



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 24, 2023

Dedication

Dedicated to my family and friends for their support and encouragement throughout my academic journey.

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 love 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.

I am immensely grateful to my advisors, 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	5
1.1	Challenges	6
1.2	Thesis Outline	7
2	Technical Background	8
2.1	Early Approaches	8
2.2	Recent Approaches	9
2.3	Terminology	11
2.3.1	SQL (Structured Query Language)	11
2.3.2	Natural Language Processing (NLP)	11
2.3.3	Tokenization	11
2.3.4	Encoder-Decoder Architecture	11
2.3.5	Transformers	11
2.3.6	Self-attention	12
2.3.7	Pre-training and Fine-tuning	12
2.3.8	LSTM (Long Short-Term Memory)	12
2.3.9	BERT (Bidirectional Encoder Representations from Transformers)	12
2.3.10	SQL Constructs	12
2.3.11	SQL Semantic Parsing	13
2.3.12	Evaluation Metrics	13
2.3.13	Baseline Model	13
2.3.14	Incremental decoding	13
2.3.15	Semantic parsing	13
3	Benchmark Dataset	14
3.1	ATIS (Air Travel Information System) Dataset	14
3.2	GeoQuery Dataset	14
3.3	IMDb Dataset	14
3.4	Advising Dataset	15
3.5	WikiSQL Dataset	15
3.6	WikiSQL Challenge	16
3.7	Spider Dataset	16

3.8	Spider Challenge	17
4	State-of-the-art Text-To-SQL solutions	19
4.1	Seq2SQL	20
4.2	SQLNet	20
4.2.1	Sketch-based query synthesis	20
4.3	SyntaxSQLNet	22
4.4	GrammarSQL	24
4.4.1	SQL Grammar	24
4.4.2	Few details on the proposed model	24
4.5	IRNet	25
4.6	EditSQL	26
4.7	RAT-SQL	28
4.8	BRIDGE	29
4.9	PICARD - Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models	29
4.9.1	Beam Search	30
4.9.2	DB Engines	31
5	Evaluation Metrics	33
5.1	Exact String Matching	33
5.2	ESM: Exact Set Matching	33
5.3	Distilled Test Suites	33
6	Experiment and Case study	34
6.1	SEOSS Dataset	34
6.2	SEOSS + T5 PICARD Experiment	34
6.2.1	Metrics and Evaluation Results	34
6.3	EZ-PICARD	34
6.3.1	Microservices Practices	34
7	Conclusion	35
7.1	Exploring Emerging Challenges	35
7.1.1	Conversational Text-to-SQL	35
7.1.2	SpaRC	35

7.1.3	CoSQL Dataset	35
7.2	Summary of findings	35
7.3	Limitations of the study	35
7.4	Conclusion	36
Bibliography		37

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.

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:

- Data analysts can use it to generate SQL queries for specific business questions, such as "What are the top ten products sold this month?"
- 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?"
- Businesses can use this technique to automate data extraction and improve efficiency.
- 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 Text-to-SQL models may provide a partial solution to this complex problem, humans still have challenges to overcome. Even experienced database administrators and developers can need help with the task of dealing with unfamiliar schema when working on database migration projects. This is often due to the fact that they have never seen the schema before and therefore need to learn how to read and interpret it correctly. Furthermore, it can take time to determine how to make the necessary changes in order to migrate the data from one database to another successfully. In spite of these challenges, it is possible to successfully complete a database migration project with the help of a text-to-SQL model, as long as the model is carefully implemented and the proper steps are taken.

This research study will examine the various natural language processing (NLP) technologies that have been utilized in recent years to convert text language into Structured Query Language (SQL). Specifically, it will explore and compare the most commonly used NLP technologies and review their effects on the effectiveness of the conversion process. Moreover, this study will also analyze the representative datasets and evaluation metrics that are utilized in the current solutions for this challenging task. By doing so, it is our hope that this research study will provide valuable insights into how NLP technologies can be effectively and efficiently utilized in the conversion of text language into SQL.

Additionally, we will undertake a comprehensive study of the SEOSS (Software Engineering Dataset for Text-to-SQL and Question Answering Tasks) dataset from our esteemed researchers at the university. We will then evaluate the execution of this dataset using the most advanced Text-to-SQL model currently available. This will enable us to understand the capabilities of the SEOSS dataset better and help us to make informed decisions.

1.1 Challenges

Text-to-SQL is an intricate task, given the complexity and diversity of natural language and the structure and regulations of SQL. One of the most challenging aspects is to decipher the intent and significance of the natural language input, as it can be ambiguous or have varied interpretations. This can result in mistakes when building the corresponding SQL query, like selecting the incorrect table or columns or not recognizing the conditions for filtering or sorting the data. Additionally, the natural language input may contain typos or unknown words, which can complicate the mapping process. Moreover, the query generated may not be in the optimal form, as it has to take into account the various data types, operations, and constraints of the underlying database. Therefore, it is crucial to develop models and algorithms that can accurately map natural language to SQL queries.

Another challenge is dealing with the diverse and dynamic nature of databases, as the schema and data may change over time, and there may be variations in naming conventions and conventions across different databases. This can make it difficult for the model to correctly map the natural language input to the appropriate SQL elements, such as table and column names, and to handle variations in the structure of the SQL queries generated. Additionally, many real-world scenarios require integration with external knowledge bases and ontologies, which can be challenging to handle, especially when the external knowledge needs to be completed or consistent. Furthermore, the system must be robust to different types of user input, such as colloquial or informal language or input that needs to be completed or clarified. Additionally, Text-to-SQL systems must be able to handle errors in the input, such as typos, as well as rare edge cases that may not have been encountered during the training process. Finally, Text-to-SQL systems must be robust to the presence of out-of-vocabulary words and rare edge cases, which can be challenging to handle without significant amounts of labeled data, as well as the need to make accurate predictions with limited training data.

