TECHNISCHE UNIVERSITÄT ILMENAU
Fakultät für Informatik und Automatisierung

Master Thesis

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, January 16, 2023

Dedication

I dedicate this thesis to the brave and heroic Iranian women who have stood up against oppression and fought for their rights and freedoms. These women, often at significant personal risk, have courageously spoken out against the injustices they have faced and have worked tirelessly to bring about positive change in their country.

Their tireless efforts and dedication to the cause of gender equality and social justice have inspired me and countless others worldwide. I am deeply grateful for their unwavering commitment to making the world a better place for all.

This thesis is also dedicated to the memory of those who have lost their lives in the struggle for equality and justice. Their sacrifice will never be forgotten, and their legacy will inspire future generations to fight for a more just and equitable world.

Also, I express my deepest gratitude to my parents, who have always been my biggest supporters and have believed in me throughout my academic journey. Their love, guidance, and encouragement have been invaluable to me.

Finally, I am also grateful to my professor, Prof. Patrick Mäder and also Martin Hofmann, who has been excellent mentor and guide throughout the process of writing this thesis. Their expertise and support have been instrumental in helping me to complete this work.

Contents

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 Motivation

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 Background and Literature Review

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 Early Approaches

3.2 Recent Approaches

4 The Main Evaluation Datasets and Challenges

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.

4.0.1 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.

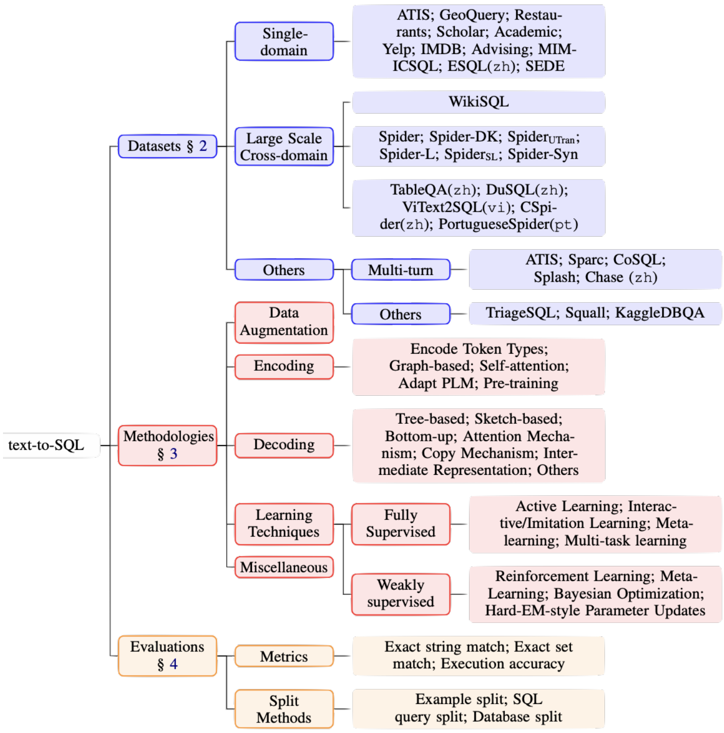

text-to-SQL
Datasets § 2
Single-domain
ATIS; GeoQuery; Restaurants; Scholar; Academic; Yelp; IMDB; Advising; MIMICSQL; ESQL(zh); SEDE
Large Scale Cross-domain
WikiSQL
Spider; Spider-DK; Spider$_{UTran}$; Spider-L; Spider$_{SL}$; Spider-Syn
TableQA(zh); DuSQL(zh); ViText2SQL(vi); CSpider(zh); PortugueseSpider(pt)
Others
Multi-turn
ATIS; Sparc; CoSQL; Splash; Chase (zh)
Others
TriageSQL; Squall; KaggleDBQA
Methodologies § 3
Data Augmentation
Encoding
Encode Token Types; Graph-based; Self-attention; Adapt PLM; Pre-training
Decoding
Tree-based; Sketch-based; Bottom-up; Attention Mechanism; Copy Mechanism; Intermediate Representation; Others
Learning Techniques
Fully Supervised
Active Learning; Interactive/Imitation Learning; Meta-learning; Multi-task learning
Weakly supervised
Reinforcement Learning; Meta-Learning; Bayesian Optimization; Hard-EM-style Parameter Updates
Miscellaneous
Evaluations § 4
Metrics
Exact string match; Exact set match; Execution accuracy
Split Methods
Example split; SQL query split; Database split

