Survey Paper: A Technology sharing about Text-to-SQL

CSDS 433 Spring 2023

Ziming Cui (zxc701)

***Abstract* — Text-to-SQL is a natural language processing (NLP) task that involves converting natural language queries into SQL (Structured Query Language) queries. The goal of text-to-SQL is to enable users to interact with databases using natural language queries, rather than having to learn SQL syntax. Despite recent advances in natural language processing and machine learning, challenges remain in encoding the semantics of natural language and translating it into SQL syntax. In this paper, we talk about recent progress in text-to-SQL such as datasets, methods and so on. We highlight the challenges in this task and suggest potential future research directions. Our survey serves as a valuable resource for researchers and can guide future work in addressing the challenges in text-to-SQL.**

Keywords—Database, NLP, Text-to-SQL, Spider, WikiSQL

I. INTRODUCTION

Text-to-SQL is a task that involves converting natural language utterances into SQL queries, which can be used to retrieve information from a database. Just like Figure 1, people ask the question that "What're core cities in the province of Gansu?". To get the SQL feedback and enable non-technical users to query databases using natural language, we need to has practical applications in building natural language interfaces to databases (NLIDBs) Additionally, text-to-SQL is a representative task in semantic parsing, which involves mapping natural language expressions to structured representations. The results of text-to-SQL can be used for downstream tasks such as Healthcare, Finance, Law question answering. Researchers in both the NLP and database communities have been studying this task for many years, and there have been significant advances in the development of text-to-SQL systems using both rule-based and machine learning-based approaches. The framework and structure of text-to-SQL technology.

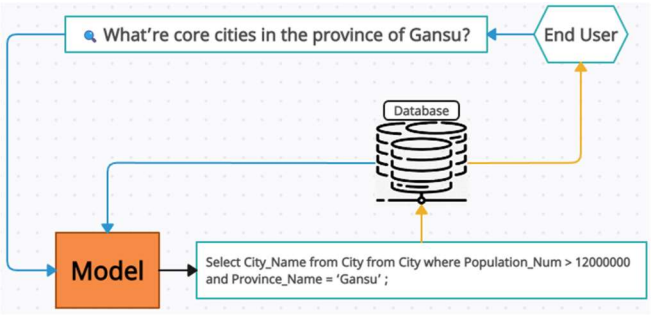

Fig. 1. The framework for text-to-SQL systems

The encoding, translating, and decoding steps are critical components of the text-to-SQL pipeline, and there has been significant work on developing methods to improve each step. Representation learning, intermediate structures, decoding, model structures, and training objectives are all aspects that have been explored in the literature to improve the performance of text-to-SQL systems. Additionally, data resources and evaluation are also critical components of the text-to-SQL research landscape. However, there is a need for more comprehensive surveys of the field, which can provide a more complete overview of the progress and potential directions for future research. This paper aims to provide such a survey, covering recent progress in text-to-SQL research, including datasets, methods, and evaluation, as well as potential future directions for research in this area.

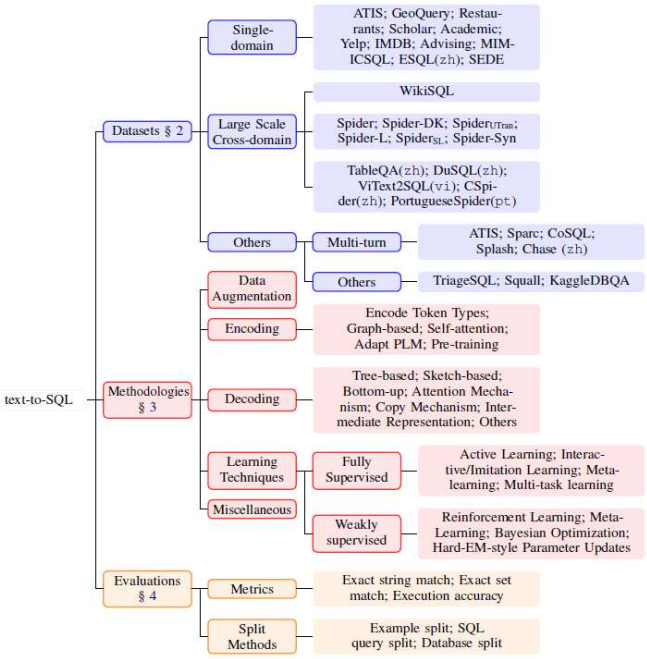

Table 1: Topology for text-to-SQL. Format adapted from Deng et al., 2022

II. DATASETS

Single-Domain Text-to-SQL datasets and Large Scale Cross-Domain Text-to-SQL datasets are two main datasets for text-to-SQL datasets.

