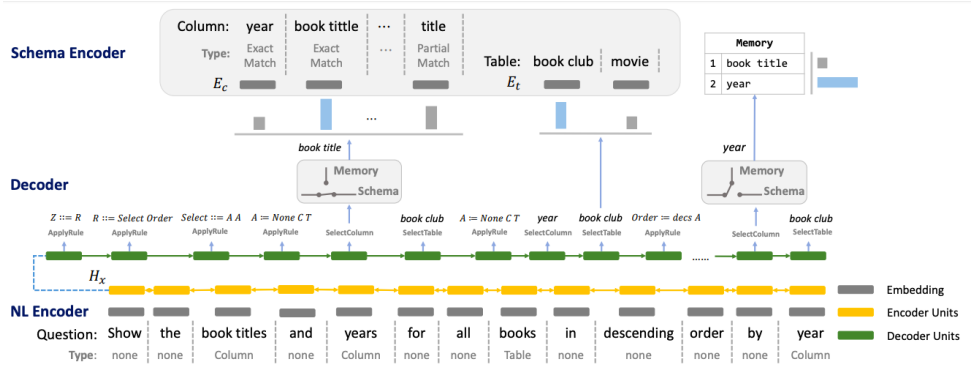

Figure 14: An overview of the neural model proposed in [GZG+19]

EditSQL

- EditSQL focuses on text-to-SQL tasks that are context-dependent across domains.
- It exploits the fact that adjacent natural language questions are dependent on one another and that corresponding SQL queries overlap.
- To improve the generation quality, they edit the previously predicted query.
- The editing mechanism reuses generation results at the token level based on SQL input sequences.
- An utterance-table encoder and a table-aware decoder are utilized to incorporate the context of the natural language and the schema when dealing with complicated tables in different domains.

- User utterances and table schemas are encoded by the utterance-table encoder. Tokens of utterances are encoded using a bi-LSTM.
- To determine the most relevant columns, Attention weighed an average of column header embedding is applied to each token.

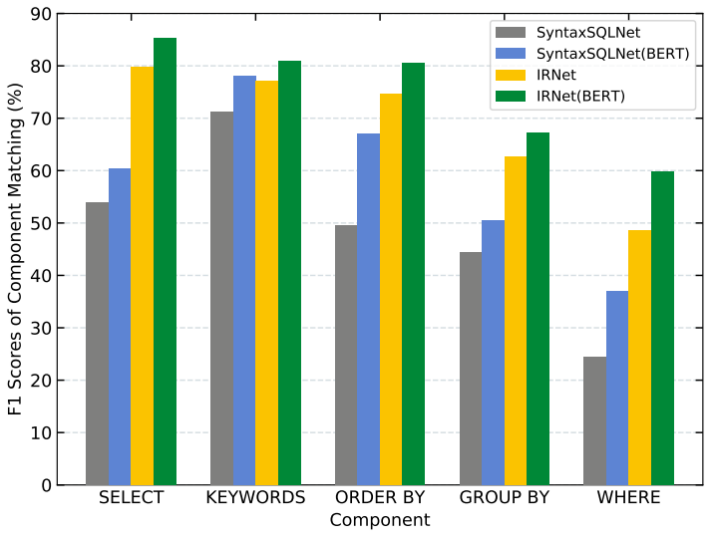

Figure 15: F1 scores of component matching of SyntaxSQLNet, SyntaxSQLNet$BERT$, IRNet and IRNet$BERT$ on the test set from [GZG+19]

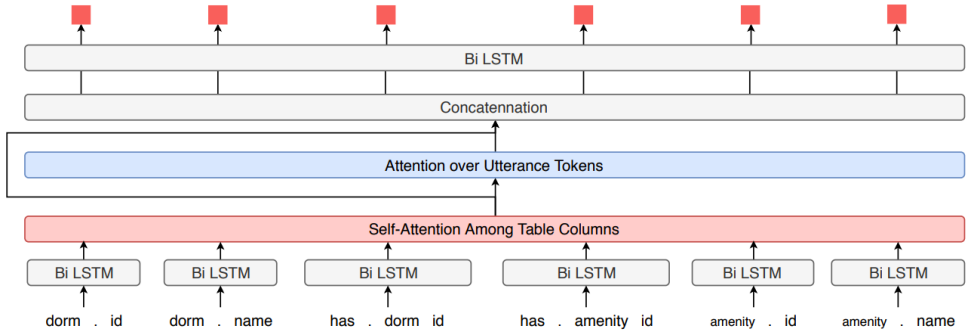

Figure 16: The model architecture of EditSQL from [ZYE+19a]