1.2 Thesis Outline

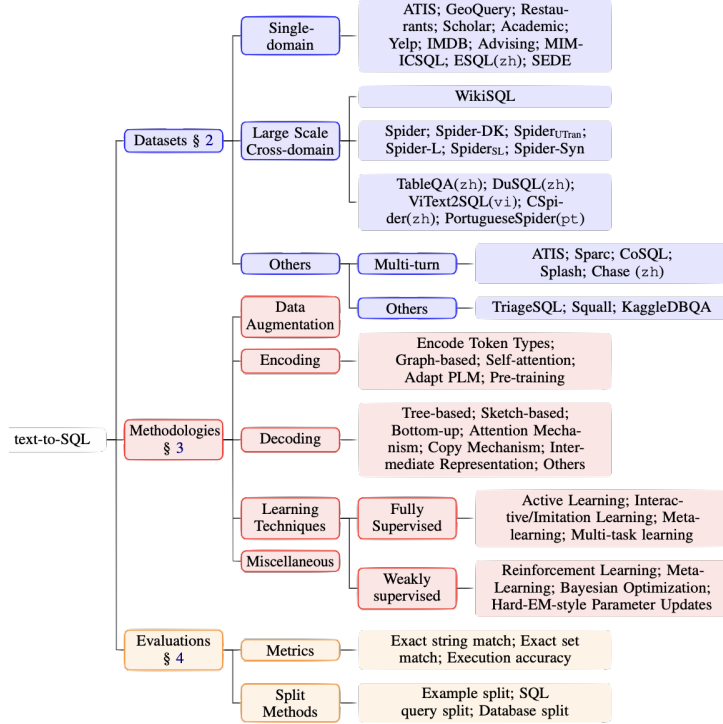


Figure 1: Mindmap of the state-of-the-art

In this section, we provide an outline of our thesis.

- Chapter 1 of the thesis provides an introduction to the topic of Text-to-SQL and discusses the challenges and contributions of the research.
- Chapter 2 provides a technical background on Text-to-SQL, including early approaches, recent approaches, and important terminology such as LSTM, encoder, decoder, transformers, BERT, semantic parsing, baseline model and incremental decoding.
- Chapter 3 describes the benchmark datasets used for evaluating Text-to-SQL methods, including the ATIS, GeoQuery, IMDb, Advising, WikiSQL, and Spider datasets.
- Chapter 4 presents an overview of state-of-the-art Text-to-SQL solutions, including Seq2SQL, SQLNet, SyntaxSQLNet, GrammarSQL, IRNet, EditSQL, RAT-SQL, BRIDGE, and PICARD.
- Chapter 5 explains the evaluation metrics used to assess the performance of Text-to-SQL systems, including exact string matching, exact set matching, and distilled test suites.
- Chapter 6 presents an experiment and case study using the SEOSS dataset and the T5 PICARD model, with a focus on the metrics and evaluation results. The thesis also includes a section on EZ-PICARD and its Microservices practices.
- Chapter 7 concludes the thesis by summarizing the findings and exploring emerging challenges such as conversational Text-to-SQL.

2 Technical Background

In this chapter, we provide background information about the technical concepts related to the main topics of this thesis, which focus on natural language understanding and text generation. We focus on early and recent approaches and the terminology needed to understand the basics of this thesis.

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.

Early approaches to Text-to-SQL relied on rule-based and template-based methods, while recent approaches use neural networks and machine learning techniques. This allows them to handle a wide range of natural language inputs and generate more accurate SQL queries, which we will discuss further.

2.1 Early Approaches

Early approaches to Text-to-SQL focused on rule-based methods and template-based methods. These approaches relied on predefined templates and a set of predefined rules to generate SQL queries. These methods were based on the idea that a fixed set of templates and rules could be used to generate SQL queries for a wide range of natural language inputs. However, these methods were limited by their reliance on predefined templates and were not able to handle a wide range of natural language inputs.

In the case of rule-based methods, a set of predefined rules were used to map the natural language input to the corresponding SQL query. These rules were based on predefined grammar and were used to identify the SQL constructs present in the input text. These methods were able to generate simple SQL queries, but they were not able to handle more complex queries or handle variations in natural language inputs.

Template-based methods, on the other hand, relied on predefined templates to generate SQL queries. These templates were based on a predefined set of SQL constructs and were used to map the natural language input to the corresponding SQL query. These methods were able to handle a limited set of natural language inputs, but they were not able to handle variations in the input or generate more complex queries.

Early research in Text-to-SQL includes work by researchers such as Warren and Pereira in 1994, who proposed a rule-based method for generating SQL queries from natural language text. Their system used a set of predefined rules to map natural language constructs to SQL constructs and was able to generate simple SQL queries. Another example of a rule-based method is the work by Zelle and Mooney in 1996, who proposed a system that used a predefined grammar to identify the SQL constructs present in the input text and generate the corresponding SQL query.

In the case of template-based methods, one example is the work by Chen and Popescu in 1996, who proposed a template-based method for generating SQL queries. Their system used

predefined templates to map natural language inputs to SQL queries and was able to handle a limited set of natural language inputs. Other early template-based techniques, such as the work of Rau and Wiederhold in 1995, used predefined templates to generate SQL queries. The system uses a set of predefined templates, where each template represents a specific SQL construct, and the system tries to match the natural language input to the suitable template to generate the corresponding SQL query.

In summary, early approaches to Text-to-SQL were limited by their reliance on predefined templates and rules, which made them unable to handle a wide range of natural language inputs and generate complex SQL queries.