A. Single-Domain Datasets

Single-domain datasets refer to text-to-SQL datasets that focus on a specific domain or topic. Examples of single-domain datasets, which focuses on generating SQL queries from questions about tables.

Moreover, Single-domain text-to-SQL datasets are typically designed to collect question-SQL pairs for a single database in a specific domain or topic, such as Yelp and IMDB(Deng et al., 2022) (These are two different datasets that contain questions and SQL queries related to a restaurant review website Yelp and a movie review website IMDB), Advising (This is another single-domain dataset containing questions and SQL queries related to a university advising system. The dataset includes information about courses, prerequisites, and graduation requirements), SEDE (This is a large-scale multi-domain dataset containing questions and SQL queries related to various Stack Exchange communities, including programming, statistics, and health), MIMICSQL

(Deng et al., 2022) (This is a single-domain dataset containing questions and SQL queries related to a medical database, including information about patient records, diagnoses, and treatments). These datasets shown in the Table 1 are useful for training and evaluating text-to-SQL systems on specific domains, and they have been used to develop and evaluate a wide range of approaches, including rule-based methods, neural network models, and hybrid models that combine both.

For Single-domain datasets, they often provide a good starting point for developing more complex cross-domain or zero-shot text-to-SQL models. However, they may not generalize well to other domains or may require significant retraining or adaptation to work with different databases or schemas.

B. *Large Scale Cross-domain Datasets*

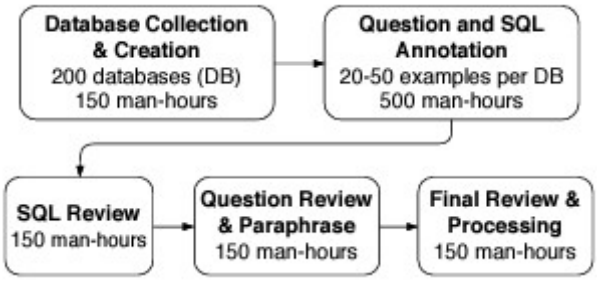

Fig. 2. The annotation process of our Spider corpus adapted from Yu et al., 2018c

WikiSQL (Zhong et al., 2017) is a large-scale text-to-SQL dataset that was proposed to evaluate the performance of text-to-SQL models. The dataset was created by extracting tables from Wikipedia and generating questions based on the contents of these tables. The dataset contains 80,654 natural language utterances paired with corresponding SQL queries. Each question in the dataset is associated with a table and a set of SQL queries that can be used to retrieve information from that table. The queries in WikiSQL(Zhong et al., 2017) are relatively simple, and they only involve a single table. However, the dataset is much larger than previous text-to-SQL datasets and has been used to evaluate a wide range of text-to-SQL models, including neural network models and rule-based systems. The WikiSQL(Zhong et al., 2017) dataset has become a standard benchmark for evaluating the performance of text-to-SQL models, particularly those that use deep learning techniques.

However, the SQL queries in WikiSQL(Zhong et al., 2017) are relatively simple, and each query only involves a single table. (Yu et al., 2018c) proposed that Spider is another large-scale dataset that contains 10,181 natural language questions paired with SQL queries across 138 databases, covering a wide range of domains such as academic publications, geographic locations, and recipes. The SQL queries in Spider (Yu et al., 2018c) are more complex than those in WikiSQL(Zhong et al., 2017) and often involve multiple tables and subqueries. These large-scale datasets are useful for evaluating the performance of text-to-SQL models at scale and for comparing different approaches for handling cross-domain text-to-SQL tasks.