- To capture the relationship between table schema and utterance, an attention layer is incorporated.
- The utterance-level encoder is built on top of an interaction-level decoder in order to capture information across utterances.
- LSTM decoding is used to generate SQL queries by incorporating interaction history, table schema, and user utterances.

- The model is evaluated on the SParC dataset, a large cross-domain context-dependent semantic parsing dataset derived from SPIDER.
- In both SPIDER and SParC, the model outperforms the previous state of the art model, IRNet.
- In cross-domain text2SQL generation, the model achieves 32.9% accuracy. A 53.4% improvement in accuracy can be achieved by using BERT embedding.

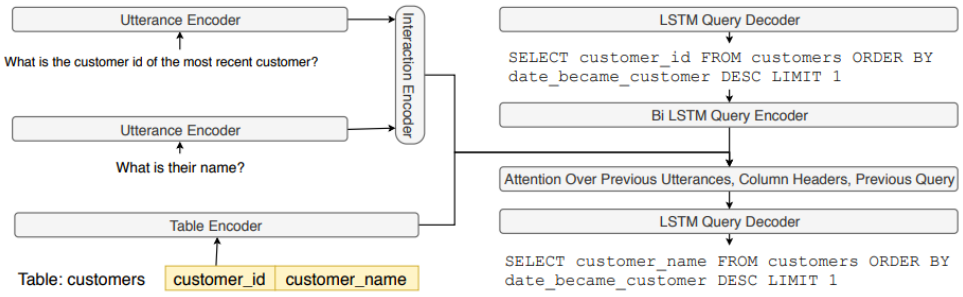

Figure 17: An example of user utterance and column headers and Utterance Encoder from [ZYE+19a]

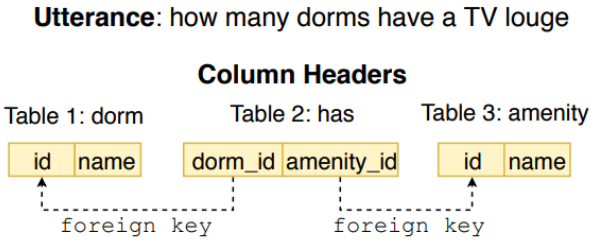

Figure 18: Table Encoder from [ZYE+19a]

RAT-SQL

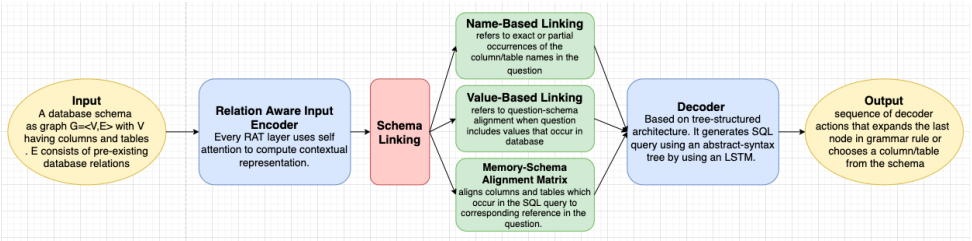

Figure 19: A flow chart of RAT-SQL model

- A major challenge in translating natural language queries into SQL queries is generalizing them to unknown database schemas.
- As part of the generalisation, it is necessary to encode database relations in an accessible way and model alignment between relevant database columns in the query.
- Within a text2SQL encoder, the proposed framework leverages the relation-aware self-attention mechanism to encode address schemas, represent features, and link schemas.
- Check out the flow chart below for an overview of RAT-SQL's encoder-decoder structure.