2.2 Recent Approaches

Recent approaches to Text-to-SQL have focused on using neural networks and machine learning techniques to generate SQL queries. These methods use large amounts of training data to learn the relationship between natural language and SQL and can generate SQL queries for a wide range of inputs. These methods are able to handle a wide range of natural language inputs and are not limited by predefined templates or rules. Additionally, recent approaches leverage pre-trained models such as BERT, GPT-2, and T5, which have been pre-trained on a large corpus of text, to fine-tune text-to-SQL tasks, which enables them to understand the natural language inputs better and generate more accurate SQL queries.

One popular approach is the use of encoder-decoder architecture, which uses an encoder to encode the natural language input and a decoder to generate the corresponding SQL query. The encoder is a pre-trained language model such as BERT, which is fine-tuned on the task of text-to-SQL, and the decoder is a neural network that generates the SQL query. This architecture has been shown to be effective in generating accurate SQL queries for a wide range of natural language inputs.

Another recent approach is the use of reinforcement learning to generate SQL queries, where a neural network generates a sequence of SQL tokens and is trained using a reward signal based on the quality of the generated query. This approach has been shown to be adequate in generating more complex SQL queries and handling variations in natural language inputs.

In recent years, the Transformer architecture has had a significant impact on natural language processing and machine learning, including in the field of Text-to-SQL. The Transformer architecture, presented in the paper "Attention Is All You Need" by Vaswani et al. in 2017, is a neural network architecture that uses self-attention mechanisms to process sequences of data, such as natural language text.

One of the key advantages of the Transformer architecture is its ability to handle long-term dependencies in sequences of data, making it well-suited for tasks such as natural language understanding and text generation. This has led to the development of pre-trained Transformer models, such as BERT, GPT-2, and T5, that have been trained on a large corpus of text and can be fine-tuned on specific tasks such as Text-to-SQL.

The use of pre-trained Transformer models such as BERT in Text-to-SQL has shown to be effective in improving the performance of the models. The pre-trained models have a good understanding of the natural language, which enables them to understand the input text better and generate more accurate SQL queries. The Transformer architecture and pre-trained models

such as BERT have had a significant impact on recent studies in the field of Text-to-SQL. The ability of the Transformer architecture to handle long-term dependencies in sequences of data and the pre-trained models' good understanding of natural language has made it possible to generate more accurate SQL queries for a wide range of natural language inputs.

In summary, recent approaches to Text-to-SQL leverage neural networks and machine learning techniques, such as the use of encoder-decoder architecture and reinforcement learning. These approaches use large amounts of training data and pre-trained models such as BERT to generate accurate SQL queries for a wide range of natural language inputs.

2.3 Terminology

Here is an updated list of key terminology and vocabulary that you may need to know before studying Text-to-SQL language models:

2.3.1 SQL (Structured Query Language)

The standard, widely used programming language designed to manage relational databases, which enables users to store, retrieve and manipulate data.

2.3.2 Natural Language Processing (NLP)

The field of study focused on the interaction between human language and computers, which ranges from understanding spoken language to generating natural language text.

2.3.3 Tokenization

The process of breaking up a sentence into individual words or phrases, which is necessary for tasks such as machine translation and text summarization.

2.3.4 Encoder-Decoder Architecture

A powerful neural network architecture that utilizes an encoder to transform the input data into a compact and meaningful representation and a decoder to generate the desired output from that representation. This architecture has been widely used in many applications such as language translation, image captioning, and text summarization to produce high-quality results. Furthermore, the encoder-decoder architecture has the advantage of being able to learn complex relationships between input and output, making it a suitable tool for many challenging tasks.

2.3.5 Transformers

The architecture introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017, known as Transformers, is a revolutionary breakthrough in the way sequences of data are processed. By utilizing self-attention mechanisms, the model is able to achieve improved efficiency and accuracy, while also being much simpler to implement and deploy. This makes it particularly appealing for a wide range of applications, from natural language processing to computer vision. Furthermore, due to its scalability, Transformers are able to accommodate large data sets, enabling it to be used to tackle more complex tasks. As such, Transformers are becoming increasingly popular in the field of machine learning and artificial intelligence, with more and more research being done to further explore its capabilities.

2.3.6 Self-attention

Self-attention is a mechanism used in the transformer architecture that enables the model to identify the significance of different elements of the input sequence, so as to be able to generate an output that is more accurate and effective. This mechanism allows the model to take into account the relationships between different parts of the input sequence and to factor those relationships into its output. Additionally, self-attention allows the model to capture patterns from the input sequence and to use those patterns to generate a more meaningful output. It is this combination of factors that make self-attention such an important tool for deep learning models.

2.3.7 Pre-training and Fine-tuning

Pre-training refers to training a model on a large dataset, and then fine-tuning it on a smaller dataset for a specific task, which helps to improve the model's performance on the specific task.

2.3.8 LSTM (Long Short-Term Memory)

A type of recurrent neural network that has been designed to store information over a longer period of time than traditional neural networks, allowing it to better capture long-term dependencies. This makes it especially well-suited for tasks such as language modeling and text generation, where it can take into account the context of the text in order to generate more accurate outputs. In addition, LSTM networks are able to more effectively identify patterns in the data that would be difficult for traditional networks to capture. This makes them ideal for tasks such as sequence prediction and classification, where they can identify patterns that would otherwise be too subtle for traditional networks to detect.