There are a few more large-scale text-to-SQL datasets in different languages besides the ones that have been mentioned. For example, there is a dataset called CoSQL, which is a large-scale corpus for complex text-to-SQL tasks that involve multiple tables and complex SQL queries. CoSQL contains 30,000+ question-SQL pairs over 2,000 tables in 6 domains, and it is designed to evaluate the ability of models to handle complex queries that require reasoning over multiple tables. Another dataset is called SParC, which stands for "Semantic Parsing with SQL as a Foreign Language". SParC is a large-scale text-to-SQL dataset that contains 4,298 question-SQL pairs over 200 databases in 138 domains, and it is designed to evaluate the ability of models to handle complex, cross-domain queries. There is also a dataset called DROP, which stands for "DuoRC-based Reading Comprehension with Out-of-domain Papers". DROP is a large-scale reading comprehension dataset that involves answering questions over tables, and it contains 20,000+ questions over 96 tables in diverse domains. While DROP is not explicitly a text-to-SQL dataset, it does require models to generate SQL queries to answer the questions.

It is worth noting that human translation is generally more accurate than machine translation, and using these human-annotated datasets can improve the performance of text-to-SQL models for different languages. However, building large-scale text-to-SQL datasets in multiple languages is still a challenging task, and more efforts are needed to create high-quality datasets for different languages and domains.

C. *Other Datasets*

There are also context-dependent text-to-SQL datasets available in the literature. These datasets are designed to evaluate the ability of text-to-SQL models to handle queries that require reasoning over context, such as questions that involve co-reference resolution, temporal reasoning, or spatial reasoning. There are also datasets that focus specifically on spatial reasoning, such as the GeoQuery dataset. GeoQuery includes queries that involve spatial relations between objects, such as "What is the capital of the state that borders Lake Erie?" Context-dependent text-to-SQL datasets provide a more realistic evaluation of text-to-SQL models, as they reflect the types of queries that users might ask in real-world scenarios.

Dataset	# Q	# SQL	# DB	# Domain	# Table /DB	ORDER BY	GROUP BY	NESTED	HAVING
ATIS	5,280	947	1	1	32	0	5	315	0
GeoQuery	877	247	1	1	6	20	46	167	9
Scholar	817	193	1	1	7	75	100	7	20
Academic	196	185	1	1	15	23	40	7	18
IMDB	131	89	1	1	16	10	6	1	0
Yelp	128	110	1	1	7	18	21	0	4
Advising	3,898	208	1	1	10	15	9	22	0
Restaurants	378	378	1	1	3	0	0	4	0
WikiSQL	80,654	77,840	26,521	-	1	0	0	0	0
Spider	10,181	5,693	200	138	5.1	1335	1491	844	388

Table 2: Comparisons of text-to-SQL datasets. Spider is the only one text-to-SQL dataset that contains both databases with multiple tables in different domains and complex SQL queries. It was designed to test the ability of a system to generalize to not only new SQL queries and database schemas but also new domains adapted from Yu et al., 2018c.

III. METHODS

Rule-based and template-based approaches were mostly used in early text-to-SQL systems because they were appropriate for simple user queries and databases. These methods typically involve manually crafting rules or templates to map natural language utterances to corresponding SQL queries. However, these methods are limited in their scalability and generalization to complex queries (Yu et al., 2018c) and databases. Deep learning models can automatically learn effective representations of natural language and SQL queries, and can generalize to handle more complex queries and databases. As a result, many state-of-the-art text-to-SQL systems now employ neural network-based models that learn to map natural language utterances to corresponding SQL queries directly from data.

In the section of text-to-SQL methods, we will divide the deep learning methods employed in text-to-SQL research into five categories: Data Augmentation, Encoding, Decoding, Learning Techniques, Miscellaneous.

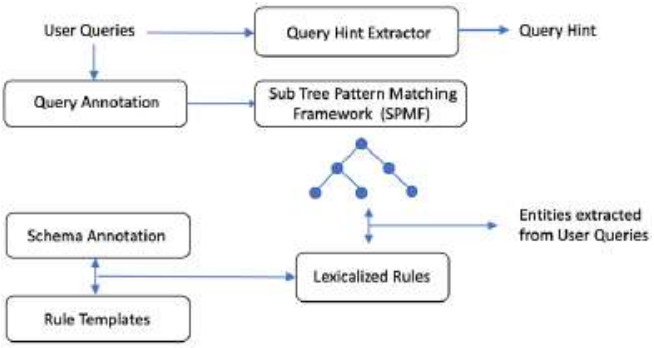

Fig. 3. Rule-based Natural Language Processing pipeline. The Query Hint Extractor module extracts query hint only if it is needed to answer the user query adapted from Yeo et al., 2022.