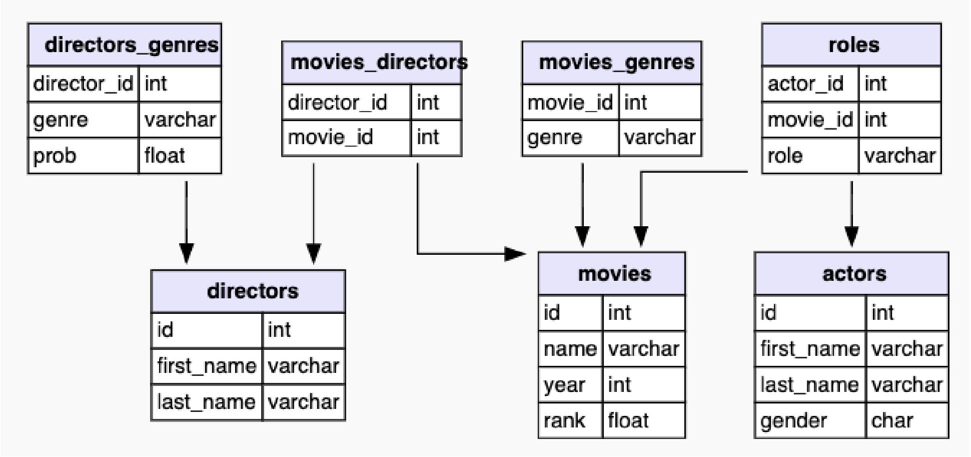
Figure 1: Mindmap of the state-of-the-art

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

4.0.3 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.



directors_genres	
director_id	int
genre	varchar
prob	float

movies_directors	
director_id	int
movie_id	int

movies_genres	
movie_id	int
genre	varchar

roles	
actor_id	int
movie_id	int
role	varchar

directors	
id	int
first_name	varchar
last_name	varchar

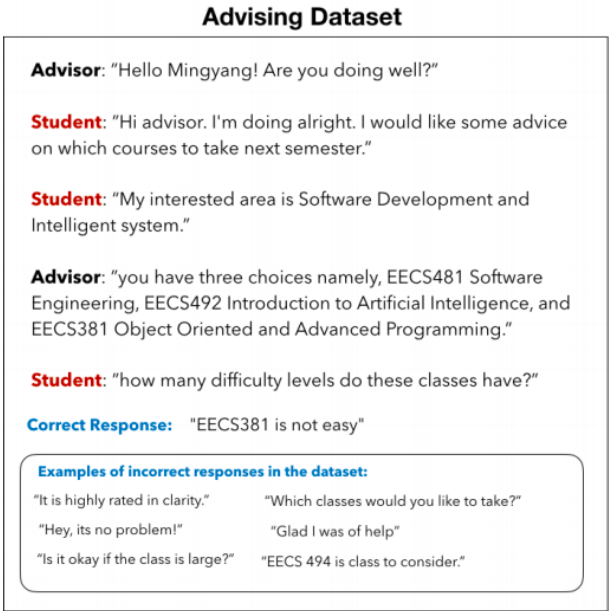
movies	
id	int
name	varchar
year	int
rank	float

actors	
id	int
first_name	varchar
last_name	varchar
gender	char

Figure 2: Database Structure of IMDb dataset

4.0.4 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.



Advising Dataset

Advisor: "Hello Mingyang! Are you doing well?"

Student: "Hi advisor. I'm doing alright. I would like some advice on which courses to take next semester."

Student: "My interested area is Software Development and Intelligent system."

Advisor: "you have three choices namely, EECS481 Software Engineering, EECS492 Introduction to Artificial Intelligence, and EECS381 Object Oriented and Advanced Programming."

Student: "how many difficulty levels do these classes have?"

Correct Response: "EECS381 is not easy"

Examples of incorrect responses in the dataset:

"It is highly rated in clarity." "Which classes would you like to take?"

"Hey, its no problem!" "Glad I was of help"

"Is it okay if the class is large?" "EECS 494 is class to consider."

Figure 3: Example from Advising dataset [VR19]

4.0.5 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.

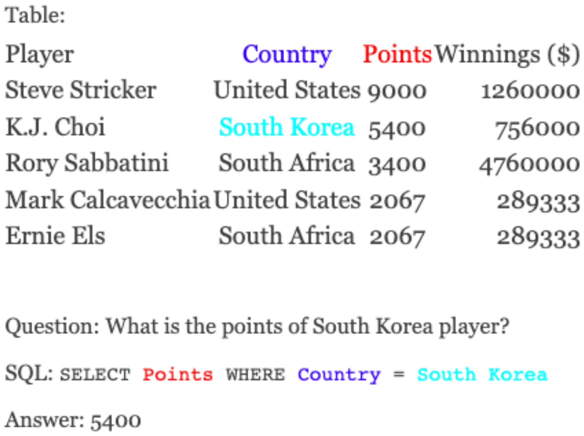


Table:

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 4: Example from WikiSQL dataset[**?**]

4.0.6 WikiSQL Challenge

The WikiSQL challenge is a research competition focused on developing natural language interfaces for databases. The challenge includes several state-of-the-art Text-to-SQL solutions proposed by different research teams.

One example of a state-of-the-art Text-to-SQL solution in the WikiSQL challenge is the Seq2SQL model, which uses a sequence-to-sequence learning framework to map natural language input to SQL queries. The model uses an attention mechanism to align the input and output sequences and a pointer network to handle SQL queries with complex structural dependencies.