2.3.9 BERT (Bidirectional Encoder Representations from Transformers)

A pre-trained Transformer model that has been trained on a large corpus of text, with the primary aim of pre-training language representations for use in natural language processing tasks. This pre-training helps to give BERT a strong understanding of the language structure and helps in faster training times for downstream tasks. BERT can be fine-tuned for various applications, such as Text-to-SQL, where it can provide better performance than non-specialized models. By leveraging the already learned representations from the pre-trained model, BERT is able to quickly adjust to the task at hand, resulting in faster training times.

2.3.10 SQL Constructs

The elements of SQL language such as SELECT, FROM, WHERE, JOIN, which are used to build queries and retrieve data from a database.

2.3.11 SQL Semantic Parsing

The task of understanding the meaning of natural language text and translating it into a semantically meaningful SQL query, which can then be used to retrieve data from a database.

2.3.12 Evaluation Metrics

Measures used to evaluate the performance of Text-to-SQL models, such as accuracy, F1-score, and BLEU score, which are used to compare different models and determine the best performing model.

2.3.13 Baseline Model

A model that serves as a reference point or starting point for comparison, providing a baseline for performance against which other models can be evaluated.

2.3.14 Incremental decoding

A decoding strategy where the model generates a sequence of tokens one at a time, at each step conditioned on the previous tokens, the input, and the context of the sentence. This approach allows for more dynamic and flexible generation of output, as it takes into account a variety of factors when making decisions about the next token. This strategy also helps the model to avoid repeating itself, providing more diverse and unique outputs. Furthermore, incremental decoding helps the model to better capture the nuance of the language as it is able to build upon previous decisions and refine its output as it progresses.

2.3.15 Semantic parsing

Semantic parsing is an area of natural language processing that involves extracting the meaning or intent from text. One type of Semantic Parsing, Text-to-SQL, involves the conversion of natural language problems into SQL query statements. This is a challenging task, one that requires the use of advanced machine learning and natural language processing algorithms. As such, the research conducted in this field seeks to explore the various solutions and practices that have been employed by researchers in order to effectively tackle this problem. Furthermore, it is also important to note that this problem is not just limited to the conversion of natural language into SQL query statements, as there are other applications of Semantic Parsing that have been explored, such as Natural Language Generation (NLG). Overall, by understanding the various techniques used for Semantic Parsing, we can gain a better understanding of the complexities involved in this task and how best to approach it.

3 Benchmark Dataset

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.

3.1 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.

3.2 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.

3.3 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.

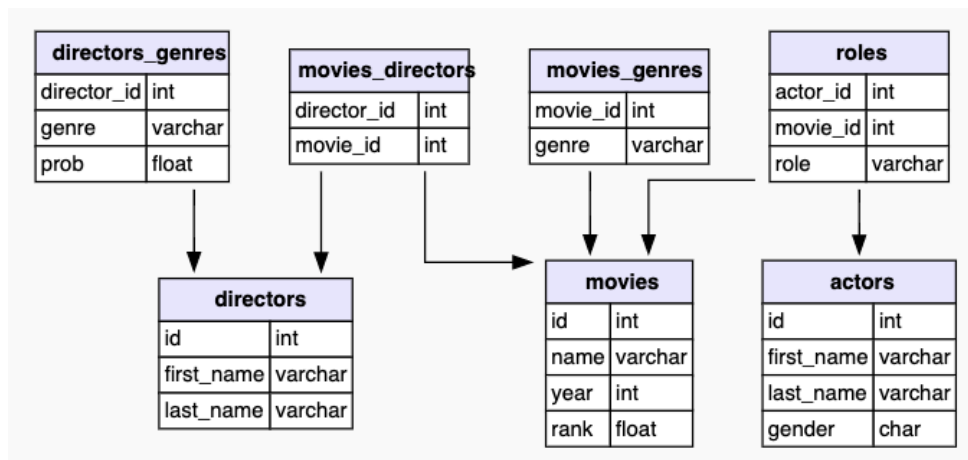


Figure 2: Database Structure of IMDb dataset

3.4 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.

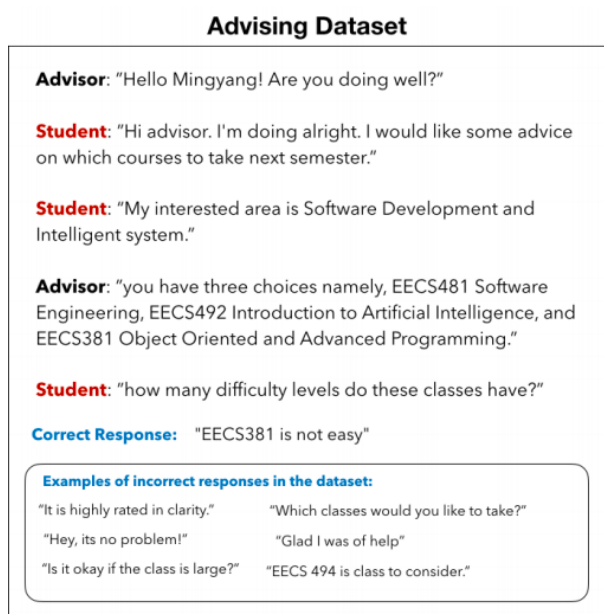


Figure 3: Example from Advising dataset [VR19]

3.5 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[?]

3.6 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.

3.7 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.

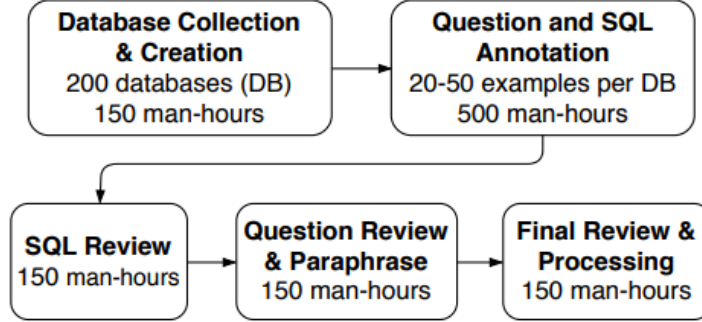


Figure 5: Example from Spider dataset[YZY⁺]

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.