1. DATA AGUMENTATION

Data augmentation (Deng et al., 2022) is a useful technique in text-to-SQL for several reasons. Firstly, it can help the model handle complex or unseen questions by generating additional training examples. This is particularly useful when the amount of annotated data is limited or when the model is expected to generalize to new, unseen questions. Secondly, data augmentation can help the model achieve state-of-the-art performance with less supervised data by generating more diverse training examples. This is important because annotated data can be expensive to obtain, and it may not always be feasible to collect a large amount of high-quality training data. Finally, data augmentation can help the model attain robustness towards different types of questions by generating examples that cover a wide range of query structures and database schemas. This is important because real-world applications often involve complex queries (Yu et al.,2018c) and diverse data sources.

Paraphrasing techniques are a kind of typical data augmentation techniques can include using synonyms, reordering words, or changing sentence structure while preserving the same meaning. (Deng et al., 2022) Filling pre-defined templates involves replacing certain slots or keywords in a template with different values to generate new questions. For example, a template for a question about restaurants might be "What are the top [number] [cuisine] restaurants in [city]?" where [number], [cuisine], and [city] are slots that can be filled with different values to generate new questions.

Neural models can be used to generate natural language utterances for SQL queries that have already been labeled, effectively increasing the size of the dataset. This is often referred to as "data expansion" or "data amplification". (Deng et al., 2022) One approach for data expansion is to use a neural text generation model, such as a sequence-to-sequence model, to generate new utterances for SQL queries. Another approach is to use back-translation, where the labeled utterances are translated into another language using a machine translation system and then translated back into the original language, effectively generating new utterances. There is one example of using a pre-trained language model like T5 for data augmentation in text-to-SQL. By fine-tuning T5 on a task of generating natural language from SQL queries, the model can learn to generate more diverse and complex natural language for SQL queries that are not present in the original dataset. This can help improve the performance of text-to-SQL models on unseen data. The synthesized data can be used to train and evaluate text-to-SQL models.

The quality of the augmented data is crucial because the models learn from the augmented data as well as the original data, and if the augmented data is of low quality, it can introduce noise and errors into the model. This can result in decreased accuracy and poorer performance on unseen data. Therefore, it is important to carefully evaluate and filter the augmented data to ensure that it is of high quality and beneficial for the model's training. Some recent works have used a hierarchical approach to generate high-quality data for text-to-SQL models. In this approach, a SQL query is first converted to an intermediate representation (such as a semantic parse tree), and then a natural language question is generated from this intermediate representation. This ensures that the generated question is syntactically and semantically correct, and is more likely to be a valid question that a human user would ask. The intermediate representation can also be used to guide the question generation process and enforce constraints on the generated questions.

Some recent studies have proposed to incorporate a latent variable in their SQL-to-text generation model which can generate more diverse and plausible natural language questions for the same SQL query. One example is the work by Shi et al. (2021), which proposed a method called LAVA (Latent-variable Augmented Variational Autoencoder) (Deng et al., 2022) to generate more diverse and natural language questions for SQL queries. LAVA uses a variational autoencoder with a latent variable to generate diverse natural language questions while maintaining the correctness of the corresponding SQL queries. The method was evaluated on the Spider dataset and achieved state-of-the-art performance on both accuracy and diversity metrics.

Some recent studies have proposed to incorporate a latent variable in their SQL-to-text generation model which can generate more diverse and plausible natural language questions for the same SQL query. One example is the work by Shi et al. (2021), which proposed a method called LAVA (Latent-variable Augmented Variational Autoencoder) to generate more diverse and natural language questions for SQL queries. LAVA uses a variational autoencoder with a latent variable to generate diverse natural language questions while maintaining the correctness of the corresponding SQL queries. The method was evaluated on the Spider dataset (Yu et al., 2018c) and achieved state-of-the-art performance on both accuracy and diversity metrics. To mimic the informal query behavior of end users, several researchers have compressed and simplified questions to add to the WikiSQL dataset (Zhong et al., 2017) . The idea is that end-users may not always ask questions in a formal or structured way, and so the models need to be robust to different variations of questions. The simplified and compressed questions are created by replacing certain phrases or words with their simpler or more common equivalents, or by removing unnecessary words or phrases. This results in a larger and more diverse dataset that can better reflect the real-world query behavior of end-users. A probabilistic context-free grammar (PCFG) has been suggested by several academics as a way to describe the structure of SQL queries and produce artificial data. The PCFG is used to generate parse trees for SQL queries, and each node in the parse tree corresponds to a specific SQL

operation. By sampling from the PCFG, it is possible to generate a large number of diverse and complex SQL queries. These synthetic queries can be used to augment existing datasets and improve the robustness of text-to-SQL models.