Another example is the Spider model, which uses a combination of recurrent and convolutional neural networks to learn the mapping between natural language and SQL queries. The model uses a hierarchical structure to process the natural language input, with separate modules for understanding the query's intent, columns, and constraints. One difference between these research approaches is the specific deep learning architecture used. The Seq2SQL model uses a sequence-to-sequence framework, while the Spider model uses a combination of RNNs and convolutional neural networks. Additionally, the Spider model uses a hierarchical structure to process the natural language input, while the Seq2SQL model processes the input linearly.

Another difference is in the evaluation metrics used. The Seq2SQL model is evaluated using the execution accuracy of the generated SQL queries, while the Spider model is evaluated using a combination of execution accuracy and natural language understanding metrics. Overall, both the Seq2SQL and Spider models are state-of-the-art Text-to-SQL solutions that have achieved high performance in the WikiSQL challenge. However, their specific architectures and evaluation metrics differ, which can affect their performance and accuracy on different tasks.

4.0.7 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.

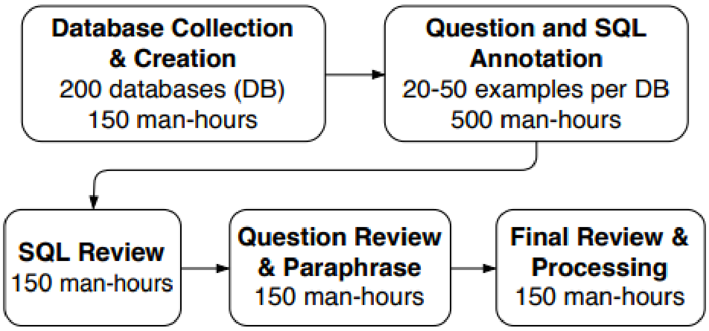


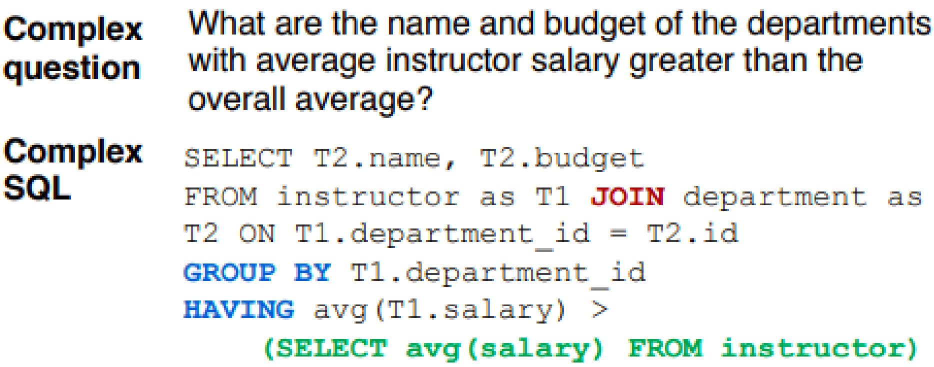
Figure 5: Example from Spider dataset[YZY+]

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.

4.0.8 Spider Challenge

The SPIDER challenge is a research competition dedicated to developing cutting-edge Text-to-SQL solutions. In this challenge, participants strive to develop algorithms that can automatically generate structured SQL queries from natural language input, to improve the performance and accuracy of Text-to-SQL models.



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

In the SPIDER challenge, numerous state-of-the-art Text-to-SQL solutions have been proposed, such as the Spider model. This model uses a combination of recurrent and convolutional neural networks to learn the mapping between natural language and SQL queries. This model also has a hierarchical structure, which allows it to process the natural language input more effectively, thereby allowing it to handle complex queries and variations in language with greater precision and accuracy. This model successfully generates accurate and efficient SQL queries from natural language inputs.

One difference between the SPIDER and WikiSQL challenges is the specific dataset that is used for evaluation. The SPIDER challenge uses a dataset of complex SQL queries and natural language questions derived from real-world databases, while the WikiSQL challenge uses a dataset of more straightforward SQL queries and natural language questions derived from Wikipedia articles. This difference in the dataset can affect the performance and accuracy of the models on the different tasks.

Another difference is in the evaluation metrics used. The SPIDER challenge evaluates the models using execution accuracy and natural language understanding metrics, while the WikiSQL challenge evaluates the models using only execution accuracy. This difference in the evaluation metrics can affect how the models are trained and their performance on the tasks.

5 State-of-the-art Text-To-SQL solutions

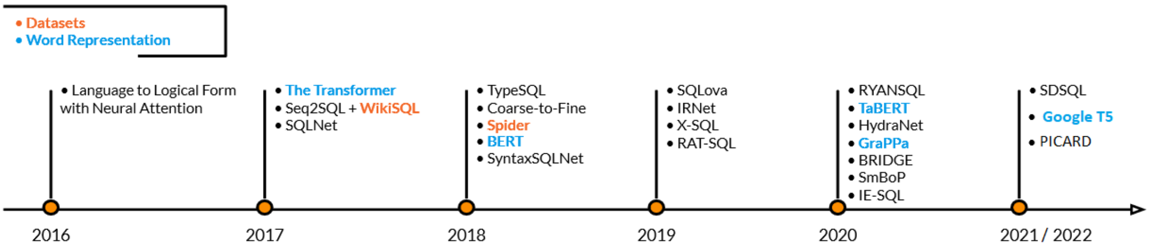


Figure 7: An overview of the deep learning process for Text-to-SQL.