Complex question	What are the name and budget of the departments with average instructor salary greater than the overall average?
Complex SQL	<pre> 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) </pre>

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

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.

3.8 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.

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.

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

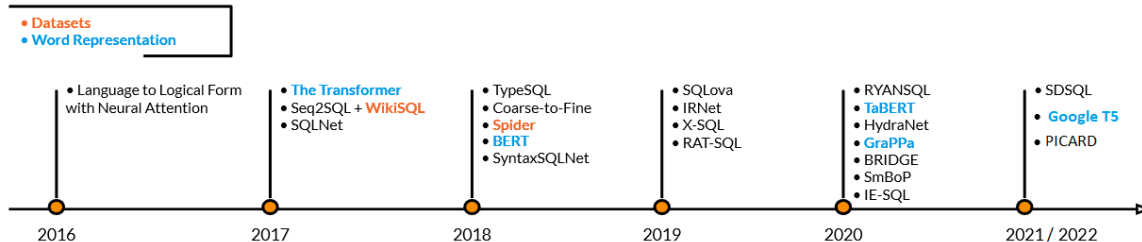


Figure 7: An overview of the deep learning process for Text-to-SQL.

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 challenge and we will continue with SQLova[?] SQLNet[XLS] and introduction 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.

4.1 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.

4.2 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.

4.2.1 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.

Column attention for sequence-to-set prediction

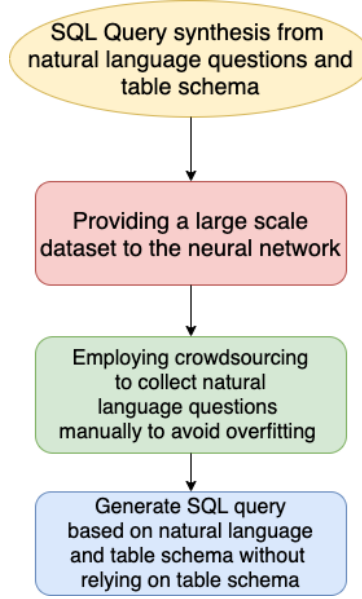


Figure 8: SQLNet

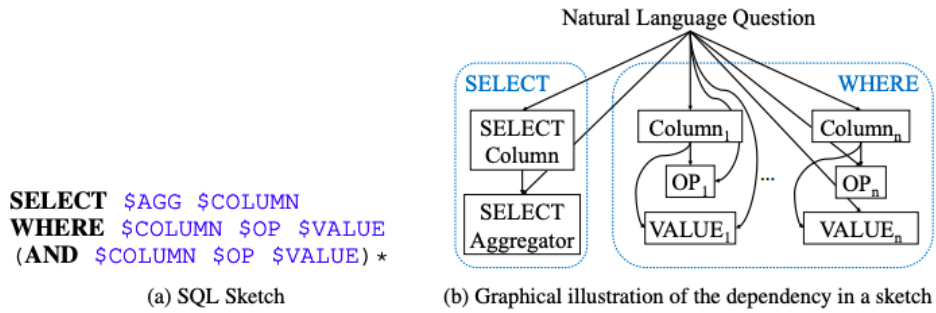


Figure 9: Sketch-based query synthesis

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						Question:	
Player	No.	Nationality	Position	Years in Toronto	School/Club Team	Who is the player that wears number 42?	
Antonio Lang	21	United States	Guard-Forward	1999-2000	Duke	SQL: SELECT player WHERE no. = 42	Result: Art Long
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		

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

4.3 SyntaxSQLNet

The main goal of developing the SyntaxSQLNet model was to generate complex SQL queries with multiple clauses and generalize them to new databases. This was achieved through the use of a syntax tree network, which is capable of addressing complex and cross-domain queries.

As is evident in the chart below, the SyntaxSQLNet model is composed of several components, each with its unique function and purpose in generating complex SQL queries. The encoders are table-aware, while 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 was employed to improve accuracy further to generate more variance during training, allowing for a greater degree of accuracy and robustness.

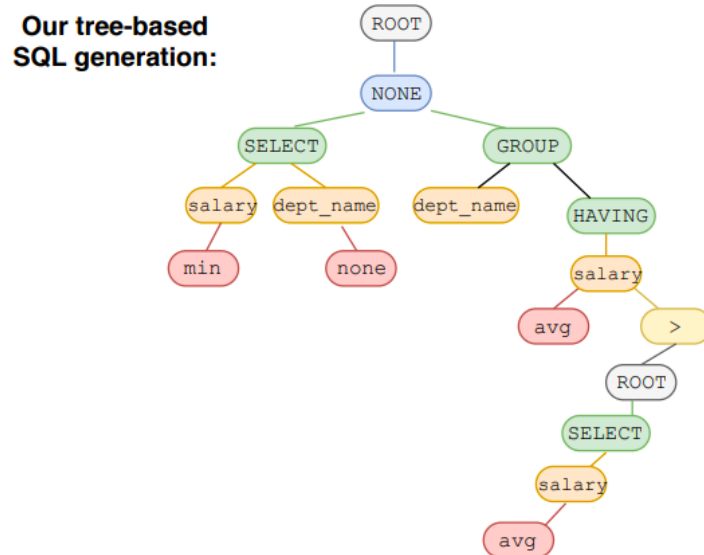


Figure 11: Tree-based SQL generator in SyntaxSQLNet

SQL Grammar and Attention Mechanism

In order to enable the decoder to handle complex queries, SQL grammar is used. This allows the decoder to make decisions at each step of recursive decoding, determining which module to invoke.