2. ENCODING

Encoding refers to the process of transforming input data, such as natural language questions, into numerical representations that can be processed by machine learning models. (Deng et al., 2022) In the context of text-to-SQL, encoding is used to transform the input question into a vector representation that can be used as input to the neural network. There are various encoding techniques used in text-to-SQL research, including word embedding-based methods, such as Encode type, Graph-based, Self-attention, Adapt PLM, Pre-training and so on.

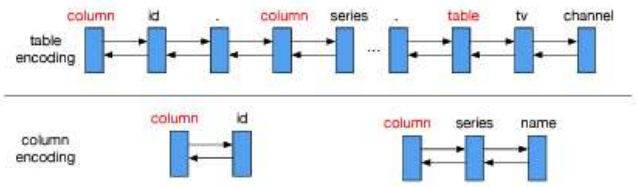

Fig. 4. An example of table encoding and column encoding adapted from Wang et al., 2020b.

Encoding token type is a technique used in text-to-SQL research to represent the different types of tokens in the input sequence. (Yu et al., 2018c) This is particularly useful when dealing with structured data, where there may be different types of entities, attributes, and keywords that need to be encoded differently.

For example, in a SQL query, keywords such as "SELECT" and "FROM" have a different meaning and function compared to table names and column names. By encoding the token type information, the model can learn to differentiate between these different types of tokens and incorporate this information into its representation of the input sequence. One common way to encode token types is to use additional embeddings or features for each token that indicate its type. These embeddings or features can be learned during training, or they can be pre-defined based on the task and the domain.

Graph-based methods refer to the approach of modeling text and SQL queries as graphs, where nodes represent words or sub-components of queries, and edges represent the relationships between them.

One example of a graph-based encoding method is the Graph Convolutional Networks (GCN) approach (Bogin et al. 2019a), which represents text and SQL queries as graphs and applies convolutional operations on the graph structure to capture the local dependencies between nodes. Another example is the SyntaxSQLNet (Bogin et al. 2019a), which represents the SQL query as a graph by parsing it into an Abstract Syntax Tree (AST) and the natural language question as a sequence. Then, it applies a graph convolution operation to extract the important information from both the AST and the sequence, and finally, feeds the extracted information to an SQL generation module.

Graph-based methods (Wang et al., 2020a) have shown promising results in text-to-SQL tasks as they can capture complex dependencies between different parts of the input and output sequences.

Self-attention is a type of neural network architecture commonly used in natural language processing (NLP) tasks, including text-to-SQL. (Wang et al., 2020a) In self-attention, the input sequence is mapped to a sequence of query-specific vectors, and each vector attends to every other vector in the sequence to obtain a weighted sum of the vectors, which represents the output. (Deng et al., 2022) This allows the model to capture dependencies between any two positions in the input sequence, and has been shown to be effective in capturing long-range dependencies in NLP tasks. The most well-known model that uses self-attention is the Transformer, which has achieved state-of-the-art performance in a wide range of NLP tasks, including text-to-SQL.

Adapt PLM refers to adapting pre-trained language models (PLMs) for the text-to-SQL task. PLMs, such as BERT, GPT, and RoBERTa, are pre-trained on large amounts of text data and have shown impressive performance in various natural language processing tasks. In the context of text-to-SQL, researchers have fine-tuned PLMs on text-to-SQL datasets to improve their performance on this task. Adapt PLM methods typically involve adding additional layers to the pre-trained PLM model and fine-tuning these added layers on text-to-SQL data. (Deng et al., 2022) The added layers can be encoder or decoder components, or a combination of both. The advantage of Adapt PLM methods is that they can leverage the knowledge learned by pre-trained models on a large amount of text data and transfer it to the text-to-SQL task, resulting in improved performance with less data.

Pre-training refers to the process of training a model on a large amount of data and then fine-tuning it for a specific downstream task. In text-to-SQL research, pre-training has been used to improve the performance of models on various datasets.