On the SPIDER dataset, RAT-SQL achieves 57.2% accuracy, an improvement of 8.7% over previous benchmark models.

With RAT-SQL, 65.6% accuracy can be achieved by combining BERT with RAT-SQL.

Model	Dev	Test
IRNet (Guo et al., 2019)	53.2	46.7
Global-GNN (Bogin et al., 2019b)	52.7	47.4
IRNet V2 (Guo et al., 2019)	55.4	48.5
RAT-SQL (ours)	**62.7**	**57.2**
With BERT:		
EditSQL + BERT (Zhang et al., 2019)	57.6	53.4
GNN + Bertrand-DR (Kelkar et al., 2020)	57.9	54.6
IRNet V2 + BERT (Guo et al., 2019)	63.9	55.0
RYANSQL V2 + BERT (Choi et al., 2020)	**70.6**	60.6
RAT-SQL + BERT (ours)	69.7	**65.6**

Figure 20: Accuracy on the Spider development and test sets, compared to the other approaches at the top of the dataset leaderboard as of May 1st, 2020 from [WSL+]

Split	Easy	Medium	Hard	Extra Hard	All
RAT-SQL					
Dev	80.4	63.9	55.7	40.6	62.7
Test	74.8	60.7	53.6	31.5	57.2
RAT-SQL + BERT					
Dev	86.4	73.6	62.1	42.9	69.7
Test	83.0	71.3	58.3	38.4	65.6

Figure 21: Accuracy on the Spider development and test sets, by difficulty from [WSL+]

BRIDGE

Seq-to-seq approaches such as Bridge[LSX] have been rare in recent times. Additionally, it implements schema-consistency-guided decoding to avoid errors at prediction time, incorporating BERT for better NL processing and fuzzy string matching for schema linking. For instance, the system is forced to predict SQL keywords in a specific order. If the table name has not already been generated, it will not be able to generate column names.

Picard

Picard[SSB] stands for "Parsing Incrementally for Constrained Auto-Regressive Decoding.". It can be used with any existing language model decoder or vocabulary based on auto-regressive language modeling.

As an optional feature, Picard can be enabled at inference time and is absent from pre-training or fine-tuning. In the case of text-to-SQL translation, Picard operates directly on the output of the language model. Picard demonstrates state-of-the-art performance on challenging Spider and CoSQL text-to-SQL translation tasks.

Picard warps model prediction scores and integrates trivially with existing greedy and beam search algorithms. In addition to the token ids of the current hypothesis, the model's language modeling head also predicts the log-softmax scores for each vocabulary token. Additionally, Picard has access to SQL schema information, including table and column names and which column resides in which table.

3.3 Evaluation method

Text-to-SQL tasks can be evaluated by two methods: accurate matching rate and execution accuracy rate. Predicted SQL statements are compared with standard statements to determine how accurate the match is. By splitting the predicted SQL statement and definitive statement into multiple clauses according to keywords, we can solve the problem of matching errors caused by order of the where clause. The matching is successful as long as the elements in both sets are the same.

$$Acc_{qm} = \frac{Successful\ matching\ of\ predicted\ SQL\ statement\ with\ standard}{All\ Questions}$$

When using the correct predicted SQL statements, the correct execution rate refers to the proportion of questions that can receive the correct answers from the database.

$$Acc_{ex} = \frac{The\ right\ question}{All\ Questions}$$

By predicting the key F1 values for SQL statements, the model can also be evaluated.

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

4 Outline

1. Introduction
   - Description
   - Applications
2. Related Works and Motivation
3. Datasets
   - Datasets and Challenges
   - WikiSQL Benchmark
   - CoSQL Benchmark
   - SPIDER Benchmark