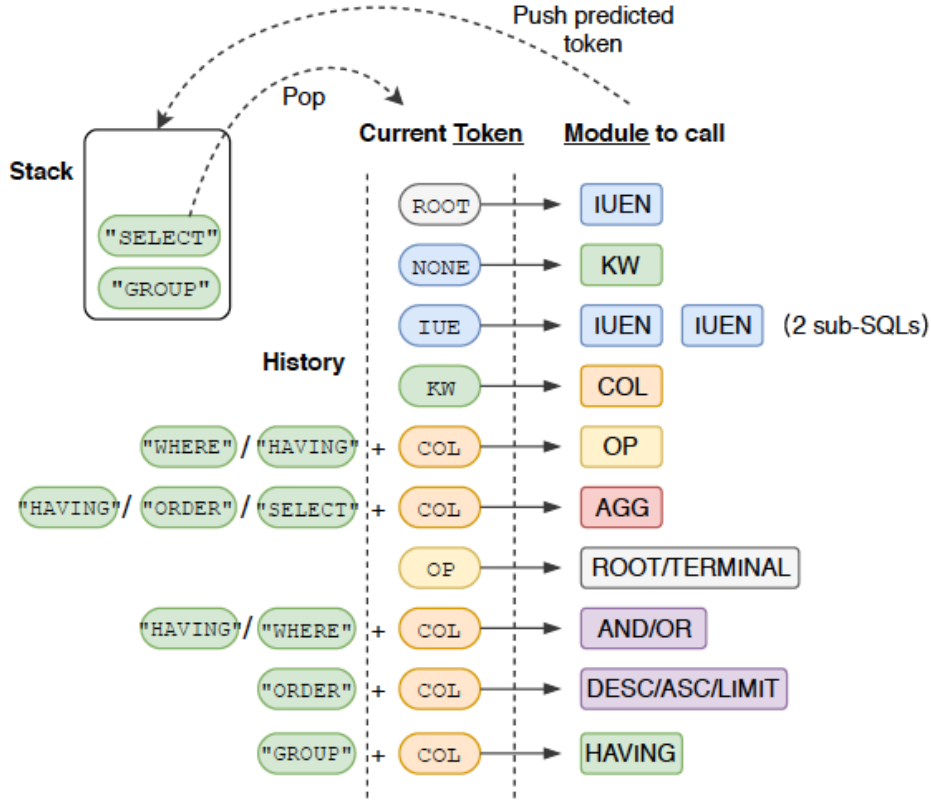


Figure 12: Modules defined in SyntaxSQLNet model

Furthermore, the prediction of the following SQL token is based on the history of SQL path generation and the current SQL tokens.

Additionally, the attention mechanism encodes the question representation and applies it to the SQL path history encoding.

This is beneficial as the history of SQL path generation can be used to determine which token should be used next. Thus, the attention mechanism helps to create a better representation of the query, allowing for an efficient and accurate response.

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%.

4.4 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.

4.4.1 SQL Grammar

The shallow parsing expression grammar (PEG) aims to capture as little SQL as possible to cover most instances in the dataset. This makes it a versatile tool for handling a wide variety of SQL queries. To further ensure consistency of table, column, and value references in SQL, the authors added non-terminals to the context-free grammar to use these references for cross-table joins.

At runtime, constraints are used to guarantee that only valid programs can be used to join different tables in a database together using a foreign key. This ensures that the data stored in the database is kept consistent and that the integrity of the database is maintained.

4.4.2 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.

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

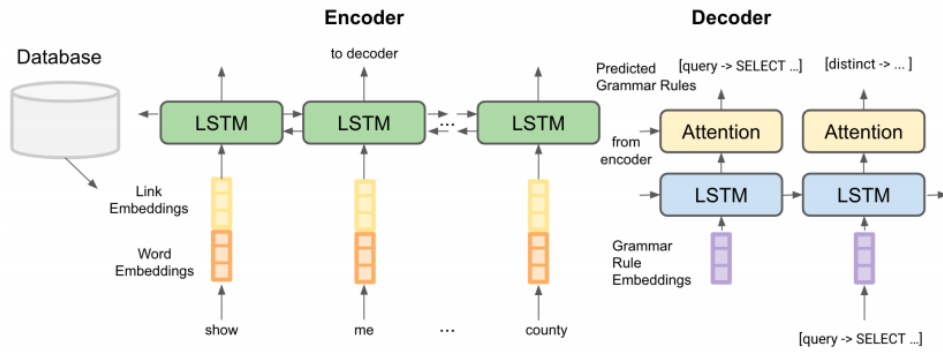


Figure 13: Structure of the proposed model

4.5 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.

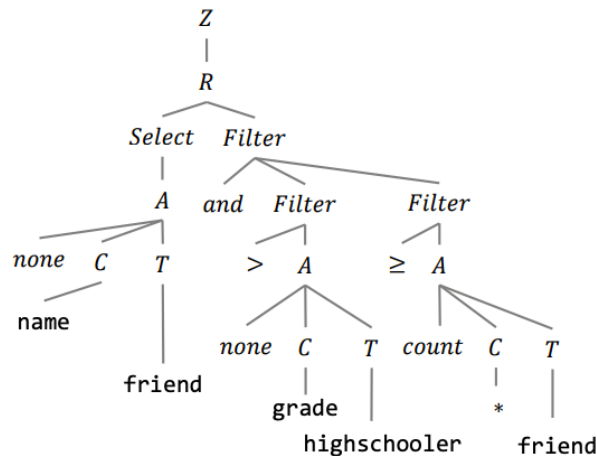


Figure 14: An illustrative example of SemSQL from [GZG⁺19]

- The model provides different functions to accomplish Text2SQL tasks.