One popular pre-training approach is to use a language model, such as BERT or GPT, to generate representations for natural language text. The pre-trained language model is then fine-tuned on a downstream text-to-SQL task, such as WikiSQL (Zhong et al., 2017) or Spider (Yu et al., 2018c), by adding task-specific layers to the model and training them on the task data.

Another pre-training approach is to use a pre-trained semantic parser, such as Seq2SQL or SQLova, to generate SQL queries for a large corpus of natural language text. The pre-trained semantic parser is then fine-tuned on a downstream text-to-SQL task by adding task-specific layers and training them on the task data.

Pre-training has been shown to improve the performance of text-to-SQL models on various datasets and reduce the amount of labeled data needed for training.

3. DECODING

There are various methods proposed for decoding in text-to-SQL research, like Tree-based, Sketch-based, Bottom-up, Attention Mechanism, Copy Mechanism, Intermediate Representations and others.

Tree-based decoding methods refer to the process of generating a SQL query by constructing a tree structure. (Dong and Lapata, 2016) These methods involve breaking down the SQL query generation process into a series of hierarchical decisions, where each decision corresponds to a node in a tree structure. The tree structure is constructed dynamically as the model generates the SQL query, and the

choices made at each decision point depend on the context and the previously made decisions. Tree-based decoding methods have been used in several text-to-SQL models, including the popular Seq2Seq (Dong and Lapata, 2016) with Attention model, to improve the accuracy and efficiency of SQL query generation. These methods have been shown to be effective in capturing the hierarchical nature of SQL queries and the dependencies between different parts of the query.

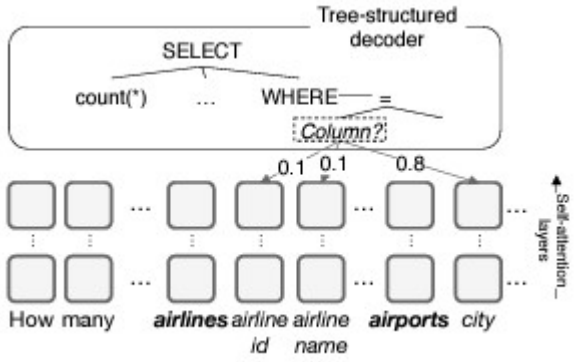

Fig. 5. Choosing a column in a tree decoder adapted from Wang et al., 2020a

Sketch-based methods refer to techniques that first generate a sketch or template of the SQL query and then fill in the details to obtain the final SQL query. This approach allows the model to focus on the high-level structure of the query, which is relatively easier to learn, while leaving the low-level details to be filled in later.

One example of a sketch-based method is the SQLova model (Dong and Lapata, 2016), which uses a sequence-to-sequence approach to generate a sketch of the SQL query and then fills in the details using a copy mechanism. The model first generates a sketch of the query consisting of a sequence of template tokens that specify the high-level structure of the query, such as the selection and aggregation operations. The model then generates the final SQL query by copying and modifying the input tokens based on the generated sketch.

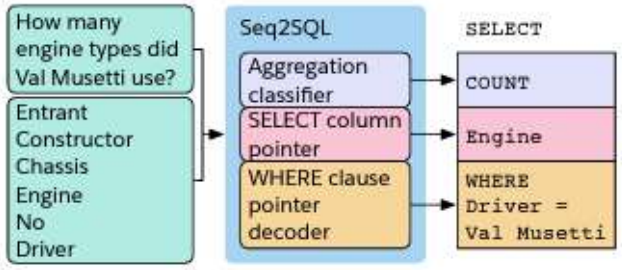

Fig. 6. Choosing The Seq2SQL model has three components, corresponding to the three components, corresponding to the three parts of a SQL query(right). The input to the model are the question (top left) and the tale column names (bottom left) adapted from Zhong et al., 2017.

Another example is the Seq2SQL model, which uses a similar approach but generates a sequence of SQL query components, such as selection, aggregation, and join operators, and then assembles them into a complete SQL query. The model uses a copy mechanism to copy and modify the input table and column names as needed to generate the final query.

Bottom-up is a decoding approach that aims to generate SQL queries from the bottom up, by first predicting the columns to select and then predicting the associated aggregation function, grouping, and filter conditions. (Deng et al., 2022) This approach is different from the traditional top-down approach, which starts with predicting the high-level structure of the SQL query, such as SELECT, WHERE, and GROUP BY clauses, and then fills in the details. The bottom-up approach has been shown to be effective for complex SQL queries, where the structure is less predictable and can vary greatly depending on the specific task.