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.

5.1 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.

5.2 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.

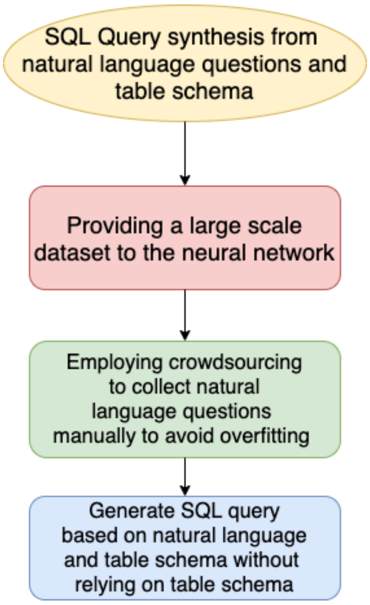
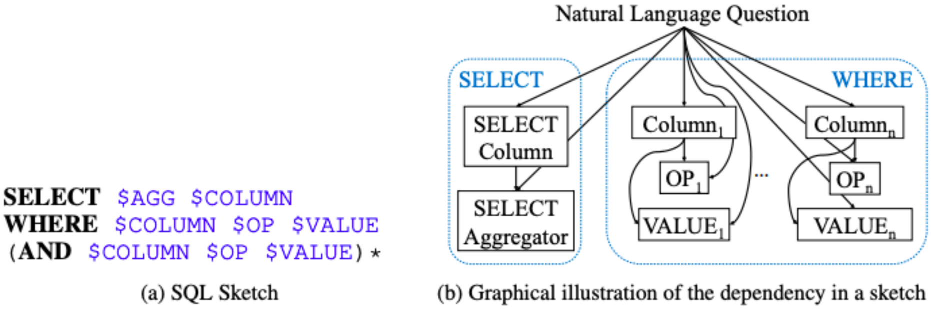

SQL Query synthesis from natural language questions and table schema
Providing a large scale dataset to the neural network
Employing crowdsourcing to collect natural language questions manually to avoid overfitting
Generate SQL query based on natural language and table schema without relying on table schema

Figure 8: 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.



```
SELECT $AGG $COLUMN
WHERE $COLUMN $OP $VALUE
(AND $COLUMN $OP $VALUE)*
```

(a) SQL Sketch

(b) Graphical illustration of the dependency in a sketch

Figure 9: Sketch-based query synthesis

Column attention for sequence-to-set prediction

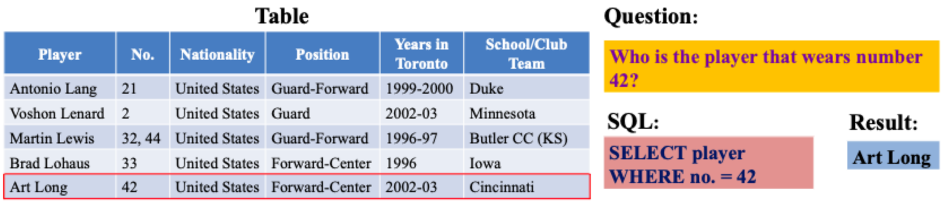
Instead of producing a sequence of column names, sequence-to-set prediction predicts the names of the columns of interest. Based on column names, column attention is part of the generic attention mechanism for computing the feature attention map on a question.

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%.



Table

Player	No.	Nationality	Position	Years in Toronto	School/Club Team
Antonio Lang	21	United States	Guard-Forward	1999-2000	Duke
Voshon Lenard	2	United States	Guard	2002-03	Minnesota
Martin Lewis	32, 44	United States	Guard-Forward	1996-97	Butler CC (KS)
Brad Lohaus	33	United States	Forward-Center	1996	Iowa
Art Long	42	United States	Forward-Center	2002-03	Cincinnati

Question:

Who is the player that wears number 42?

SQL:

SELECT player
WHERE no. = 42

Result:

Art Long

Figure 10: An example of a query executed by SQLNet on WikiSQL

5.3 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.

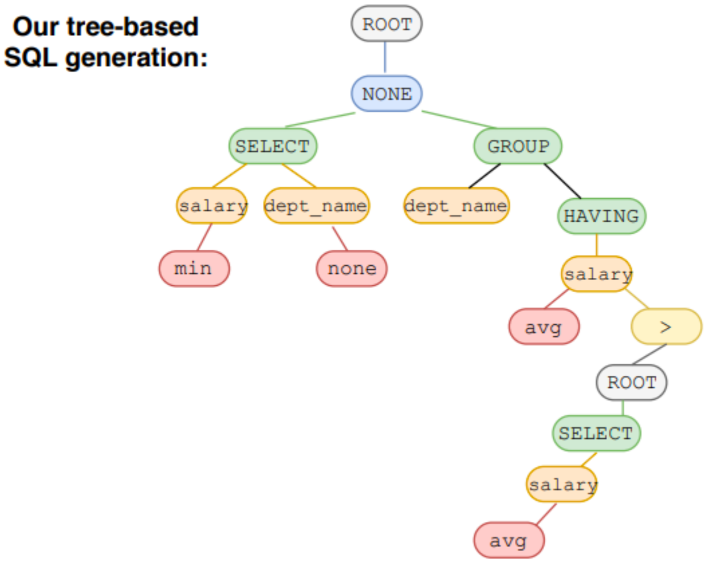


Figure 11: Tree-based SQL generator in SyntaxSQLNet

- 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.