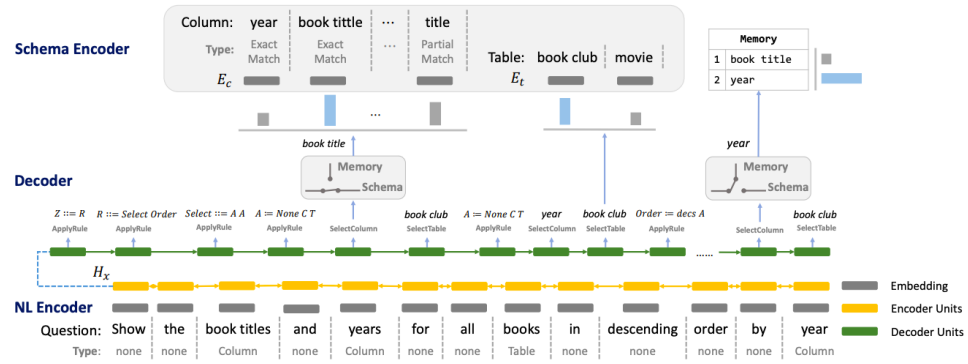


Figure 15: An overview of the neural model proposed in [GZG+19]

- 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.

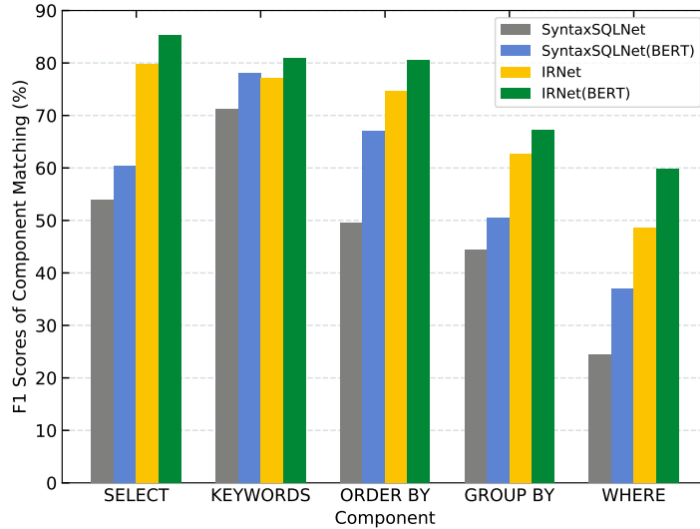


Figure 16: F1 scores of component matching of SyntaxSQLNet, SyntaxSQLNetBERT, IRNet and IRNetBERT on the test set from [GZG+19]

4.6 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.

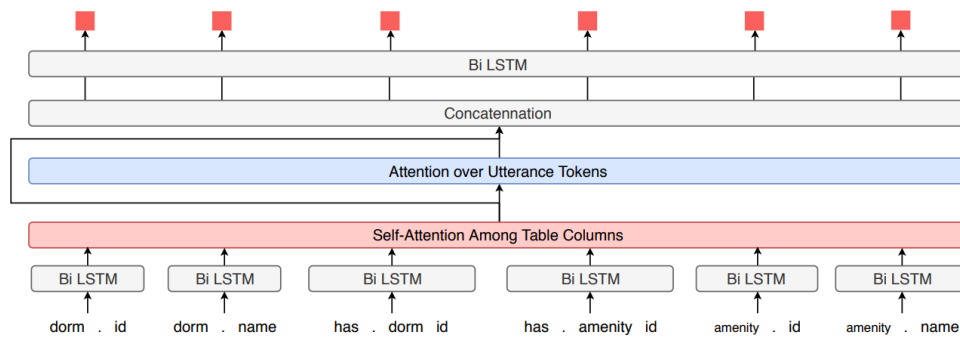


Figure 17: The model architecture of EditSQL from [ZYE⁺19]

- 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.

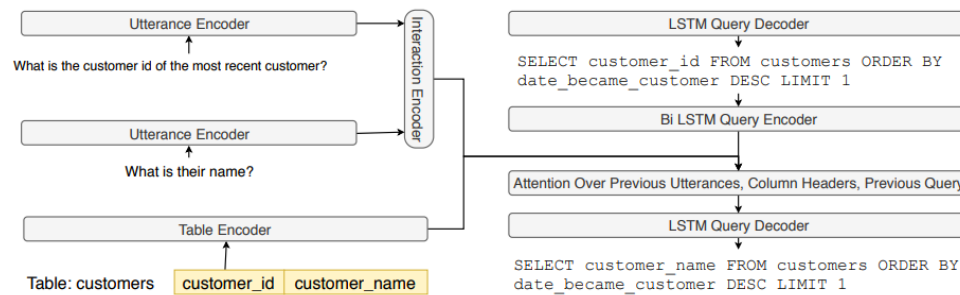


Figure 18: An example of user utterance and column headers and Utterance Encoder from [ZYE⁺19]

- 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.

Utterance: how many dorms have a TV lounge

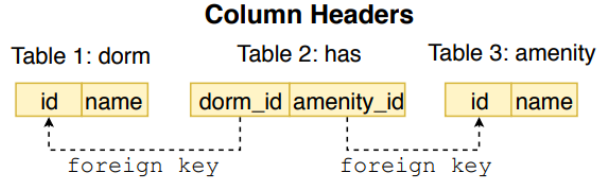


Figure 19: Table Encoder from [ZYE⁺19]

- 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.