4. Models and State of the Art approaches
    - Theoretical background and the review of world literature
    - Natural Language Processing Concepts
    - Transformers and Google T5 [RSR+20]
    - Evaluation Metrics [VR19]
    - Study Different Researches in Text-to-SQL Challenges
        - Seq2SQL [ZXS]
        - SQLNet [XLS]
        - SyntaxSQLNet [YYY+18]
        - EditSQL [ZYE+19b]
        - RAT-SQL [WSL+]
        - RYANSQL [CSKS]
        - HydraNet [LCH+20]
        - BRIDGE [LSX]
        - T5 - PICARD [SSB]
        - Alternative approaches
        - Challenges and Research Opportunities
5. Results and Discussion
6. Implementation
    - Implementation Details
    - Implementation Online Applications of a Model
7. Summary and Future Work
    - Implement and test on private dataset
    - Discussion of the Results
    - Effect on other researches, like Text-to-SPARQL
    - Conclusions
8. Bibliography

5 Timeline

The timeline for the project will be broken down into four phases: literature review, establishing the theoretical framework and research implementation, using some datasets on a state-of-the-art model, and finally, documenting our study.

Phase	Description
Phase Zero	**Working on Expose**: Background research and preparing proposal
Phase One	**Literature Review**: The first phase will be a review of any literature considering Text-to-SQL solutions and challenges. It includes examining how other researchers have approached the topic and discussing challenges around this research problem. Many considerations need to be addressed in this phase, such as the goal, the data, and how researchers find solutions for such a task. Each week I will try to focus on one state-of-the-art technique.
Phase Two	**Theoretical Framework and Research implementation**: The second phase is where A theoretical framework of a model will be established and implemented. In order to do this, a variety of different approaches will be explored, which are possible to implement with our hardware and legal license limitations.
Phase Three	**Software Development of a Text-to-SQL prototype**: Here, the previously collected dataset will be used to test our model and validate the accuracy. It is planned to have an online web application that can showcase Text-To-SQL scenarios with custom input for users.
Final Phase	**Documenting our study**: It will be planned to finish a draft regarding our study and our outcome in a month, and getting prepared for thesis defense and presentation.

The following timeline is a rough draft of what I would like to do for this project. It is not a complete timeline, and it is subject to change.

Week Nr.	Month	Goal
33	August 2022	Register for Master Thesis
34-35	August 2022	Research on Related Work and Datasets and going into more details of their structure
36-37	September 2022	Research on WikiSQL Language models and challenges
38-39	September 2022	Research on SPIDER Language models and challenges
40-41	October 2022	Research on SPIDER Language models and challenges
42	October 2022	Collecting all gained information and Documenting it
43-44	October 2022	Research on Evaluation techniques and overall comparison
45-46	November 2022	Working on technical implementation of these state of the art researches
47-48	November 2022	Working on development of the text-to-sql software
49	December 2022	Work on deployment and publishing the web application
50	December 2022	Evaluate and completing the work already done before Christmas Holidays
2-3	January 2023	Working on Documentation of the Report
4-5	January 2023	Working on Documentation of the Report
6	February 2023	Final Evaluation and Review of the Thesis Report
7	February 2023	Submitting the final Master Thesis

References

[CSKS] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. version: 1.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[GZG+19] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. *CoRR*, abs/1905.08205, 2019.

[LCH+20] Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. Hybrid ranking network for text-to-SQL. 2020.

[LHCG19] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504, 2019.

[LSX] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing.

[MDP+11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[Ron14] Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.

[RSR+] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[SSB] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models.

[VR19] Jesse Vig and Kalai Ramea. In *Comparison of Transfer-Learning Approaches for Response Selection in Multi-Turn Conversations*, 2019.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[WSL+] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers.

[XLS] Xiaojun Xu, Chang Liu, and Dawn Song. SQLNet: Generating structured queries from natural language without reinforcement learning.

[YNYR20] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *CoRR*, abs/2005.08314, 2020.

[YWL+20] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. Grappa: Grammar-augmented pre-training for table semantic parsing. *CoRR*, abs/2009.13845, 2020.

[YYY+18] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir R. Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. *CoRR*, abs/1810.05237, 2018.

[YZY+] 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. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task.

[ZXS] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. version: 6.

[ZYE+19a] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. Editing-based SQL query generation for cross-domain context-dependent questions. *CoRR*, abs/1909.00786, 2019.

[ZYE+19b] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. Editing-based SQL query generation for cross-domain context-dependent questions. *CoRR*, abs/1909.00786, 2019.