- 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.

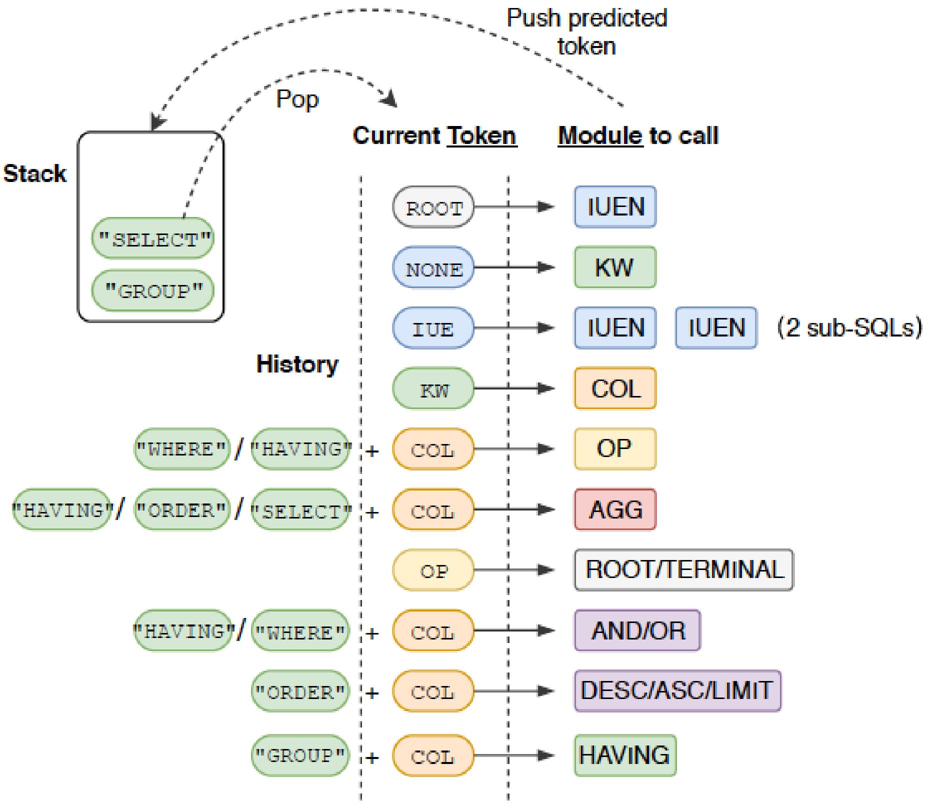


Figure 12: Modules defined in SyntaxSQLNet model

- 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%.

5.4 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.

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.

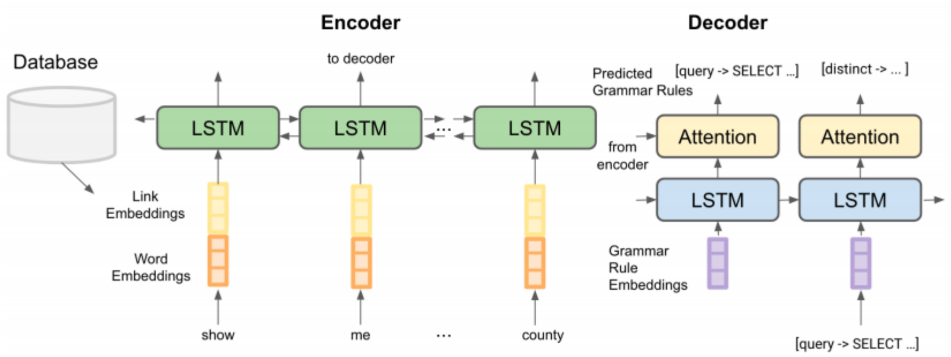


Figure 13: Structure of the proposed model

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

5.5 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.