4.7 RAT-SQL

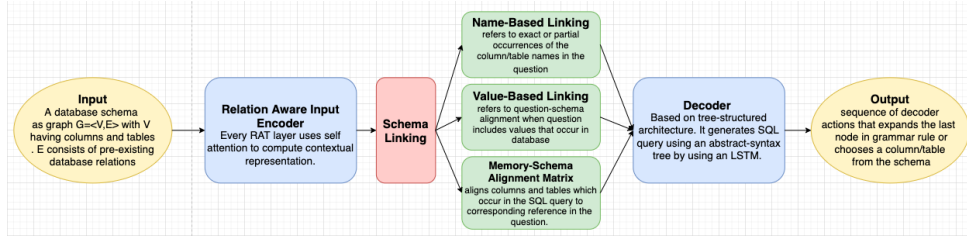


Figure 20: A flow chart of RAT-SQL model

- 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.

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
<i>With BERT:</i>		
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⁺]

Split	Easy	Medium	Hard	Extra Hard	All
<i>RAT-SQL</i>					
Dev	80.4	63.9	55.7	40.6	62.7
Test	74.8	60.7	53.6	31.5	57.2
<i>RAT-SQL + BERT</i>					
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⁺]

4.8 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 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.

4.9 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 decod-

ing 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.

4.9.1 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 se-

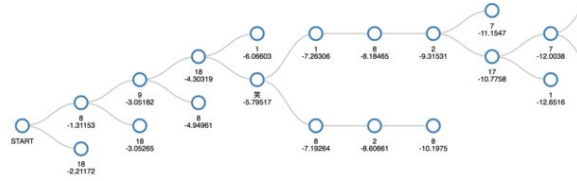


Figure 23: Beam Search

quences 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.

4.9.2 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.

5 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.

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

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.

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

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

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

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

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

5.1 Exact String Matching

5.2 ESM: Exact Set Matching

the current Spider official metric

5.3 Distilled Test Suites

6 Experiment and Case study

6.1 SEOSS Dataset

6.2 SEOSS + T5 PICARD Experiment

6.2.1 Metrics and Evaluation Results

6.3 EZ-PICARD

6.3.1 Microservices Practices

7 Conclusion

In this thesis, we studied cross-view learning with limited supervision. We provided both empirical and theoretical analyses that how we can leverage the information across different data views to learn good representations when having access to only limited supervision signals (e.g., without supervised labels or with only auxiliary information of data). This chapter provides a succinct summary of the main contributions, and then we discuss certain limitations of our work. Studying these limitations helps us better understand the topic - cross-view learning with limited supervision - and hence attempting to address these limitations can be potential future research directions.

7.1 Exploring Emerging Challenges

7.1.1 Conversational Text-to-SQL

7.1.2 SpaRC

7.1.3 CoSQL Dataset

7.2 Summary of findings

This thesis contributes in three folds. Firstly, we presented approaches that take into account the heterogeneous structures across views when modeling multi-view data. In particular, Chapter 3 introduced the Multimodal Transformer that attends to interactions between views across distinct time steps and latently correlates the cross-view signals. Chapter 4 introduced the Factorized Multimodal Model that disentangles multi-view data into multi-view discriminative factors and view-specific generative factors. We showed that, when considering the cross-view heterogeneous structures, we can learn representations that achieve better data generation, discriminative performance (i.e., multi-view prediction), and interpretability (both local and global interpretability) for the model.

7.3 Limitations of the study

A first observation about this thesis is that we considered multi-view data from mostly two or three different views (e.g., the human multi-modal utterance with visual, acoustic, and textual views). While this was an important stepping stone, data can often include a larger number of views, such as signals for aircraft sensors that track oil temperature, fuel pressure, air speed measurement, lightening detection, vibration detection, etc. Some of our proposed multi-view representation learning approaches may have trouble to scale up to larger number of views, and we acknowledge this potential limitation as representation learning with a large number of views. Second, our empirical and theoretical analysis on self-supervised learning lie mainly within visual modality. In particular, we considered augmented variants as different views of an image, and then we analyzed why the self-supervised learned representations can reach good downstream performance and the practical deployments of different self-supervised objectives. Although we manifested good results in visual modality, we have not yet shown that

our approaches can generalize to other modality, such as audio or textual modalities. We identify this limitation as self-supervised learning beyond visual modality. Lastly, when studying multi-view representation learning, the thesis focuses primarily on the task of perception and less about action generation (e.g., action generation for navigation). The perception and action generation procedures are two important phases for an intelligent agent, where the perception phase receives multi-view signals and transfers them into high-level representations, and the action generation phase takes the internal representations and generates actions. Our thesis does not directly tackle the problem of action generation in multi-view representation learning, such as the movement of a robot after receiving the multi-view sensory signals (e.g., visual inputs from camera or distance measurements from ultrasonic sensors). We identify this limitation as multi-view representation learning for action generation. In the following sub-section, we discuss these potential limitations and point towards promising future research directions.

7.4 Conclusion

In this chapter, we studied three potential limitations (representation learning with a large number of views, self-supervised learning beyond visual modality, and multi-view representation learning for action generation) of our thesis work on the topic - cross-view learning with limited supervision. We also discussed the potential future work to address these limitations. Nonetheless, in addition to the studied challenges (heterogeneous structures, relationship quantification, and learning with limited supervision) and the discussed limitations, there still remain lots of challenges and sub-challenges waiting for us to solve. To conclude, we hope that our thesis sheds light on advantages of leveraging information across different data views to learn good representations. We also believe that our work can potentially open up new horizons for learning representations when having access to limited supervision signals

Bibliography

- [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.