The attention mechanism is a neural network component that allows the model to selectively focus on different parts of the input sequence when producing each output element. It was originally proposed for machine translation tasks, but has since been widely used in many NLP tasks, including text-to-SQL.

In the context of text-to-SQL, attention mechanisms (Dong and Lapata, 2016) can be used to help the model align the input question and the corresponding SQL query during the decoding process. Specifically, the model can use attention weights to determine which words in the input question are most relevant to the current output element being generated, and use that information to guide the SQL generation process.

There are several different types of attention mechanisms, including additive attention, dot-product attention, and multi-head attention, among others. Each type has its own strengths and weaknesses, and the choice of attention mechanism often depends on the specific task and data at hand.

The copy mechanism is a technique used in sequence-to-sequence models that allows the model to selectively copy input tokens to the output sequence instead of always generating new tokens. (Wang et al., 2020b) This is particularly useful in tasks like text-to-SQL, where some of the SQL tokens may directly correspond to input tokens.

In the context of text-to-SQL, the copy mechanism allows the model to copy column names, table names, or values directly from the input question to the SQL query instead of generating them from scratch. This not only improves the accuracy of the model but also makes the generated SQL queries more interpretable and explainable.

The copy mechanism works by computing a copy probability for each input token. (Wang et al., 2020b) The copy probability measures the likelihood that a particular input token should be copied to the output sequence. During decoding, the model can use the copy probability to determine whether to generate a new token or copy an input token.

The copy mechanism is often combined with other techniques like attention mechanism and pointer networks to improve the accuracy and effectiveness of text-to-SQL models.

Intermediate representations in text-to-SQL refer to the hinter mediate stages, which can facilitate the decoding process. These representations can be learned using various techniques such as sequence-to-sequence models, graph-based neural networks, and semantic parsers.

For example, the SQLNet model (Bogin et al. 2019a) introduced an intermediate representation called the schema embedding, which encodes the table schema information and allows the model to better understand the relationship between tables and columns during the decoding process. Similarly, the TreeSQL model uses a syntax tree as an intermediate representation to capture the structural information of the

input SQL query, making it easier for the model to generate a valid and semantically correct SQL query.

Other models, such as the EditSQL model, use a hybrid representation that combines both natural language and SQL tokens into a single sequence, allowing the model to capture the dependencies and interactions between them more effectively. Overall, intermediate representations can help text-to-SQL models to achieve better performance by providing additional context and structure during the encoding and decoding stages.

4. Learning Techniques

Learning Techniques in text-to-SQL research can be broadly classified into fully supervised and weakly supervised techniques.

Fully supervised learning techniques refer to the training process where both the input text and corresponding output SQL query are available during training. This is the most common approach in text-to-SQL research, as it allows for direct optimization of the end-to-end performance of the model.

Fully supervised learning techniques include traditional sequence-to-sequence models (Wang et al., 2020b), attention-based models, and transformer-based models. These models learn to map the input text to the output SQL query by optimizing a loss function, such as cross-entropy or sequence-to-sequence loss.

One popular approach for fully supervised learning in text-to-SQL is to use a hybrid model that combines both sequence-to-sequence and sequence-to-set models. The sequence-to-sequence model generates the SQL query one token at a time, while the sequence-to-set model predicts the selection, aggregation, and filtering operations as sets of labels.

Weakly supervised learning techniques are used when there is a lack of fully labeled data for training. These techniques aim to learn from partially labeled data or even unlabeled data, with the goal of achieving comparable performance to fully supervised methods.

IV. Evaluations

1. Metrics

Early works on text-to-SQL evaluation used database querying results and exact string matching as the evaluation metrics. (Deng et al., 2022) However, these metrics have limitations as execution accuracy can create false positives for semantically different SQL queries even if they yield the same execution results, and the exact string match can be too strict as two different strings can still have the same semantics.

However, the exact set match (ESM) metric compares the set of clauses in the predicted SQL query with the set of clauses in the ground-truth SQL query, rather than comparing the entire SQL query as a string. This approach is more flexible and allows for some variation in the structure of the SQL query while still ensuring that the query is semantically correct. The ESM metric has become a widely used evaluation metric in text-to-SQL research, especially on the Spider dataset (Yu et al., 2018c).