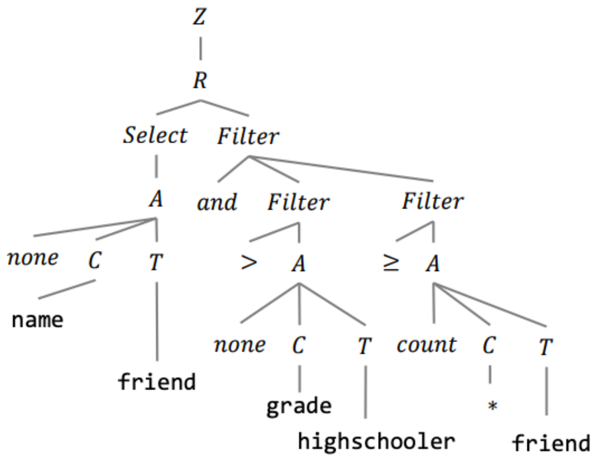


Figure 14: An illustrative example of SemSQL from [GZG$^+$19]

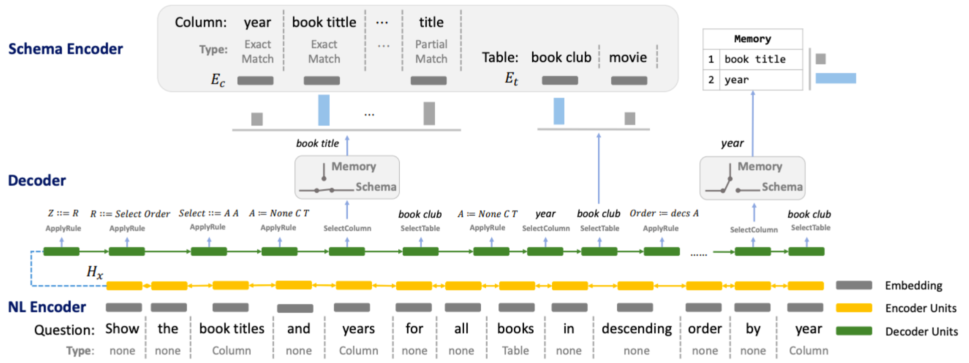


Figure 15: An overview of the neural model proposed in [GZG$^+$19]

- 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.

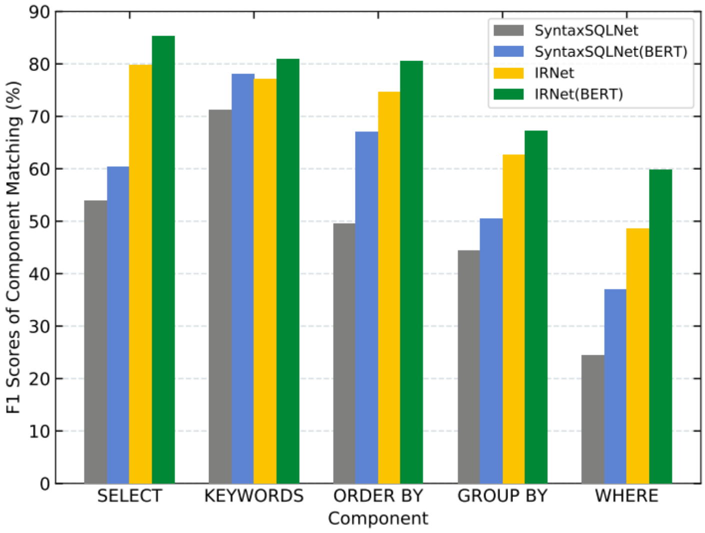


Figure 16: F1 scores of component matching of SyntaxSQLNet, SyntaxSQLNet$BERT$, IRNet and IRNet$BERT$ on the test set from [GZG+19]

5.6 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.

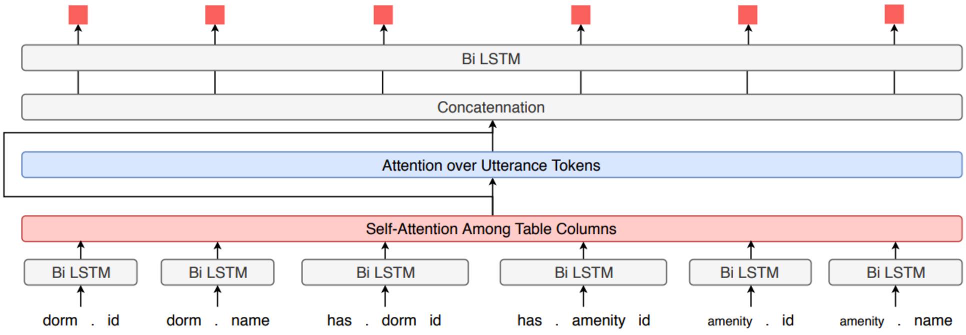
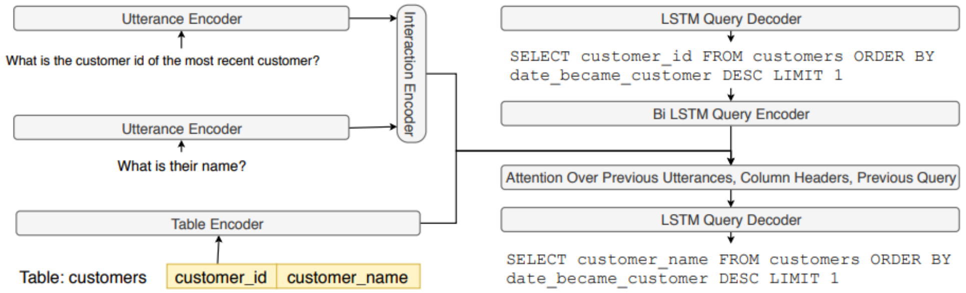


Figure 17: The model architecture of EditSQL from [ZYE+19]



```
SELECT customer_id FROM customers ORDER BY
date_became_customer DESC LIMIT 1
```

```
SELECT customer_name FROM customers ORDER BY
date_became_customer DESC LIMIT 1
```

Figure 18: An example of user utterance and column headers and Utterance Encoder from [ZYE+19]

- 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.

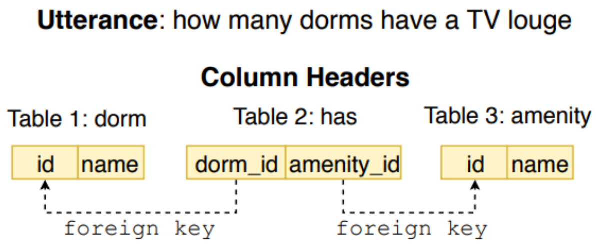


Figure 19: Table Encoder from [ZYE+19]

- 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.

5.7 RAT-SQL

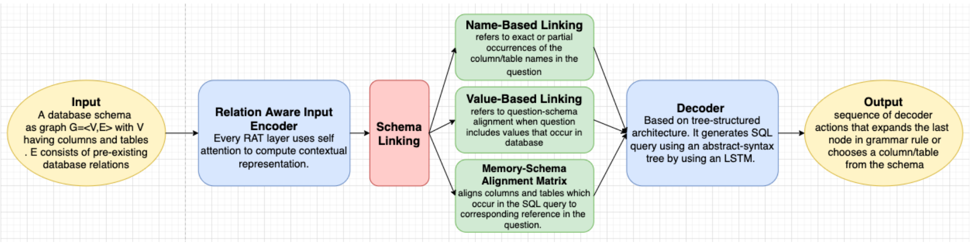
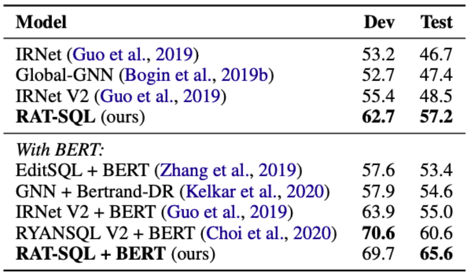

Input
A database schema as graph G=<V,E> with V having columns and tables . E consists of pre-existing database relations
Relation Aware Input Encoder
Every RAT layer uses self attention to compute contextual representation.
Schema Linking
Name-Based Linking
refers to exact or partial occurrences of the column/table names in the question
Value-Based Linking
refers to question-schema alignment when question includes values that occur in database
Memory-Schema Alignment Matrix
aligns columns and tables which occur in the SQL query to corresponding reference in the question.
Decoder
Based on tree-structured architecture. It generates SQL query using an abstract-syntax tree by using an LSTM.
Output
sequence of decoder actions that expands the last node in grammar rule or chooses a column/table from the schema

Figure 20: 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.



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**

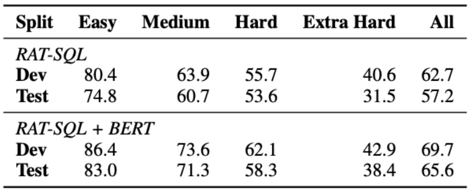
Figure 21: Accuracy on the Spider development and test sets, compared to the other approaches at the top of the dataset leaderboard as of May 1st, 2020 from [WSL$^+$]

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.

5.8 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



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

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.

5.9 PICARD - Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models

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.

PICARD allows for the generation of executable code by constraining the output of the language model to be syntactically and semantically correct.

It does this by integrating with standard beam search, a technique used in natural language processing to generate a sequence of words or tokens by expanding a beam of hypotheses step by step. At each decoding step, PICARD checks whether the most likely tokens are valid and if not, it discards them. PICARD is compatible with any model that generates a sequence of tokens and can be used with character, subword, and word-level language models without requiring exceptional recovery.

It effectively improves the performance of existing models and achieves state-of-the-art performance on tasks such as text-to-SQL translation. Warps model prediction scores and integrates trivially with existing greedy and beam search algorithms used in auto-regressive decoding from language models.

At each generation step, Picard first restricts prediction to the top-k highest probability tokens and then assigns a score of negative infinity to those that fail Picard’s numerous checks.

Four Picard mode settings control their comprehensiveness: off (no checking), lexing, parsing without guards and parsing with guards-the highest mode.

Picard can detect spelling errors in keywords or reject table and column names that are invalid for the given SQL schema. ”Out-of-distribution compositional generalization and natural language variation” refers to the ability of a natural language processing (NLP) system to handle novel combinations of words and phrases that it has not seen before while also being able to handle variations in language usage. Compositional generalization refers to the ability of an NLP system to understand and generate novel combinations of words and phrases by using its knowledge of the meanings and relationships of individual words and phrases. This is an essential aspect of NLP because it allows the system to understand and generate language flexibly and adaptively.