Early datasets for the text-to-SQL task usually followed the standard practice of randomly splitting the data into training, development (or validation), and test sets to evaluate the performance of different models. (Yu et al., 2018c) This is a common practice in machine learning and natural language processing tasks.

The SQL query split is a more rigorous evaluation setup that helps to evaluate the generalization ability of text-to-SQL models. By ensuring that no SQL query appears in more than one set, it tests the model's ability to handle unseen queries that have similar structures to the ones it has seen in the training data. This approach is particularly useful for domains with a large number of possible queries, where a random split may not guarantee sufficient coverage of the query space in the test set.

Metrics	Datasets	Errors
Naiive Execution Accuracy	GeoQuery, IMDB, Yelp, WikiSQL, etc	False positive
Exact String Match	Advising, WikiSQL, etc	False negative
Exact Set Match	Spider	False negative
Test Suite Accuracy (execution accuracy with generated databases)	Spider, GeoQuery, etc	False positive

Table 3: The summary of metrics, datasets that use these metrics, and their potential error cases adapted from Deng et al., 2022.

V. Discussion and Future Directions

Text-to-SQL systems have made significant progress in recent years, but there are still several challenges that must be addressed to improve their performance and usability. Some of the main challenges and future directions for text-to-SQL systems are:

Handling Complex Queries: One of the main challenges of text-to-SQL systems is handling complex queries that involve multiple tables and complex SQL expressions. Future systems need to be able to handle these complex queries more effectively, possibly by combining multiple techniques and approaches.

Improving Data Augmentation: Data augmentation is a key technique in text-to-SQL systems, but it can be challenging to generate high-quality synthetic data that accurately reflects the real-world distribution of natural language inputs. Future research needs to focus on improving the quality and diversity of data augmentation techniques to improve the robustness of text-to-SQL models.

Adapting to New Domains and Languages: Text-to-SQL systems are currently limited to specific domains and languages, and it can be challenging to adapt them to new domains and languages. Future systems need to be more flexible and adaptable, possibly by incorporating domain-specific knowledge and using transfer learning techniques.

Improving Explainability: Text-to-SQL systems can be difficult to interpret and debug, particularly when they make errors or produce unexpected outputs. Future systems need to be more transparent and explainable, possibly by incorporating techniques from explainable AI and natural language processing.

Addressing Bias and Fairness: Text-to-SQL systems, like all machine learning systems, can be biased and unfair, particularly if they are trained on biased or unrepresentative

data. Future systems need to be designed with fairness and bias mitigation in mind, possibly by incorporating techniques such as counterfactual analysis and data auditing.

Overall, the future of text-to-SQL systems lies in developing more flexible, robust, and explainable models that can handle complex queries, adapt to new domains and languages, and address issues of bias and fairness. Achieving these goals will require continued research and innovation in a range of areas, including data augmentation, encoding, decoding, learning techniques, and miscellaneous techniques.

REFERENCES

[1] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. *RATSQL: Relation-aware schema encoding and linking for text-to-SQL parsers.* In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7567–7578, Online. Association for Computational Linguistics.

[2] Huajie Wang, Mei Li, and Lei Chen. 2020b. PG-GSQL: *Pointer-generator network with guide decoding for cross-domain context-dependent text-to-SQL generation.* In Proceedings of COLING-2022, the 28th International Conference on Computational Linguistics, pages 370–380, Barcelona, Spain (Online). Association for Computational Linguistics.

[3] Li Dong and Mirella Lapata. 2016. *Language to logical form with neural attention*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 33–43, Berlin, Germany. Association for Computational Linguistics.

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

[5] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. *Seq2SQL: Generating structured queries from natural language using reinforcement learning.* ArXiv preprint, abs/1709.00103.

[6] 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. 2018c. *Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task.* In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

[7] Deng, N., Chen, Y., Zhang, Y.: *Recent advances in text-to-SQL: a survey of what we have and what we expect.* In: Proceedings of the 29th International Conference on Computational Linguistics, pp. 2166–2187. International Committee on Computational Linguistics, Gyeongju, Republic of Korea (2022)

[8] Hangu Yeo, Elahe Khorasani, Vadim Sheinin, Irene Manotas, Ngoc Phuoc An Vo, Octavian Popescu, Petros Zerfos. (2022). *Natural Language Interface for Process Mining Queries in Healthcare.* 2022 IEEE International Conference on Big Data (Big Data).