Natural language variation refers to the fact that there are many different ways that people can express the same ideas or concepts in natural language. This can be due to differences in dialect, style, or tone, and it can make it challenging for an NLP system to understand and generate language accurately.

Together, out-of-distribution compositional generalization and natural language variation represent fundamental challenges in the field of NLP. They require NLP systems to handle a wide range of language input and output in order to be effective.

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.

Motivated by the success of Shaw et al. (2021), who demonstrated that a pre-trained T5-Base or T5-3B model could effectively learn the text-to-SQL task, generalize to never-before-seen databases, and even rival the state-of-the-art methods of Choi et al. (2021) and Wang et al. (2020) without any modifications to the model itself, the researchers opted to use T5 as the baseline for all their experiments. The results from Shaw et al. (2021) suggest that T5-based models had the potential to improve the field of natural language processing significantly. Therefore, the researchers sought to take advantage of the capabilities of T5 in order to gain new insights into how natural language can be effectively utilized to solve complex tasks.

5.9.1 Beam Search

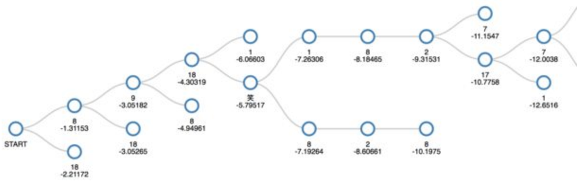


Figure 23: Beam Search

Beam search is a widely used search algorithm in natural language processing and machine learning. It is beneficial in sequence-to-sequence (seq2seq) models, which generate output sequences based on input sequences. Beam search is used to find the most likely sequence of output words given an input sequence.
The basic idea behind beam search is to maintain a set of the most likely sequences at each step of the decoding process. This set of sequences, called the "beam," is initially set to the starting point of the decoding process, and at each step, new sequences are generated by considering all

the following possible words. The new sequences are then ranked based on their likelihood, and the highest-ranking sequences are added to the beam. The process is repeated until a stopping criterion is met [MLGftADNI19]. Beam search is handy in seq2seq models because it allows the model to generate multiple output sequences rather than just a single sequence. This is important because, in many cases, there may be multiple valid outputs for a given input sequence. By generating multiple outputs, beam search allows the model to explore the space of possible outputs and find the most likely sequences.
One of the critical advantages of beam search is that it is computationally efficient. Because it only considers a small number of sequences at each step, it can quickly find the most likely sequences without exploring the entire space of possible outputs. This makes it well-suited for use in applications with limited computational resources, such as on mobile devices or in real-time systems. Another advantage of beam search is that it can be used with other techniques, such as attention mechanisms, to improve the performance of seq2seq models. Attention mechanisms allow the model to focus on specific parts of the input sequence when generating the output, which can help to improve the quality of the generated sequences.
In conclusion, Beam Search is a robust algorithm widely used in natural language processing and machine learning, particularly in the context of sequence-to-sequence (seq2seq) models. It allows the model to generate multiple output sequences rather than just a single sequence and is computationally efficient, making it well-suited for use in applications where computational resources are limited. Additionally, it can be combined with other techniques, such as attention mechanisms, to improve the performance of seq2seq models.

5.9.2 DB Engines

The Picard Method is a method for constrained inference on top of an existing model, but it is not a model itself. Currently, the PICARD parser and the supporting software are not supported for PostgreSQL, MySQL and others, which would require changes to the PICARD parser, translation of Spider databases and text-to-SQL data, and retraining models to produce MSSQL code. To use the Picard Method, a complex toolchain of Haskell code is built with CABAL and requires a complicated toolchain for the Facebook Thrift library.

After the setup, the Picard server can be started by running the compiled standalone executable PICARD. This executable is responsible for providing the necessary information to the user, such as specific parameters and options within the constrained inference. It is important to note that the Picard Method is not a full-fledged model; therefore, it is necessary to combine it with an existing model to get a complete inference system.

The thrift library is used for communication between the parser and the beam search algorithm. The parser, written in the efficient and powerful Haskell programming language, is used in combination with the hf transformers, which is a Python package. To further expand the scope of the system, new SQL engines can be supported by adding a parser for each one.

These parsers also need to be written in Haskell, as the existing SQLite parser is of limited use in this regard, as it has been written to work best on Spider's subset of SQLite and only supports part of the SQLite specification. This means that more advanced parsers must be created to maximize the system's capabilities.

Additionally, these parsers need to be written with a high level of precision in order to ensure that the system can effectively communicate with various engines and databases.

6 Evaluation Metrics

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.

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

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

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

7 Case study

7.1 EZ-PICARD: Microservices-based Architecture

7.2 Results and Use Cases

8 Conclusion

8.1 Future Work

8.2 Summary of findings

8.3 Limitations of the study

References

[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.

[MLGftADNI19] P. J. Moore, T. J. Lyons, J. Gallacher, and for the Alzheimer's Disease Neuroimaging Initiative. Random forest prediction of alzheimer's disease using pairwise selection from time series data. *PLOS ONE*, 14(2):1–14, 02 2019.

[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.

[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.

[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+19] 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.